Parallel kinetic schemes for conservation laws, with large time steps

Pierre Gerhard, Philippe Helluy, Victor Michel-Dansac, Bruno Weber

PARALLEL KINETIC SCHEMES FOR CONSERVATION LAWS, WITH LARGE TIME STEPS

PIERRE GERHARD, PHILIPPE HELLUY, VICTOR MICHEL-DANSAC, BRUNO WEBER

Abstract. We propose a new parallel Discontinuous Galerkin method for the approximation of hyperbolic systems of conservation laws. The method remains stable with large time steps, while keeping the complexity of an explicit scheme: it does not require the assembly and resolution of large linear systems for the time iterations. The approach is based on a kinetic representation of the system of conservation laws previously investigated by the authors [14, 5, 15, 19, 27]. In this paper, the approach is extended with a subdomain strategy that improves the parallel scaling of the method on computers with distributed memory.

1. Introduction

The Discontinuous Galerkin (DG) method is generally used to approximate hyperbolic systems of conservation laws, see for instance [9, 24, 31, 13] and included references. The DG method is well suited for parallel computations, and it is often used in the context of large scale simulations. However, the time step Δt of the DG method is limited by the Courant-Friedrichs-Lewy (CFL) condition, which takes the form

$$\Delta t \leq K \frac{h}{c},$$

where h is the diameter of the smallest cell in the mesh, c is the maximal wave speed of the system of conservation laws, and K is a mesh-independent constant. Unfortunately, this constant is often small, especially for high orders of approximation, which impedes the efficiency of the method. Such a restriction is often a problem because, in many applications, the time step imposed by the CFL condition is much smaller than time steps sufficient to ensure a good accuracy of the time integration.

To avoid such a restrictive condition, one possibility is to construct time-implicit schemes, which are free from a CFL condition, but which involve having to solve a linear system. However, the cost of inverting such a linear system can become prohibitive, see for instance [12, 33] and references therein. One can also build locally implicit schemes, but the associated CFL condition is still constrained by the size of the smallest cell in the interface between the explicit and the implicit regions, see [9, 18]. Indeed, it is not easy to perform an automatic partitioning whose interfaces would not contain small cells. Yet another approach is to use a local time stepping strategy, where more time steps are performed on the smallest cells than on larger cells, see for instance [22, 40, 2, 17]. In practice, the efficiency gained by using this method can be disheartening, since there are usually many small cells, on which lots of computations are still needed. A common drawback of the above methods is that, in the context of meshes with cells of uniform size, standard explicit methods will not be outperformed.

In the context of the finite difference method, recent work [23, 25, 26, 44] has proposed an unconditionally stable time integration method. This method is based on the filtering of high frequencies in the solution in order to recover a less restrictive CFL condition. This approach relies on the computation of a few eigenvectors of the spatial operator, associated to low frequency eigenvalues.

In [27], we have proposed an unconditionally stable method whose complexity (both in terms of computation time and storage) is in $\mathcal{O}(n)$, with n the number of degrees of freedom in the spatial approximation. As a consequence, this method has the same complexity as an explicit scheme. It relies on a vectorial kinetic interpretation of the system of conservation laws based on [8, 3]. The whole algorithm then reduces to the resolution of independent transport equations coupled through relaxation source terms. This whole system can be approximated implicitly while retaining the complexity of an explicit scheme. Indeed, the implicit resolution of the transport equation with a DG scheme can be performed with a downwind visiting of the mesh in the direction of the transport velocity, and the implicit relaxation source terms can be applied locally. In addition,

2020 *Mathematics Subject Classification.* 65M60, 65Y05.

Key words and phrases. discontinuous Galerkin, kinetic approximation, CFL-less, unconditional stability, parallelization.

the algorithm is parallelizable, but dependencies in the computation reduce the parallel efficiency: the scaling is not optimal when considering too large a number of threads.

In this paper, we propose an improvement of the aforementioned kinetic approach, which relaxes the constraints in the parallel algorithm in order to improve the scaling. The main idea is to perform a decomposition of the computational domain. The transport equations are then solved with an iterative algorithm on each subdomain, while the relaxation source terms are treated like in the non-partitioned case. Hence, the main change compared to the non-partitioned case consists in having to solve the transport equation multiple times, instead of once, per time iteration. In fact, thanks to the structure of the DG transport solver, we can prove that this iterative algorithm converges to the solution of the fully implicit solver in at most three iterations, under the non-restrictive CFL condition

$$\Delta t \leq \tilde{K}\frac{H}{c},$$

where this time H is the diameter of the smallest subdomain, and $\tilde{K}$ still is a mesh-independent constant. Moreover, this subdomain decomposition allows a great improvement in the parallel scaling of the whole method, because it relaxes dependencies in the transport solver.

The goal of this paper is to present this strategy and to validate this approach, both by comparing it to established solvers and by performing a large simulation a real-life situation (the interaction of waves emitted by an antenna with a human body). The paper is organized as follows. In Section 2, we recall the vectorial kinetic approximation of conservation and balance laws. Then, the DG solver is presented in Section 3. Section 4 explains the thread-based parallel algorithm, while Section 5 is devoted to the subdomain decomposition algorithm. Finally, numerical results are proposed in Section 6. In Section 7, a conclusion concludes the paper.

2. Kinetic approximation of first order conservations laws

2.1. **Kinetic approximation.** In this paper, we are interested in the numerical approximation of a system of m conservation laws in dimension d, governed by:

$$\partial_t W + \sum_{i=1}^{d} \partial_i Q^i(W) = 0, \tag{2.1}$$

where the unknown is a vector $W(X,t) \in \mathbb{R}^m$ depending on the space variable $X = (x_1, \ldots, x_d) \in \Omega \subset \mathbb{R}^d$ and on the time variable $t \in \mathbb{R}$. For the partial derivatives, we have introduced the notation

$$\partial_i = \frac{\partial}{\partial x_i}, \quad \partial_t = \frac{\partial}{\partial t}.$$

We assume the system to be hyperbolic. To introduce this property, let $N = (N_1, \ldots, N_d) \in \mathbb{R}^d$ be an arbitrary space direction. The flux in direction N is then defined by

$$Q(W,N) = \sum_{i=1}^{d} N_i Q^i(W).$$

The Jacobian matrix of the flux $d_W(Q(W,N))$ is then supposed to be diagonalizable with real eigenvalues $\lambda_r(W,N)$, for $r \in \{1, \ldots, m\}$. By definition, this means that the system (2.1) is hyperbolic.

The numerical approximation of such systems is, in general, a difficult subject. One of the difficulties is that explicit schemes are subject to restrictive time step conditions. Implicit schemes do not suffer from time step conditions but require solving large sets of linear equations. In previous work (see [27] and included references), we have proposed a method, based on a kinetic approach, that avoids these constraints. We now recall the principles of this kinetic representation.

We consider a set of $d+1$ kinetic velocities $V_k \in \mathbb{R}^d$, $k \in \{0, \ldots, d\}$, associated to **vectorial** kinetic functions $F_k(W) \in \mathbb{R}^m$. Additional kinetic velocities and function could be introduced, but we here the minimum number of velocities, $d+1$, is considered for simplicity. The macroscopic data and the kinetic data are related by

$$W = \sum_{k=0}^{d} F_k.$$

We also define "Maxwellian" equilibrium functions $M_k(W) \in \mathbb{R}^m$ such that

$$W = \sum_{k=0}^{d} M_k(W).$$

The kinetic BGK representation then is given by transport equations with relaxation source terms [8, 3]:

$$\forall k \in \{0, \dots, d\}, \quad \partial_t F_k + V_k \cdot \nabla_X F_k = \frac{1}{\tau}\big(M_k(W) - F_k\big), \tag{2.2}$$

with τ a (small) relaxation time.

When the relaxation time τ goes to 0^+, the kinetic model (2.2) is formally equivalent to the initial system of conservation laws (2.1) as long as

$$W = \sum_{k=0}^{d} M_k(W) \quad \text{and} \quad \forall i \in \{1, \dots, d\}, \quad \sum_{k=0}^{d} V_k^i M_k(W) = Q^i(W). \tag{2.3}$$

Equations (2.3) constitute a linear system of size $m(d+1) \times m(d+1)$ whose unknowns are the Maxwellian functions $M_k(W)$. One expects this system to have a unique solution as soon as the set of kinetic velocities is well chosen.

Theoretical arguments show that the formal limit described above actually is the true limit, provided a so-called sub-characteristic condition is satisfied [8, 3]. This condition states that the kinetic velocities have to be greater than the largest wave speed of the underlying hyperbolic system:

$$\forall k \in \{0, \dots, d\}, \forall W \in \mathbb{R}^m, \forall N \in \mathbb{R}^d, \quad \|V_k\| > \max_r |\lambda_r(W, N)|.$$

We now present the algorithm we use in practice to solve the kinetic equations (2.2).

2.2. **Kinetic algorithm.** In practice, directly solving the BGK system (2.2) is difficult. It is usually better to split the equations into a transport step and a relaxation step.

At the initial time, we start with initial data $W(\cdot, 0)$. We have to choose kinetic vectors $F_k(\cdot, 0)$ such that $W = \sum_k F_k$. Obviously, this choice is not unique. A natural choice is to take $F_k(\cdot, 0) = M_k(W(\cdot, 0))$ for all $k \in \{0, \dots, d\}$.

Now, at each time step, to go from time t to time $t + \Delta t$, we adopt the following kinetic algorithm.

(1) Start with given kinetic data $F_k(\cdot, t)$ for all $k \in \{0, \dots, d\}$: thus $W(\cdot, t) = \sum_k F_k(\cdot, t)$.
(2) For each k in $\{0, \dots, d\}$, solve the free transport equation $\partial_t F_k + V_k \cdot \nabla_X F_k = 0$ for a duration of Δt. In the continuous description of the scheme, the free transport step can be solved exactly. It is given by shift operations

$$\forall k \in \{0, \dots, d\}, \quad F_k(X, t + \Delta t^-) = F_k(X - \Delta t V_k, t). \tag{2.4}$$

In practice we prefer to approximate this step by a Discontinuous Galerkin solver. It is described below, in Section 3.
(3) Define

$$W(\cdot, t + \Delta t^-) = \sum_{k=0}^{d} F_k(\cdot, t + \Delta t^-),$$

and take

$$W(\cdot, t + \Delta t^+) = W(\cdot, t + \Delta t^-) =: W(\cdot, t + \Delta t).$$

(4) Apply a relaxation, with parameter $\omega \in [1, 2]$:

$$\forall k \in \{0, \dots, d\}, \quad F_k(\cdot, t + \Delta t^+) = \omega M_k(W(\cdot, t + \Delta t^+)) + (1 - \omega) F_k(\cdot, t + \Delta t^-). \tag{2.5}$$

Remark. *Note that, even though $F_k(\cdot, 0) = M_k(W(\cdot, 0))$ at the initial time $t = 0$, this is no longer the case for subsequent iterations. Indeed, this would not be desirable, as it would lead to a scheme with first order accuracy in time.*

Let us point out that the approximation of the conservative data is continuous in time:

$$W(\cdot, n\Delta t^-) = W(\cdot, n\Delta t^+),$$

while the approximation of the kinetic data is discontinuous at times $t_n = n\Delta t$: in general,

$$F_k(\cdot, n\Delta t^-) \neq F_k(\cdot, n\Delta t^+).$$

In the presence of source terms on the conservation law, the conservative data are no longer continuous in time. The treatment of source terms is discussed in Section 2.3, the next section.

In the relaxation step (2.5), the parameter ω plays an important role. A natural choice would be to take $\omega = 1$. This corresponds to a projection of the kinetic data on the Maxwellian state at the end of each time step. This choice presents many interesting features: it leads to an entropy dissipative, first order scheme. In addition, it is unconditionally stable with respect to the time step Δt. It enters the large category of kinetic schemes. It has been observed a long time ago that these schemes are free of CFL conditions, see for instance [10, 41]. However, this interesting property is rarely exploited in practical applications. Another choice corresponds to taking $\omega = 2$, leading to an over-relaxation procedure. This choice, and its consequences, are described in Section 2.4.

2.3. **Handling source terms.** This whole method can be extended to balance laws, i.e., conservation laws with a source term, which take the form

$$\partial_t W + \sum_{i=1}^{d} \partial_i Q^i(W) = S(W).$$

We refer, for instance, to [15, 27], where the kinetic equations are given by

$$\forall k \in \{0, \dots, d\}, \quad \partial_t F_k + V_k \cdot \nabla_X F_k = G_k + \frac{1}{\tau}\left(M_k(W) - F_k\right),$$

with $G_k = (\nabla_W M_k(W))S(W)$ for all $k \in \{0, \dots, d\}$.

In presence of source terms, Item 3 of the kinetic algorithm is modified as follows.

(3) Define

$$W(\cdot, \Delta t^-) = \sum_k F_k(\cdot, \Delta t^-).$$

Solve the differential equation

$$\partial_t U(\cdot, t) = S(U(\cdot, t)), \tag{2.6}$$

with the initial condition

$$U(\cdot, 0) = W(\cdot, \Delta t^-).$$

Finally, take

$$W(\cdot, \Delta t^+) = U(\cdot, \Delta t).$$

In practice, the differential equation (2.6) is discretized with a Crank-Nicolson scheme, which reads

$$\frac{W(\cdot, \Delta t^+) - W(\cdot, \Delta t^-)}{\Delta t} = \frac{S(W(\cdot, \Delta t^+)) + S(W(\cdot, \Delta t^-))}{2}. \tag{2.7}$$

For more details, we refer to [27].

2.4. **Equivalent equation.** Because the first-order scheme is generally not accurate enough, it is often better to consider the over-relaxed choice $\omega = 2$. In this case, the scheme becomes second-order accurate. We briefly sketch the proof of this property. For more details, we refer to [15, 27].

During the computations, one expects that, for all $k \in \{0, \dots, d\}$, $F_k \simeq M_k(W)$, and therefore that $\sum_k V_k^i F_k \simeq Q^i(W)$. We thus introduce the **approximate flux** Z and the **flux error** Y, defined as follows:

$$Z^i \coloneqq \sum_{k=0}^{d} V_k^i F_k, \quad \text{and} \quad Y^i \coloneqq Z^i - Q^i(W) = \sum_{k=0}^{d} V_k^i \left(F_k - M_k(W)\right). \tag{2.8}$$

The whole kinetic algorithm is a functional operator $\mathcal{M}(\Delta t)$ that maps $(W(\cdot,0), Y^i(\cdot,0))$ to $(W(\cdot,\Delta t), Y^i(\cdot,\Delta t^+))$. The operator $\mathcal{M}$ is made of (linear) shift operations and (non-linear) local relaxations. In the (W, Y^i) variables, the relaxation operation (2.5) simply reads, arguing (2.8):

$$\begin{cases} \omega = 1 \implies F_k(\cdot,\Delta t^+) = M_k(W(\cdot,\Delta t)), \\ \qquad \implies Y^i(\cdot,\Delta t^+) = \displaystyle\sum_{k=0}^{d} V_k^i \left(F_k(\cdot,\Delta t^+) - M_k(W(\cdot,\Delta t))\right) = \sum_{k=0}^{d} V_k^i \left(M_k(W(\cdot,\Delta t)) - M_k(W(\cdot,\Delta t))\right) \\ \qquad \implies Y^i(\cdot,\Delta t^+) = 0, \\ \omega = 2 \implies F_k(\cdot,\Delta t^+) = 2M_k(W(\cdot,\Delta t)) - F_k(\cdot,\Delta t^-), \\ \qquad \implies Y^i(\cdot,\Delta t^+) = \displaystyle\sum_{k=0}^{d} V_k^i \left(F_k(\cdot,\Delta t^+) - M_k(W(\cdot,\Delta t))\right) = \sum_{k=0}^{d} V_k^i \left(M_k(W(\cdot,\Delta t) - F_k(\cdot,\Delta t^-)\right), \\ \qquad \implies Y^i(\cdot,\Delta t^+) = -Y^i(\cdot,\Delta t^-). \end{cases}$$

Therefore, the choice $\omega = 2$ induces fast oscillations of the flux error. For the forthcoming analysis, it is thus better to replace $\mathcal{M}$ with $\mathcal{M} \circ \mathcal{M}$.

In principle it is now easy, although tedious, to compute the equivalent equation of the kinetic algorithm. It consists in computing a Taylor expansion of

$$\frac{\mathcal{M}(\Delta t/2) - \mathcal{M}(-\Delta t/2)}{\Delta t},$$

with respect to Δt, up to order $\mathcal{O}(\Delta t^2)$.

During the calculation of the Taylor expansion, the term $X - \Delta t V_k$ in the shift operation (2.4) generates partial derivatives in space. In addition, because of symmetries, when $\omega = 2$, the even-order terms of the expansion vanish. And finally, the relaxation introduces non-linearities. We end up with a system of non-linear partial differential equations of first order in (W, Y^i). The calculations are tedious, but can be automated through a Computer Algebra System.

We illustrate the results obtained for $d = 1$, i.e., in one space dimension. In this case, we have $d+1 = 2$ kinetic velocities. We set $\lambda \in \mathbb{R}_+^*$ and we choose $V_0 = -\lambda$ as well as $V_1 = \lambda$. In this one-dimensional case, $W = F_0 + F_1$. The equivalent equation for $\omega = 2$ is then, in conservative variables and up to $\mathcal{O}(\Delta t^2)$:

$$\partial_t W + \partial_x Q(W) = 0.$$

We indeed recover the desired conservation laws. It is also possible to compute a second-order equivalent equation for Y:

$$\partial_t Y - d_W(Q(W))\partial_x Y = 0.$$

We observe that the system is hyperbolic and that the waves for W and Y move in opposite directions [19]. We emphasize that there is no assumption of smallness of Y. In practice, we indeed observe second-order accuracy, even when the initial Y is of order $\mathcal{O}(1)$. For $d = 1$, the second-order expansion is not sufficient to analyze the stability of the approximation. By analyzing the third-order term, it is, however, possible to prove stability under a sub-characteristic stability condition

$$\lambda \geq \max_{1 \leq i \leq m} |\lambda_i(W)|,$$

where $\lambda_i(W)$ are the eigenvalues of $d_W(Q(W))$, see [19].

We can also perform the calculations in the case $d = 2$. We then need $d+1 = 3$ kinetic velocities. We can take

$$V_k = \begin{pmatrix} \cos\left(\dfrac{2k\pi}{3}\right) \\ \sin\left(\dfrac{2k\pi}{3}\right) \end{pmatrix}$$

The equivalent equation on W for $\omega = 2$ at order $\mathcal{O}(\Delta t^2)$ is, of course, the system of conservation laws (2.1)

$$\partial_t W + \partial_1 Q^1(W) + \partial_2 Q^2(W) = 0.$$

Setting $A^i(W) = d_W(Q^i(W))$, the equation for Y is

$$\partial_t \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} + \begin{pmatrix} \frac{\lambda}{2} I - A^1(W) & 0 \\ -A^2(W) & -\frac{\lambda}{2} \end{pmatrix} \partial_1 \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} + \begin{pmatrix} 0 & -\frac{\lambda}{2} I - A^1(W) \\ -\frac{\lambda}{2} & -A^2(W) \end{pmatrix} \partial_2 \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \mathcal{O}(\Delta t^2).$$

We can prove that the equivalent system is hyperbolic if λ is large enough. In this case, the sub-characteristic condition arises from the analysis of the first order terms. To get a finer bound, there exist more sophisticated analyses, based on the entropy: we refer for instance to [8, 20].

3. Unconditionally stable DG approximations

In this section, we recall how to construct an unconditionally stable Discontinuous Galerkin (DG) approximation of the initial system of conservation laws (2.1). The reader is referred to [27], where this procedure is explained in detail.

The kinetic algorithm presented in Section 2 relies on transport steps and relaxation steps. The relaxation step is generally easy to implement at each interpolation point of the approximation. In addition, it is embarrassingly parallel.

The implementation difficulty of the kinetic algorithm lies in the transport step, given by Item 2. In this step, the shift operation (2.4) consists in solving $(d+1) \times m$ transport equations of the form

$$\partial_t f + V \cdot \nabla f = 0. \tag{3.1}$$

Note that V represents one of the $(d+1)$ kinetic velocities and that f represents one of the m dimensions of the $(d+1)$ kinetic unknowns.

If the computational domain has a simple shape and if the solution is computed on a structured Cartesian grid, it is natural to solve this transport equation by the characteristic method (2.4). With well-chosen time step Δt and kinetic velocities V_k, this approach leads to the so-called Lattice Boltzmann method, see for instance [6], and included references.

In a domain Ω with a complex geometry, or discretized with an unstructured grid, the characteristic method is no longer a good choice because it leads to difficulties such as instabilities or loss of the conservation property. In addition, the treatment of boundary conditions is not natural in this framework. Instead, in the unstructured case, we prefer to rely on an approximation of the shift operation (2.4), based on discretizing (3.1) in the DG framework. For a general presentation of the DG approach, we refer to the book of Hesthaven and Warburton [31]. The idea to solve a kinetic BGK model with a DG approximation of the transport step was already proposed in [42], but with an explicit scheme. The novelty of our approach is to adopt an implicit DG approximation, instead of an explicit one, in order to get rid of the CFL condition. It turns out that the implicit scheme is not more complicated to solve than the explicit scheme, because the matrix of the implicit step is triangular and can thus be solved in an explicit fashion. For the sake of completeness, we briefly describe the implicit DG approach to approximate solutions to (3.1). More details can be found in [5, 15, 27].

3.1. DG scheme.

We consider an unstructured mesh $\mathcal{M}$ of the computational domain Ω made of tetrahedral cells. On each cell, we define n_n basis functions $(\psi_j^L(X))_{j\in\{1,\dots,n_n\}}$. The transported function f is then approximated in cell L by a linear expansion on basis functions

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

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

We now write an implicit DG approximation scheme to compute the unknown coefficients $f_{L,j}^n$ at time t^n from the known coefficients $f_{L,j}^{n-1}$ at time t^{n-1}. For simplicity, we describe the case of an implicit first order Euler method. The strategy can be extended to other more accurate schemes, such as the Crank-Nicolson scheme (which we use in practice) or DIRK (Diagonally Implicit Runge-Kutta) approaches, see for instance [1, 36]. The DG scheme then reads as follows: for each cell L and each basis function ψ_i^L,

$$\int_L \frac{f_L^n - f_L^{n-1}}{\Delta t} \psi_i^L - \int_L f_L^n V \cdot \nabla \psi_i^L + \sum_{\alpha=1}^{n_f} \int_{\partial L_\alpha} \left(f_L^n (V \cdot N_\alpha)_+ + f_{R_\alpha}^n (V \cdot N_\alpha)_- \right) \psi_i^L = 0. \tag{3.2}$$

In this formula, n_f denotes the number of faces of cell L (for a tetrahedron, $n_f = 4$), ∂L_α denotes the part of the boundary of L where face α is located, and R_α denotes the neighboring cell along ∂L_α. The situation is depicted in Figure 3.1, in 2D for simplicity. In addition, we use standard notation for the upwind numerical flux:

$$(V \cdot N_\alpha)_+ = \max(V \cdot N_\alpha, 0) \quad \text{and} \quad (V \cdot N_\alpha)_- = \min(V \cdot N_\alpha, 0),$$

where the vector N_α is the unit normal vector on ∂L_α oriented from L to R_α.

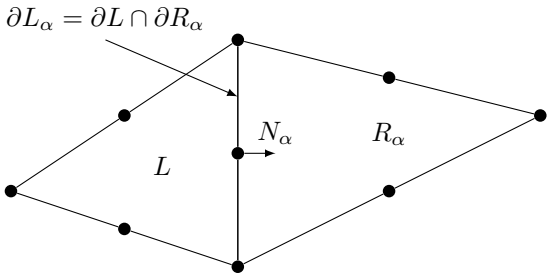

FIGURE 3.1. Notation for the Discontinuous Galerkin approximation.

3.2. **Downwind algorithm.** The scheme is implicit and it seems that one would need to assemble and solve a large linear system in order to compute f_L^n from f_L^{n-1}. However, we can exhibit an algorithm with explicit complexity, dubbed *downwind algorithm*, that solves efficiently – and in parallel – the set of equations (3.2). The method is described in detail in [14, 5, 15, 27]. In this manuscript, we only recall its major steps.

In the preprocessing phase, we construct a graph G from the mesh $\mathcal{M}$. Its nodes correspond to the cells of the mesh and its edges to the faces between cells. Each edge is then oriented with respect to the velocity V. Between two nodes L and R (corresponding to two cells), the edge is oriented from L to R if V is oriented from L to R, i.e. if V crosses the edge $\partial L \cap \partial R$ from L to R Because the velocity is constant, it is possible to prove that the graph G is direct and acyclic. It can thus be sorted in topological order, using Breadth-First Search (see [27] for a comparison between Breadth-First Search and Depth-First Search on such problems). Note that this preprocessing phase is executed only once at the beginning of the computations.

In the main computation phase, corresponding to the time loop, the linear system (3.2) is then solved by visiting the cells of the mesh in this topological order. The algorithm is parallel and its storage can be optimized: the solution can be replaced in memory during the computations, see Figure 3.2 for a example. After solving the linear system, the relaxation step, which is embarrassingly parallel, is applied.

4. THREAD-BASED NUMERICAL IMPLEMENTATION

We have implemented the kinetic algorithm in a code written in Rust. Rust is a recent programming language oriented toward security and efficiency. Most common bugs are avoided at compile time. For instance, memory leaks, segmentation faults, uninitialized data and race conditions are forbidden by the compiler. In addition, Rust proposes automatic parallelization tools through the `rayon` library, based on a work stealing strategy [37]. This library is particularly well suited to the parallel implementation of the downwind algorithm presented in Section 3.2. For more detail, we refer to [27]. Furthermore, meshes are generated using the `Gmsh` tool, described in [28].

The implementation presents a good scaling for a moderate number of threads, as shown in Table 1 from [27].

However, in Table 1, we also observe that the efficiency is not 100 %. Indeed, dependencies in the computations limit the parallel scaling of the downwind algorithm. For instance, in the mesh from Figure 3.2, it is clear that launching more than three threads is useless because the additional threads will have to wait for computations to be finished before starting to work. Similar behavior occurs for larger meshes, whose parallel regions (the cells which can be treated in parallel) are, on average, larger, but which contain small, efficiency-limiting parallel regions close to edges.

To address this issue, we propose in the next section a modification of the downwind algorithm, both to improve the parallel efficiency of the method, and to deal with the distributed-memory setting.

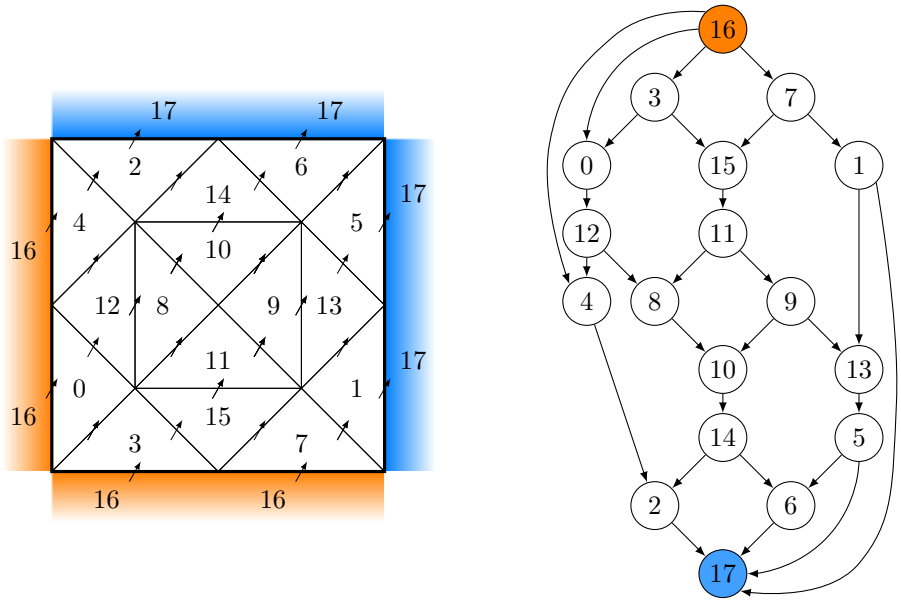

FIGURE 3.2. Example of a mesh $\mathcal{M}$ (left panel) and its associated graph G (right panel). The nodes of the graph correspond to the cells of the mesh. Two additional, fictitious nodes are considered: the upwind node (in orange) and the downwind node (in blue). The solution can be explicitly computed by following a topological ordering of a Direct Acyclic Graph (DAG) using Breadth-First Search, e.g. 3, 7, 0, 15, 1, *etc.* In addition, the parallel capabilities of the method are visible on the graph: first, cells 3 and 7 can be computed in parallel; then cells 0, 15 and 1 can be computed in parallel, *etc.*

TABLE 1. Multithread efficiency of the downwind algorithm executed on a server with an Intel Xeon E5-2680 v3 processor (24 physical cores, 2.50 GHz). The scalability is computed on coarse to fine meshes, with several refinement levels. We observe that the efficiency stalls at around 60 %.

refinement		it/s		µs/dof/it			
level	elements	serial	parallel	serial	parallel	scalability	heap
8	1808	72.58	346.1	0.425	0.089	4.769	11.85 MB
16	9199	11.34	102.2	0.569	0.063	9.012	42.50 MB
32	56967	1.698	20.19	0.664	0.056	11.89	266.5 MB
48	175138	0.531	7.753	0.718	0.049	14.60	808.9 MB
64	386806	0.236	3.579	0.747	0.049	15.17	1.777 GB
72	544030	0.165	2.531	0.765	0.050	15.34	2.515 GB

5. SUBDOMAIN PARALLELISM

As explained above, the downwind algorithm is parallelized with a work stealing thread-based algorithm. The parallel scaling is good for a few threads, but seems to be capped as the mesh becomes finer.

In order to provide better scaling capabilities, we now describe a subdomain strategy that relaxes the computation dependencies. The main idea is to apply the above time-implicit downwind algorithm in each subdomain, but with a time-explicit coupling between the subdomains, so as to relax the dependencies between regions. Because of the explicit coupling, it will become necessary to apply an iterative algorithm to compute the approximate solution in a stable way. The algorithm can be proved to converge in a finite number of iterations. In most configurations, three iterations are sufficient. Let us now describe the principles of this subdomain iterative algorithm.

As in Section 3, the main task is the resolution of the initial value problem for the transport equation:

$$\begin{cases} \partial_t f + V \cdot \nabla f = 0 & \text{for } X \in \Omega \times [0, \Delta t], \\ f(X, 0) = f^0(X) & \text{for } X \in \Omega. \end{cases}$$

5.1. **Iterative algorithm.** We assume that Ω is decomposed into a finite number of subdomains $(\Omega_i)_{i \in \{1,\dots,n_d\}}$. To simplify the presentation, we assume that Ω is either a periodic domain or the whole space domain, in order to avoid having to describe the boundary conditions. However, the approach is also valid when $\partial\Omega \neq \emptyset$.

We then denote by f_i the restriction of f to subdomain Ω_i, by $N_i(X)$ the outward normal vector on $\partial\Omega_i$, by $\mathcal{N}(\Omega_i)$ the subdomains neighboring Ω_i, and by $\partial\Omega_i^-$ the upwind part of the boundary of Ω_i:

$$\partial\Omega_i^- = \{X \in \partial\Omega_i \mid N_i(X) \cdot V < 0\}.$$

We initialize the algorithm by setting $f_i^0(X, t) = f_i^0(X)$. Thus, the initial iteration does not depend on time.

We then propose an iterative algorithm to compute the successive time-dependent iterations f_i^p in subdomain Ω_i, for $p \geq 1$. To compute f_i^p from f_i^{p-1}, we solve the following time-dependent boundary value problems:

$$\begin{cases} \partial_t f_i^p + V \cdot \nabla f_i^p = 0 & \text{for } X \in \Omega_i \text{ and } t \in (0, \Delta t), \quad \text{(5.1a)} \\ f_i^p(X, 0) = f_i^0(X) & \text{for } X \in \Omega_i, \quad \text{(5.1b)} \\ f_i^p(X, t) = f_j^{p-1}(X, t) & \text{for } X \in \partial\Omega_i^- \cap \partial\Omega_j, \ \forall \Omega_j \in \mathcal{N}(\Omega_i). \quad \text{(5.1c)} \end{cases}$$

We can then prove the following result.

Proposition 1. *let $\mathcal{L}$ be the maximal subdomain diameter. Under the condition*

$$\Delta t \leq \frac{\mathcal{L}}{|V|},$$

the above algorithm (5.1) converges to the exact solution in at most three iterations: $f_i^3 = f_i$.

Proof. The proof relies on the method of lines. It is briefly sketched in Figures 5.1 and 5.2. □

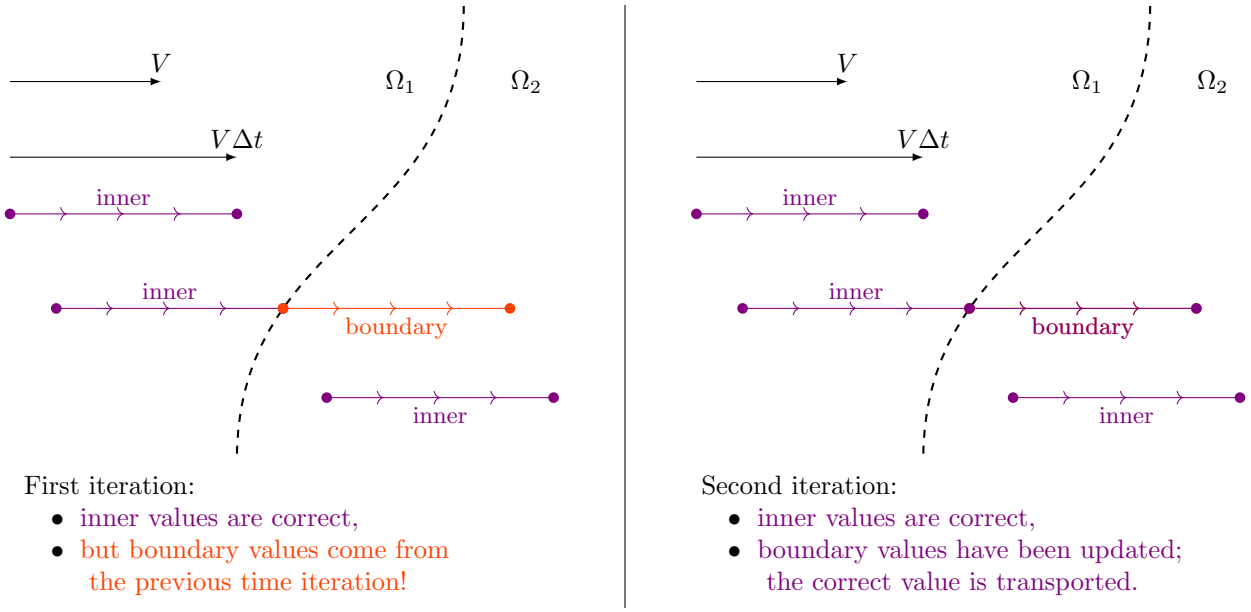

FIGURE 5.1. Subdomain algorithm, when the subdomain decomposition is aligned with the transport velocity. In this case, the iterative algorithm reaches the exact solution in at most two iterations. During the first iteration, in the left panel, the boundary values of the subdomains are updated. During the second iteration, in the right panel, the correct boundary values are transported. The purple color corresponds to the transport of a correct value, while the red color corresponds to the transport of a wrong value.

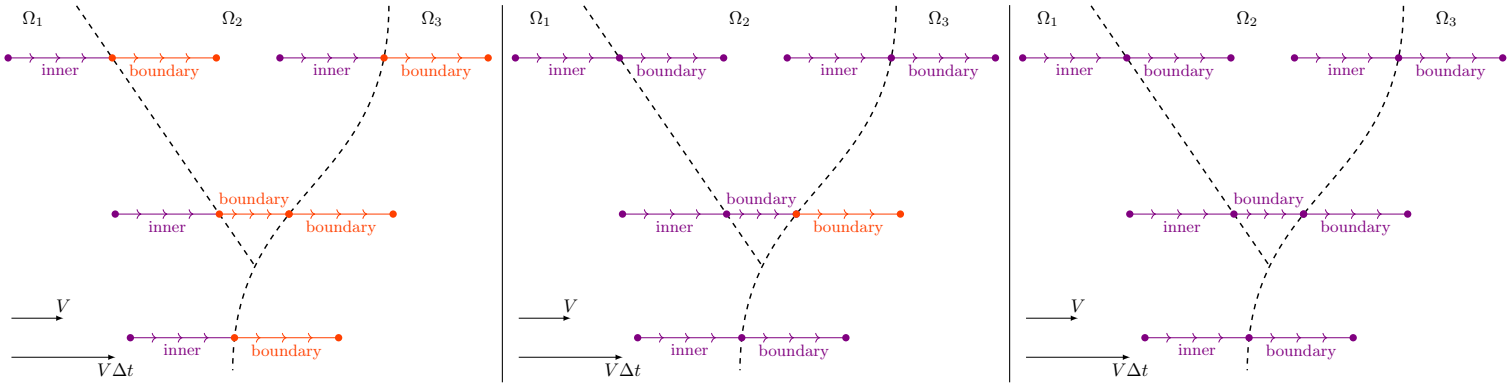

FIGURE 5.2. Subdomain algorithm, in a generic subdomain decomposition, with corners shared by several subdomains. In this case, the iterative algorithm reaches the exact solution in at most three iterations. First iteration, left panel: the boundary value on $\partial\Omega_2^-$ is updated. Second iteration, center panel: the boundary value on $\partial\Omega_3^-$ is updated. Third iteration, right panel: the correct boundary value is transported. The purple color corresponds to the transport of a correct value, while the red color corresponds to the transport of a wrong value.

5.2. **Stability.** We have implemented the above iterative algorithm in our Rust code. The thread-based parallelism within each subdomain is managed, like before, by the Rust `rayon` library. Communications between the subdomains are managed through calls to the MPI (Message Passing Interface) library. The subdomains are constructed using the METIS graph partitioning tool [35].

First experiments allowed us to verify the stability properties of the transport solver. They indicate that the number of iterations of the iterative algorithm is indeed important for the stability of the method. For a general domain decomposition and with large time steps, the algorithm is stable provided that three iterations are performed at each time step. An illustration is given in Figure 5.3.

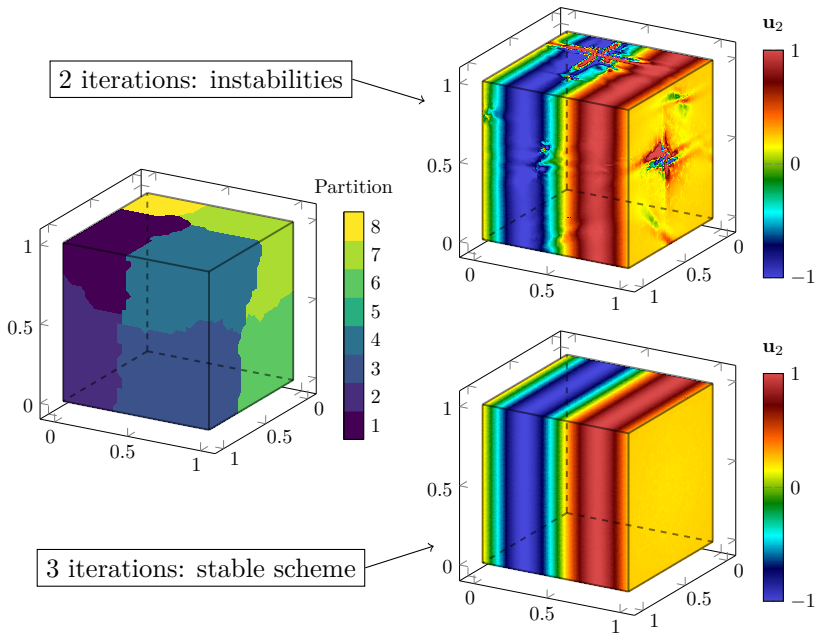

FIGURE 5.3. Stability of the subdomain iterative algorithm applied on a mesh of the unit cube with 8 subdomains. Left: structure of the subdomains; Top right: scheme with two iterations; Bottom right: scheme with three iterations. We observe that the iterative algorithm is stable, even with large time steps, but that three iterations seem to be necessary.

The objective of the subdomain algorithm was to relax the computational dependencies and to achieve a better parallel (strong) scaling of the method. This goal is achieved, as shown in Table 2. In this table, we compare the time spent in the iterative algorithm with a varying number of threads and subdomains. We define the efficiency e of the parallelization as the ratio between the elapsed time of the algorithm and the time that we would get with an ideal perfect strong scaling. The efficiency is perfect if $e = 1$.

TABLE 2. Multithread and MPI strong scaling test on a mesh with about 3.5M elements, on a server equipped with an AMD EPYC 7713 x2 (128 physical cores). For a computation done with 128 threads, we observe that it is better to evenly split the computational work between threads and subdomains (green-tinted rows) rather than prioritizing threads (red-tinted row).

# Subdomains	# Threads	# CPU	Time (s)	Efficiency
1	1	1	11350	1.0
1	2	2	7913.9	0.717
1	4	4	3918.7	0.724
1	8	8	1896.0	0.748
1	16	16	1061.1	0.668
1	32	32	646.12	0.549
1	64	64	424.70	0.418
1	128	128	455.70	0.195
2	64	128	250.74	0.354
4	32	128	186.59	0.475
8	16	128	155.03	0.572
16	8	128	155.08	0.572
32	4	128	161.14	0.550
64	2	128	162.38	0.546
128	1	128	162.66	0.545

We observe, for instance, that with a single subdomain, the efficiency with 128 threads drops to $e = 0.195$ (as displayed in the red-tinted row), while with 8 subdomains and 16 threads per subdomain, or vice versa, the efficiency increases to $e = 0.572$ (as displayed in the green-tinted rows). We have thus validated the efficiency of this approach. Note that the efficiency generally drops with the number of threads, but this is due to the fact that the mesh remains too small to provide enough work for each thread.

Of course, the whole algorithm is impacted by a slowdown imposed by the additional iterations. However, the weak scaling of the method on a supercomputer is now certainly ensured for very large computations. Indeed, explicit subdomain decomposition methods are known to be well adapted to the architecture of supercomputers, see for instance [11, 21].

6. NUMERICAL RESULTS

In this section, we present several numerical results obtained with the kinetic method in three space dimensions. We apply the method to Maxwell's equations, and the numerical setup is described in Section 6.1. Several numerical experiments are performed, namely the propagation of a plane wave in Section 6.2 and the simulation of a conductive wire in Section 6.3. Lastly, a real-world simulation of the interaction of waves emitted by an antenna with the human body is presented in Section 6.4.

6.1. **Setup of the numerical experiments.**

As a first step, we briefly describe the model used in our numerical experiments, Maxwell's equations, in Section 6.1.1. Then, we mention in Section 6.1.2 how the CFL condition is chosen for this 3D problem, before defining the kinetic velocities $(V_k)_{k\in\{0,\dots,3\}}$ in Section 6.1.3. According to Table 2, balancing between number of subdomains and number of threads leads to the best efficiency. Unless otherwise mentioned, we use such a setup for each experiment.

6.1.1. *Maxwell's equations.*

Maxwell's equations are a hyperbolic system of conservation laws, where the vector $W \in \mathbb{R}^6$ of conservative variables is made of the electric field $E \in \mathbb{R}^3$ and the magnetic field $H \in \mathbb{R}^3$, as follows:

$$W = (E^\intercal, H^\intercal)^\intercal.$$

Maxwell's equations read

$$\begin{cases} \partial_t E - \nabla \times H = -\sigma E, \\ \partial_t H + \nabla \times E = 0. \end{cases} \tag{6.1}$$

The flux of Maxwell's equations in direction $N \in \mathbb{R}^3$ is given by

$$Q(W, N) = \begin{pmatrix} -N \times H \\ N \times E \end{pmatrix},$$

and we also consider the following source term, which models a conductive material with conductivity σ:

$$S(W) = \begin{pmatrix} -\sigma E \\ 0 \end{pmatrix}.$$

6.1.2. *CFL condition.* In order to properly compare methods, we have to define the CFL condition. The reader is referred to [27] for a more in-depth discussion on CFL conditions for DG methods. Here, we define the time step Δt as follows:

$$\Delta t = \beta \frac{1}{\lambda_{\max}} h_{\min}. \tag{6.2}$$

In this definition, $\lambda_{\max}$ is the maximum eigenvalue of the Jacobian matrix of the flux associated to Maxwell's equations (here, $\lambda_{\max} = 1$). In addition, $h_{\min}$ is defined as the size of the smallest cell in the mesh:

$$h_{\min} = \min_{L \in \mathcal{M}} \text{size}(L), \quad \text{where} \quad \text{size}(L) = \frac{\text{volume}(L)}{\text{surface}(\partial L)}.$$

Finally, β is the CFL number. The maximum possible value for β is constrained by the scheme under consideration; for classical explicit DG schemes, β must be of the order of 1 to get stability. As we will see in the numerical experiments, we are able to take β as large as we want without loss of stability (but incurring a loss in precision). This hold whatever value of $\omega < 2$ is chosen, so we take $\omega = 2 - 10^{-12}$ in order to ensure stability while remaining second-order accurate.

6.1.3. *Kinetic velocities.* The last ingredient needed to define the scheme is the set of kinetic velocities. The simplest choice is to choose the following velocities, called "D3Q4" in the Lattice-Boltzmann community:

$$V_0 = \begin{pmatrix} \lambda \\ \lambda \\ \lambda \end{pmatrix}, \quad V_1 = \begin{pmatrix} \lambda \\ -\lambda \\ -\lambda \end{pmatrix}, \quad V_2 = \begin{pmatrix} -\lambda \\ \lambda \\ -\lambda \end{pmatrix}, \quad V_3 = \begin{pmatrix} -\lambda \\ -\lambda \\ \lambda \end{pmatrix},$$

with $\lambda = \sqrt{3}$ to satisfy the subcharacteristic condition. Note that, with this velocity set, the equilibrium functions $M_k(W)$ read, for all $k \in \{0, 1, 2, 3\}$:

$$M_k(W) = \frac{W}{4} + \frac{Q(W, V_k)}{4\lambda^2}.$$

6.2. **Plane wave.** In order to validate the subdomain decomposition, we first run an experiment already performed in [27] without the subdomain decomposition. We expect the results to be almost the same. The computational domain is the unit cube. For this test, the conductivity σ is set to zero. We consider two meshes, represented on Figure 6.1, one with a uniform cell size (labeled $\mathcal{M}_1$) and one with a non-uniform cell size (labeled $\mathcal{M}_2$). We compute, with the meshes described above, the propagation of a plane wave with frequency ν. The exact solution therefore is

$$W(X, t) = \begin{pmatrix} 0 \\ 0 \\ \cos(2\pi\nu(x_1 - t)) \\ 0 \\ -\cos(2\pi\nu(x_1 - t)) \\ 0 \end{pmatrix},$$

and we prescribe this exact solution on the boundary with Dirichlet boundary conditions.

We check the CFL-less feature in Table 3. To that end, we define the error $e_r(\nu)$ between the exact and approximate solutions. For a more precise description of this error and of the whole setup, the reader is referred to [27]. In this table, we verify that the scheme is stable, even at very high CFL numbers β, for both meshes. In addition, for fixed Δt (which is not the same as fixed β according to the definition (6.2) of Δt), the scheme is about as precise on both meshes. Note that, for a standard third order explicit DG scheme to be stable in this

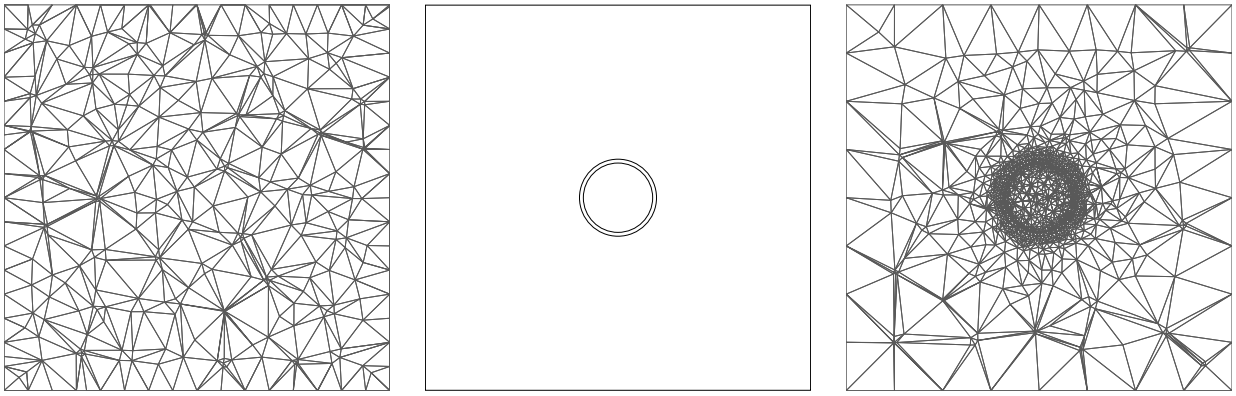

FIGURE 6.1. Two meshes of the unit cube, sliced at $x_2 = 0.5$. The left panel contains a mesh with uniformly spaced tetrahedra, labeled $\mathcal{M}_1$, while the other two focus on a mesh with local refinement at the center, labeled $\mathcal{M}_2$. The center panel contains the geometry of this local refinement (in the shape of a torus), while the right panel depicts the locally refined mesh itself.

configuration, a CFL condition $\beta \leq 1.85$ is required. However, the kinetic scheme remains stable and precise when using larger time steps, for which an explicit DG scheme would not be stable any longer. For instance, on the locally refined mesh $\mathcal{M}_2$, we are able to take a CFL number $\beta = 37 = 20 \times 1.85$ while retaining about the same error as with $\beta = 1.85$. Moreover, the results are the same as the single-subdomain version from [27], which further validates our approach.

TABLE 3. Numerical results for the experiment described in Section 6.2 and pictured in Figure 6.1. For both meshes, we collect the time step Δt and the error e_r with respect to the value of the CFL number and to the choice of frequency.

	mesh $\mathcal{M}_1$			mesh $\mathcal{M}_2$		
CFL β	Δt	$e_r(2)$	$e_r(5)$	Δt	$e_r(2)$	$e_r(5)$
0.37	0.00084	0.00046	0.00627	0.00009	0.00103	0.01467
0.93	0.00211	0.00047	0.00657	0.00023	0.00103	0.01467
1.85	0.00422	0.00062	0.00891	0.00046	0.00103	0.01467
3.70	0.00845	0.00162	0.02397	0.00091	0.00103	0.01468
9.25	0.02112	0.00960	0.14851	0.00228	0.00104	0.01479
18.50	0.04223	0.03990	0.42444	0.00456	0.00115	0.01619
37.00	0.08447	0.14919	0.34411	0.00912	0.00210	0.02992
92.50	0.21117	0.25771	0.67218	0.02281	0.01107	0.16589
185.00	0.42234	0.45671	0.49513	0.04562	0.04509	0.40344

Color plots of the numerical results are given in Figure 6.2. These plots illustrate the stability of the computations at high CFL numbers, and we note that the approximate solution remains stable even for extremely large values of the CFL number β. Of course, the accuracy of the computation depends on the frequency of the plane wave: the more the solution oscillates, the smaller the time step should be in order to accurately capture the oscillations.

6.3. **Conductive wire.** In this test, we activate the source term, i.e., we take a nonzero σ (at least in some part of the domain). We consider a small electric wire located in the middle of the computational domain. The unstructured mesh of the unit cube conforms with the small wire, which means large cells far from the wire and locally refined cells close to the wire (see Figure 6.3). Once again, this experiment was also performed in [27], and we present it here to further validate the transport algorithm on the subdomain decomposition.

In order to validate the proposed methodology, we compare our DG method with a well-validated FDTD (Finite-Difference Time-Domain) solver [30], based on the Yee scheme [45], that can handle electric wires. This FDTD solver requires a uniform Cartesian grid, which means it is easy to parallelize. Therefore, the fine mesh within the antenna implies a uniformly fine mesh everywhere in the domain; in practice, we use $1000^3 = 10^9$ cells.

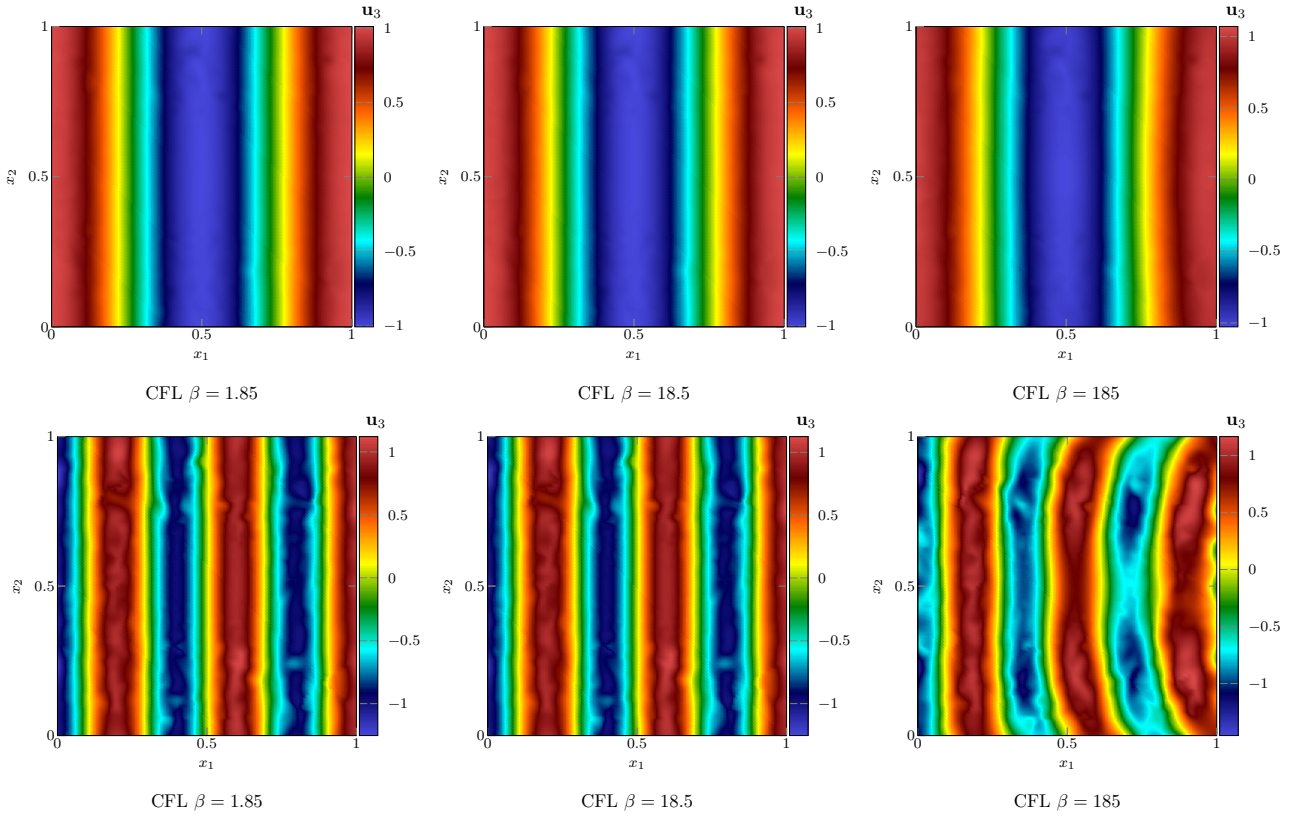

FIGURE 6.2. Plane wave propagation from Section 6.2 on the locally refined mesh $\mathcal{M}_2$: depiction, for $x_3 = 0.5$, of the third component $W_3(X,t) = E_3(X,t)$ of the approximate solution. We simulated plane waves with frequencies $\nu = 2$ (top panels) and $\nu = 5$ (bottom panels). From left to right, we have set the CFL number β to 1.85, 18.5 and 185.

We also compare our results to an explicit RK2-DG solver, called CLAC (Computation Laws on mAny Cores). This solver is contained within a well-validated code, parallelized on a GPU, as opposed to our implicit kinetic solver.

The differences between the solvers are summarized in Table 4.

TABLE 4. Summary of the algorithms used in this section: algorithm type, programming language and parallelization type.

Solver	Algorithm	Language	Parallelization
FDTD	finite differences	`Fortran`	CPU, distributed
CLAC	explicit RK2-DG	`C++`, `OpenCL`	GPU
KOUGLOFV	implicit kinetic DG	`Rust`	CPU, shared & distributed

To set up the numerical experiment, a plane wave pulse is sent through the vacuum, towards the wire. This amounts to solving Maxwell's equations with the conductivity source term $S(W) = (\sigma E, 0)^\intercal$, where the conductivity σ vanishes outside the wire. To define the initial and boundary conditions, we consider the following exact solution of Maxwell's equations without source term:

$$W(X,t) = \begin{pmatrix} 0 \\ 0 \\ \psi(x_2 - x_c - t) \\ -\psi(x_2 - x_c - t) \\ 0 \\ 0 \end{pmatrix},$$

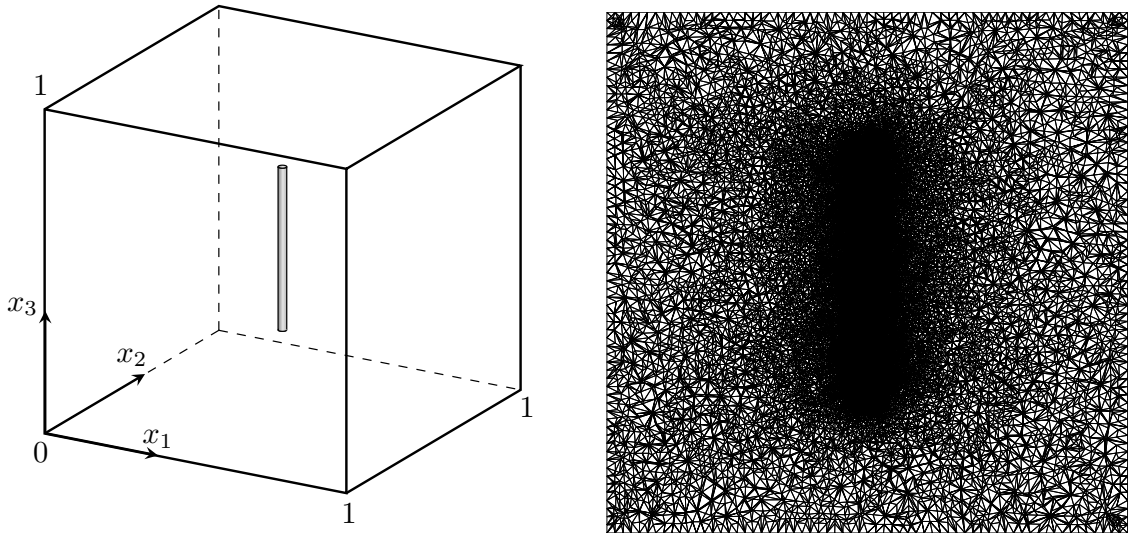

FIGURE 6.3. Geometry (left panel) and mesh (right panel) for the conductive wire test from Section 6.3. The mesh is locally refined around the wire, where the conductivity σ is nonzero.

where $x_c = 0.25$ and where ψ is a compactly supported bump function:

$$\psi(X) = \begin{cases} \exp\left(1 - \dfrac{1}{1 - \frac{\|X\|}{\eta}}\right) & \text{if } \|X\| < \eta, \\ 0 & \text{otherwise,} \end{cases}$$

with $\eta = 0.25$ the size of the bump. Then, the initial condition is $W(X, 0)$, the boundary conditions consist in imposing the solution $W(X, t)$ at the boundaries. Note that $W(X, t)$ is not an exact solution of the problem with source term; however, the antenna is far enough away from the boundaries for this fact not to matter when running simulations.

Note that, in this case, the application of the source term in the scheme, according to (2.7), reads:

$$\begin{cases} \dfrac{E(\cdot, \Delta t^+) - E(\cdot, \Delta t^-)}{\Delta t} = \dfrac{\sigma E(\cdot, \Delta t^+) + \sigma E(\cdot, \Delta t^-)}{2}, \\ \dfrac{H(\cdot, \Delta t^+) - H(\cdot, \Delta t^-)}{\Delta t} = 0. \end{cases}$$

This is a linear equation in the unknown $E(\cdot, \Delta t^+)$, which gives:

$$\begin{cases} E(\cdot, \Delta t^+) = \mu E(\cdot, \Delta t^-), \\ H(\cdot, \Delta t^+) = H(\cdot, \Delta t^-), \end{cases} \quad \text{where} \quad \mu = \frac{1 - \sigma \dfrac{\Delta t}{2}}{1 + \sigma \dfrac{\Delta t}{2}}. \tag{6.3}$$

We now apply the solvers to two different cases. Within the wire, we consider two values of the conductivity σ: first, a small conductivity $\sigma = 3$ in Section 6.3.1; then, an infinite conductivity $\sigma \to +\infty$ in Section 6.3.2. For the KOUGLOFV solver, we take a CFL number $\beta = 7$. Recall that the scheme is stable whatever the value of β, but this choice yields a good compromise between precision and computation speed. Since the three solvers give comparable results, a computation time comparison is proposed in Section 6.3.3.

6.3.1. *Wire with a low conductivity.* We first consider $\sigma = 3$. We display on Figure 6.4 the approximate solution obtained by the KOUGLOFV solver. We observe a good agreement with the expected results, since the electric charge is concentrated at the ends of the antenna, and the magnetic field rotates around the antenna.

Then, to compare the KOUGLOFV solver with the FDTD and CLAC solvers, the first and second components of the magnetic field H_x and H_y, along the line $y = z = 1/2$ and when the pulse reaches the wire, are represented on Figure 6.5. Note that, for CLAC, the results are displayed on the domain $(0.1, 0.9)$ instead of $(0, 1)$. This is due to a technical limitation of the CLAC code related to the boundary conditions, which is not present for the FDTD and KOUGLOFV solvers.

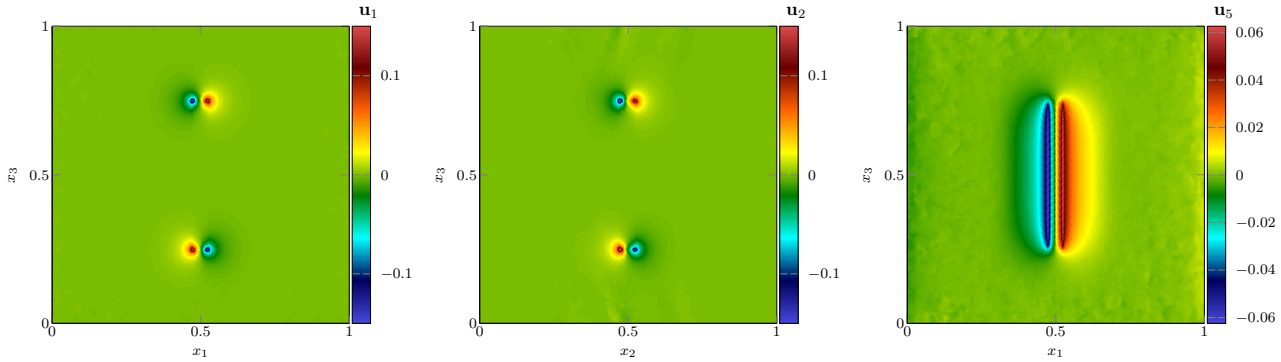

FIGURE 6.4. Conductive wire simulation from Section 6.3.1, with $\sigma = 3$, solution at $t = 0.75$: left panel: $E_1\big|_{x_2=0.5}$; center panel: $E_2\big|_{x_1=0.5}$; right panel: $H_2\big|_{x_2=0.5}$.

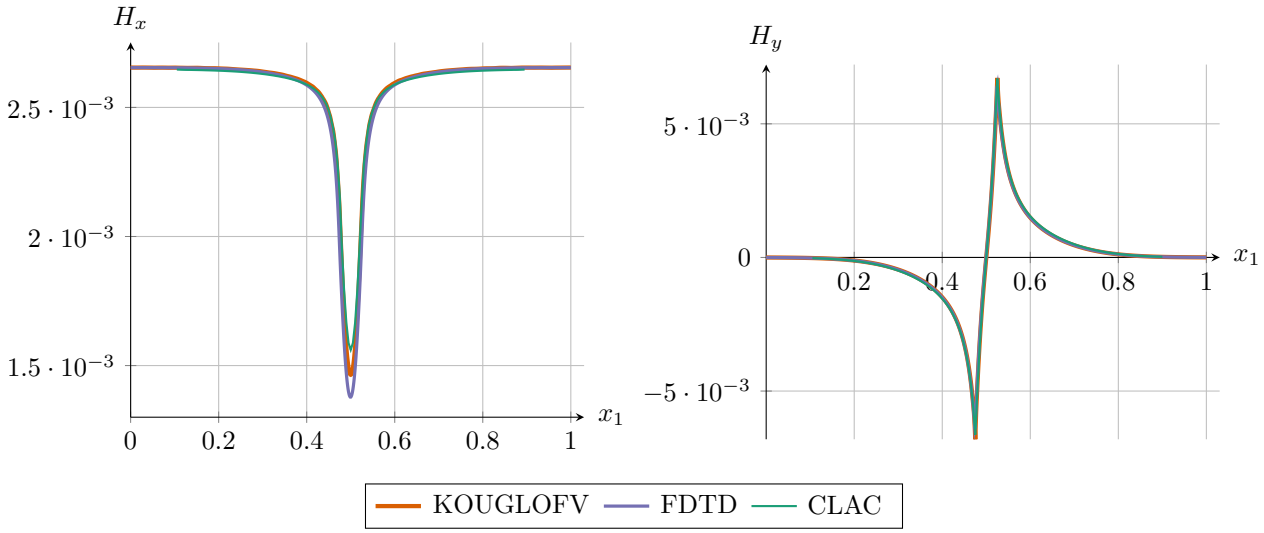

FIGURE 6.5. Conducting wire with small conductivity $\sigma = 3$, comparison between the implicit DG solver (KOUGLOFV), the FDTD solver and a standard explicit DG solver (CLAC).

We observe an excellent agreement with the FDTD and CLAC solvers. Let us mention that, to obtain these results, our implicit kinetic DG solver does not apply any charge conservation correction, while this is generally considered to be necessary for such simulations, see for instance [39, 16]. Further investigation is needed for understanding this good behavior, which is perhaps linked to the test case under consideration.

6.3.2. *Infinitely conductive wire.* In this section, we consider a test with a large conductivity $\sigma \to +\infty$. In this case, the source term is very stiff, but is handled without issue by our method, thanks to the implicit source term treatment. Indeed, recall equation (6.3): when $\sigma \to +\infty$, we get

$$\begin{cases} E(\cdot, \Delta t^+) = -E(\cdot, \Delta t^-), \\ H(\cdot, \Delta t^+) = H(\cdot, \Delta t^-). \end{cases}$$

This behavior is consistent with what would happen in perfect electrical conductors (PECs). In practice, we take $\sigma = 10^{12}$ to mimic infinity. This gives the same results as writing the source term in the $\sigma = +\infty$ limit. The numerical results are depicted in Figure 6.6. We observe a good agreement with the expected values of the electric and magnetic fields. Indeed, the electric field (left and center panels) remains constant, equal to zero, within the wire. In addition, the right panel clearly shows that the wave (traveling from left to right) has just passed the wire, but has not created a magnetic field within the wire.

Contrary to this, in the explicit methods, a vanishingly small CFL condition (proportional to $\frac{1}{\sigma}$) is needed for stability. This is, of course, unusable in practice, and a specific PEC formulation has to be used. This is

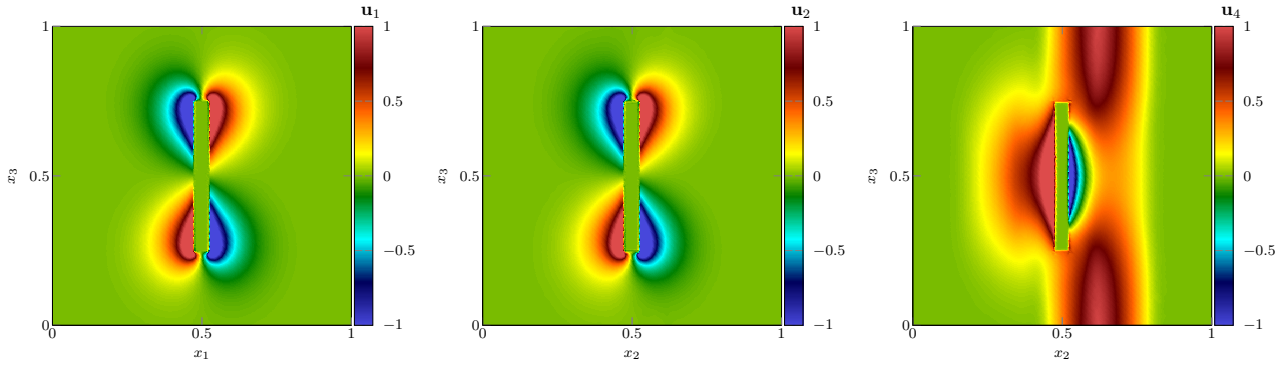

FIGURE 6.6. Infinitely conductive wire simulation from Section 6.3.2, with $\sigma \to +\infty$, solution at $t = 0.375$: left panel: $E_1\big|_{x_2=0.5}$; center panel: $E_2\big|_{x_1=0.5}$; right panel: $H_1\big|_{x_1=0.5}$.

another advantage of our method: there is no need to implement complex PEC conditions to handle infinite conductivities. We compare the approaches on Figure 6.7, where we observe very good agreement between the solutions. The implicit DG solver presents a few small oscillations close to the wire. They can mostly be attributed to visualization, since the software used to create the slices (Paraview, see [4]) performs an interpolation. A large part of the oscillation amplitude is due to these visualization artifacts, and a small part is due to standard dispersion effects (since we consider a high-order scheme with a large CFL number).

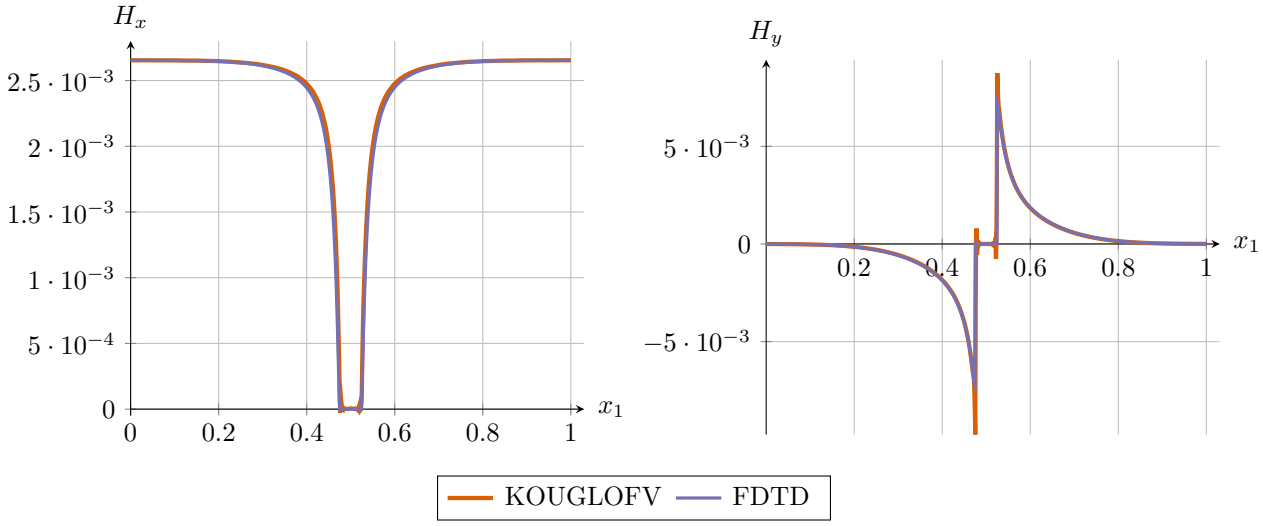

FIGURE 6.7. Conducting wire with infinite conductivity $\sigma \to +\infty$, comparison between the implicit DG solver (KOUGLOFV), and the FDTD solver.

6.3.3. *Computation time comparison.* Finally, in light of the similar results obtained in both the low and the infinite conductivity cases, we report, in Table 5, the computation time taken by each solver. For the FDTD solver, the sheer size of the mesh makes obtaining the approximate solution quite a bit slower than with the KOUGLOFV solver. The CLAC solver is much faster than the FDTD solver, but because of the restrictive CFL condition, obtaining the solution is still faster with the KOUGLOFV solver. Note that the CLAC solver is parallelized on a GPU, while the KOUGLOFV solver only involves a multi-core CPU. Even so, the KOUGLOFV solver manages to be significantly faster than the CLAC solver, and it takes a comparable computation time when ran on fairly ancient desktop CPUs.

6.4. **Real-world simulation: interaction of waves from an antenna with an anthropomorphic mannequin.** In this configuration, we compare our DG method to the CLAC solver on a large mesh composed of over 6 million tetrahedrons (precisely 6 533 341). The purpose of this test is to validate the scaling of the solver, as well as display its results on a realistic test case. The treated mesh represents an anthropomorphic

TABLE 5. CPU time taken to solve the conductive wire test case: comparison between the FDTD method, the explicit RK2-DG solver (CLAC), and the kinetic DG solver (KOUGLOFV). For KOUGLOFV, we take a CFL number $\beta = 7$.

solver	number of cells	hardware platform	computation time
FDTD	1000^3	AMD EPYC 7302×2, 32 cores, 3 GHz	15.3 hours
CLAC	236k	Nvidia GeForce GTX 1070	5.98 minutes
KOUGLOFV	516k	Intel i7-5820K, 6 cores, 3.3 GHz	17.1 minutes
		AMD EPYC 7713×2, 128 cores, 2 GHz	78.7 seconds

mannequin named Kyoto, and it has been used in other works. So far, it has not been included in peer-reviewed articles, but it was used in several PhD theses [43, 34] and in a PRACE SHAPE project in collaboration with the AxesSim company, see the white paper [29] and the report [7]. In these contexts, the tetrahedra were cut into four hexahedra each, which increased the total number of elements fourfold.

The body model is composed of 12 organs (including the skeleton and the skin). To handle these different body parts, and to include a source term modeling the behavior of a current generated by an antenna, we modify Maxwell's equations (6.1), as follows:

$$\begin{cases} \partial_t(\varepsilon E) - \nabla \times H = -\sigma E - J, \\ \partial_t H + \nabla \times E = 0. \end{cases} \tag{6.4}$$

In (6.4), J is the time- and space-dependent electric current density and ε is the permittivity of the material, which obviously depends on the material (and therefore on the space variable X). We introduce the vacuum permittivity ε_0, to write $\varepsilon = \varepsilon_r \varepsilon_0$ with ε_r the relative permittivity of the material. For each body part, the values of ε_r and σ are listed in Table 6. In addition, we set $\widetilde{E} = \varepsilon_r E$. Reformulating (6.4), we obtain, assuming that each material has the same permeability:

$$\begin{cases} \partial_t \widetilde{E} - \nabla \times H = -\sigma \dfrac{\widetilde{E}}{\varepsilon_r} - \dfrac{1}{\varepsilon_r} J, \\ \partial_t H + \nabla \times \dfrac{\widetilde{E}}{\varepsilon_r} = 0. \end{cases} \tag{6.5}$$

Note that the flux in (6.5) is discontinuous as soon as ε_r is discontinuous, which is the case here since different materials have different relative permittivities. In practice, we solve for $\widetilde{E}$ and H.

TABLE 6. Electromagnetic constants for each material in the anthropomorphic mannequin simulation.

Material	ε_r	σ (Sv m^{-1})
brain	48.34	2.02
heart	58.67	3.02
lung	22	0.36
liver	41.82	1.9
gallbladder	60	2
spleen	56.75	2.46
pancreas	56.75	2.46
kidney	56.83	2.62
colon	48.5	0.93
bladder	20	0.7
muscle	50	1.33
bone	11.41	0.43
cartilage	36	1.6

A volume-meshed dipole antenna has been placed next to the left arm of the body model, and a volumetric source term has been imposed along that dipole. This leads to the electric current density J in (6.5) being nonzero only within the antenna. In dimensional quantities, the cells in the antenna have average size 200 µm,

while the cells in vacuum have average size 2 cm. This means that the ratio of largest cell size over smallest cell size is around 100; hence, this mesh provides a good framework to test our CFL-less methodology. In addition, to mimic a Bluetooth antenna, this dipole antenna emits a modulated Gaussian pulse, with dimensional frequency 2.4 GHz, which lasts for about 1.5 ns. After that time has elapsed, the source term J vanishes in the whole domain.

TABLE 7. Differences between CLAC and KOUGLOFV.

	CLAC	KOUGLOFV
space order	3 (10-point tetrahedra)	3 (10-point tetrahedra)
time order	3 (explicit, RK3)	2 (implicit, Crank-Nicolson)
floating-point precision	simple	double
boundary conditions	Silver-Müller	homogeneous Dirichlet
time step	$\Delta t = 9.04 \times 10^{-6}$ ns	$\Delta t = 3.33 \times 10^{-4}$ ns

In Table 7, we sum up the differences between the two solvers. Since the two codes have different time stepping strategies, different boundary conditions, etc., we do not expect their behaviors to be quantitatively comparable. However, they should, qualitatively, lead to similar results. These differences between the two solvers also explain why the KOUGLOFV code takes about as much computation time on a few hundred CPUs than the CLAC code on 6 GPUs. The computation time figures are reported in Table 8, and we observe that the KOUGLOFV code takes about as much time to run as the CLAC solver despite being in double-precision arithmetic and running on 128 CPU cores rather than 6 GPUs.

Indeed, this good behavior happens thanks to the CFL-less implicit time-stepping in the scheme underlying in the KOUGLOFV solver. To correctly implement this implicit scheme, we have to give a value to the over-relaxation parameter $\omega \in [1, 2]$ from (2.5), see the discussion at the end of Section 2.2. Recall that taking $\omega = 2$ leads to a second-order scheme, while other choices lead to a first-order scheme, with larger values of ω corresponding to higher resolutions. Here, we made the choice to take $\omega = 1.8$, since larger values led to small instabilities stemming from the discontinuous flux function in (6.5): indeed, lowering the order of the scheme helped curb these spurious oscillations. This is reminiscent of ideas from, for instance, [32, 38]. In addition, another choice to make is the value of the time step, since the scheme is unconditionally stable. We chose to take $\Delta t = 1 \times 10^{-4}$ in non-dimensional form, leading to a time step about 40 times larger than the stability limit of the scheme from the CLAC code. This made it possible to have good results from a relatively short simulation.

We first present the modulus of E, in dB(V/m), in the (x_1, x_3) plane. They are displayed on Figure 6.8 for $t = 0.5$ ns, on Figure 6.9 for $t = 0.9$ ns, and on Figure 6.10 for $t = 1.8$ ns. In each case, we observe good agreement between the results of CLAC and KOUGLOFV, despite the differences in the two approaches.

To get more precise results, we compare in Figures 6.11 to 6.13 the two numerical results at four points:

(1) close to the antenna (in vacuum), top left panels;
(2) in the liver, top right panels;
(3) in the brain, bottom left panels;
(4) on the left side (in vacuum), bottom right panels.

At each point, the signals have the same shape for the two codes, although the signal from KOUGLOFV is more diffused compared to the one from CLAC. Close to the antenna, the signal produced by KOUGLOFV has about 15% relative error with respect to the CLAC signal, which can be attributed to the difference in source

TABLE 8. Computation time for both KOUGLOFV and CLAC codes, for the anthropomorphic mannequin simulation.

solver	hardware platform	computation time
CLAC	6 × Nvidia GeForce GTX 1080 Ti single precision arithmetic	31 h
KOUGLOFV	AMD EPYC 7713x2, 128 cores, 2 GHz double precision arithmetic	20 h

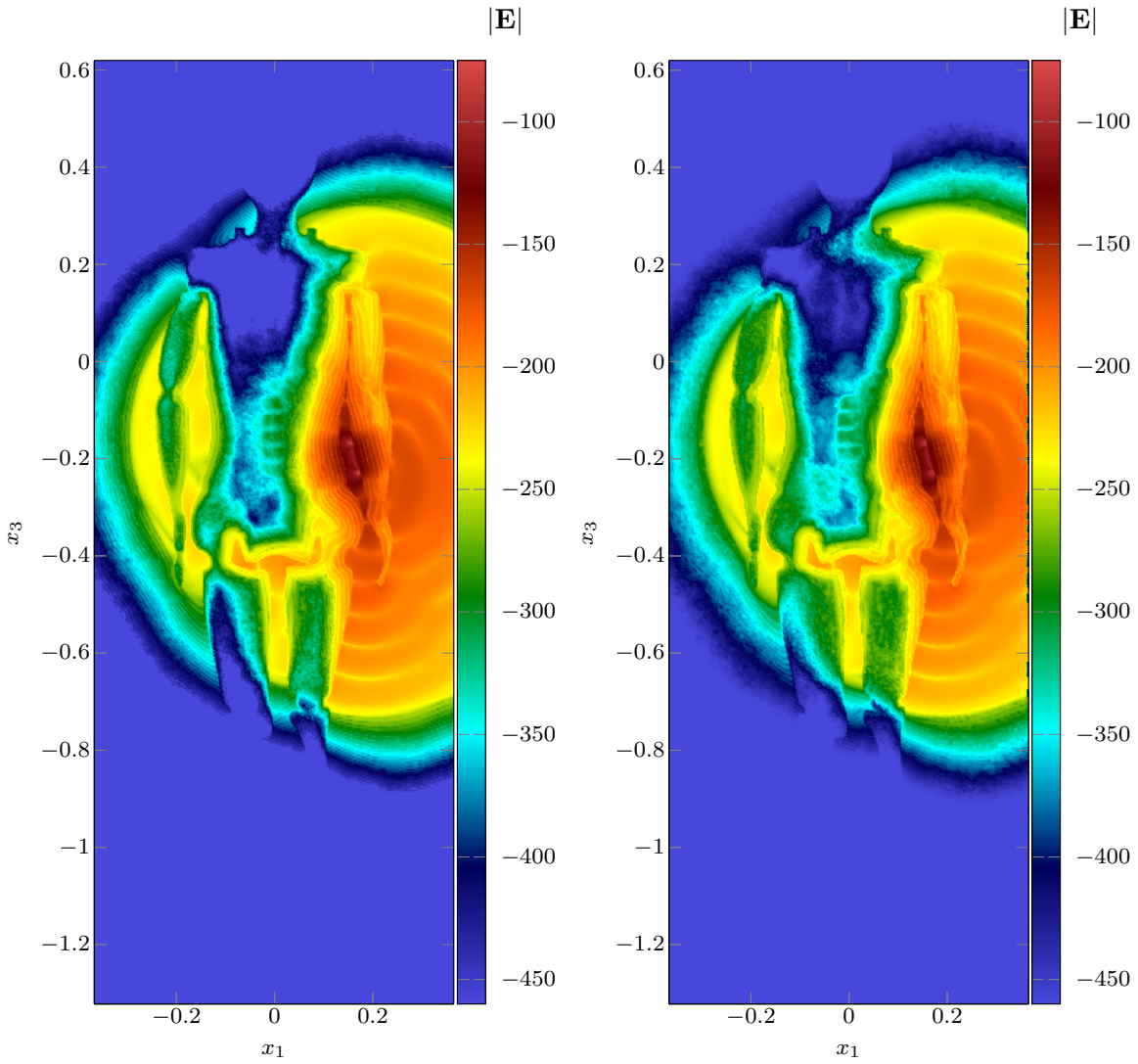

FIGURE 6.8. Numerical approximation of $|E|$, in dB(V/m), sliced in the (x_1, x_3) plane, at $t = 0.5\,\text{ns}$. Left panel: results from CLAC; right panel: results from KOUGLOFV.

term discretizations between the two implementations. That relative error can be used as a baseline to compare the two results. Moving further away from the source, for instance in the brain, the two signals have the same shape but the signal from KOUGLOFV is more diffused than the one from CLAC: this can be attributed to the fact that lowering the order of the scheme was necessary to handle the discontinuous flux in this simulation.

7. Conclusion

We presented an adaptation of the kinetic DG method introduced in [27]. The method can handle arbitrary conservation laws and complex unstructured meshes. It has the complexity of a time-explicit scheme but is CFL-free.

The method presents good parallelization features, for both shared memory and distributed memory computers. To improve the parallel scaling on distributed memory computers, we have proposed a subdomain decomposition method that relaxes the task dependencies of the kinetic scheme but keeps the possibility to use large time steps. The method has been tested and validated on realistic electromagnetic simulations.

In our future works, we plan to apply the method to other conservation laws arising for instance in the modeling of multiphase compressible flows. Investigations are also needed for a more rigorous treatment of the boundary conditions.

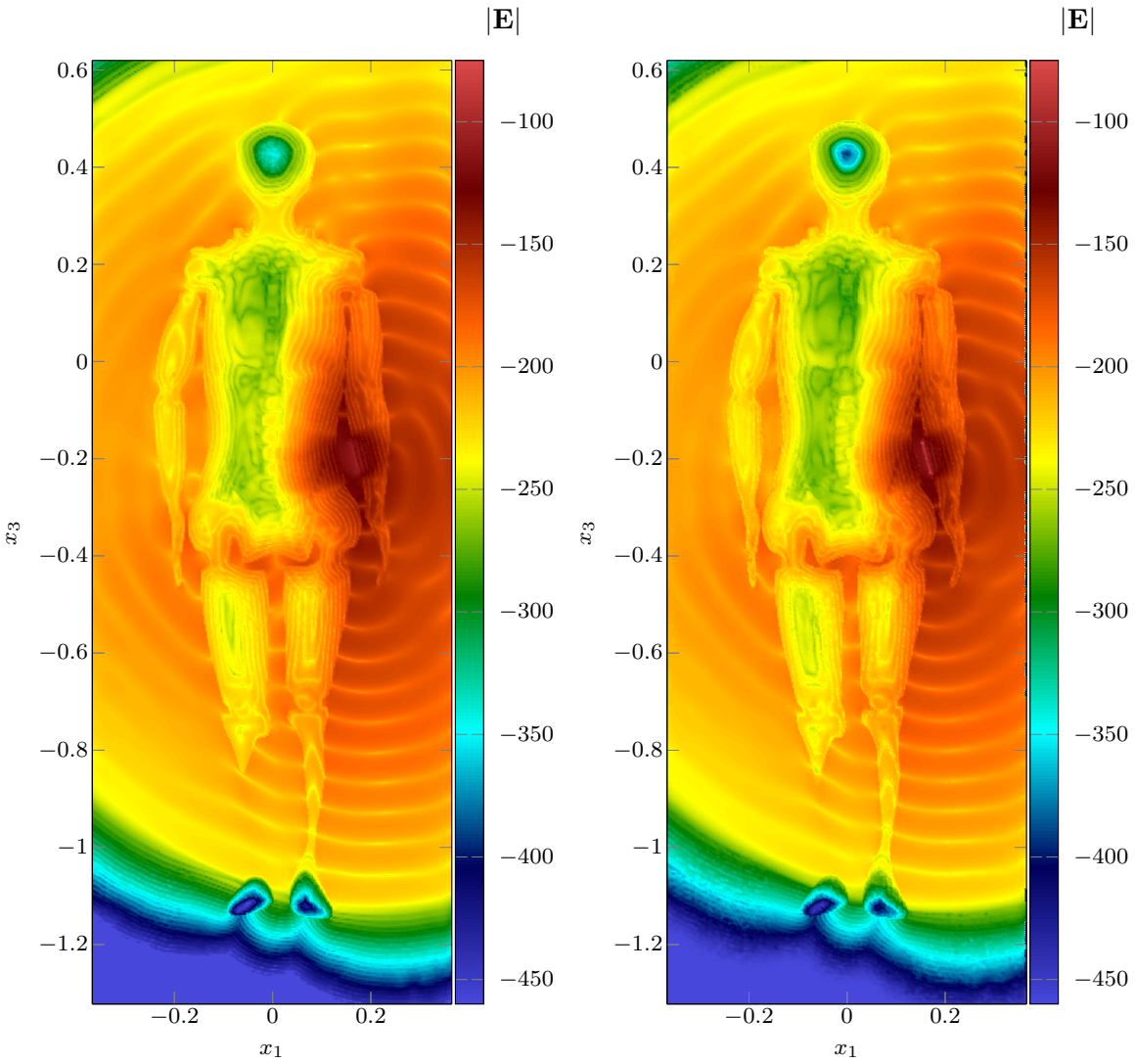

FIGURE 6.9. Numerical approximation of $|E|$, in dB(V/m), sliced in the (x_1, x_3) plane, at $t = 0.9\,\text{ns}$. Left panel: results from CLAC; right panel: results from KOUGLOFV.

REFERENCES

[1] R. Alexander. Diagonally Implicit Runge–Kutta Methods for Stiff O.D.E.'s. *SIAM J. Numer. Anal.*, 14(6):1006–1021, 1977.

[2] C. Altmann, T. Belat, M. Gutnic, Ph. Helluy, H. Mathis, É. Sonnendrücker, W. Angulo, and J.-M. Hérard. A local time-stepping Discontinuous Galerkin algorithm for the MHD system. *ESAIM Proc.*, 28:33–54, 2009.

[3] D. Aregba-Driollet and R. Natalini. Discrete Kinetic Schemes for Multidimensional Systems of Conservation Laws. *SIAM J. Numer. Anal.*, 37(6):1973–2004, 2000.

[4] U. Ayachit. *The ParaView guide : updated for ParaView version 4.3*. Kitware, Clifton Park, New York, 2015.

[5] J. Badwaik, M. Boileau, D. Coulette, E. Franck, Ph. Helluy, C. Klingenberg, L. Mendoza, and H. Oberlin. Task-Based Parallelization of an Implicit Kinetic Scheme. *ESAIM: Proceedings and Surveys*, 63:60–77, 2018.

[6] H. Baty, F. Drui, Ph. Helluy, E. Franck, C. Klingenberg, and L. Thanhäuser. A robust and efficient solver based on kinetic schemes for Magnetohydrodynamics (MHD) equations. *Applied Mathematics and Computation*, 440:127667, 2023.

[7] M. Boileau, C. Girard, Ph. Helluy, M. Houillon, N. Muot, G. Prin, T. Strub, and B. Weber. Simulation de l'interaction électromagnétique des objets connectés avec le corps humain . https://www.genci.fr/sites/default/files/grands-challenges-idris-2020_0.pdf, 2020.

[8] F. Bouchut. Construction of BGK Models with a Family of Kinetic Entropies for a Given System of Conservation Laws. *J. Stat. Phys.*, 95(1/2):113–170, 1999.

[9] F. Bourdel, P.-A. Mazet, and Ph. Helluy. Resolution of the non-stationary or harmonic Maxwell equations by a discontinuous finite element method. Application to an EMI (electromagnetic impulse) case. In *10th international conference on computing methods in applied sciences and engineering on Computing methods in applied sciences and engineering*, pages 405–422. Nova Science Publishers, Inc. Commack, NY, USA, 1992.

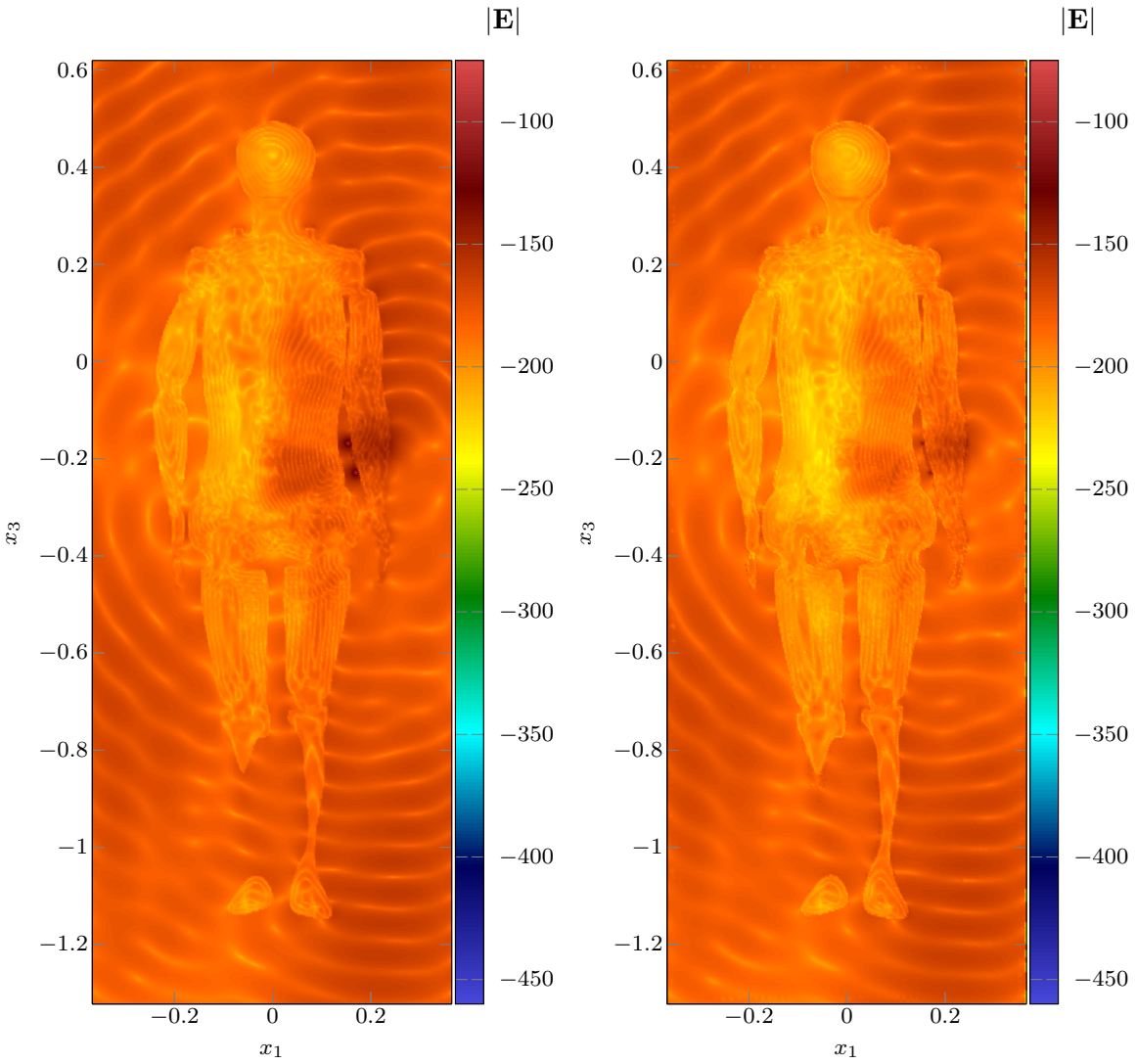

FIGURE 6.10. Numerical approximation of $|E|$, in dB(V/m), sliced in the (x_1, x_3) plane, at $t = 1.8\,\text{ns}$. Left panel: results from CLAC; right panel: results from KOUGLOFV.

[10] Y. Brenier. Averaged Multivalued Solutions for Scalar Conservation Laws. *SIAM J. Numer. Anal.*, 21(6):1013–1037, 1984.
[11] A. Breuer, A. Heinecke, S. Rettenberger, M. Bader, A.-A. Gabriel, and C. Pelties. Sustained Petascale Performance of Seismic Simulations with SeisSol on SuperMUC. In *Lecture Notes in Computer Science*, pages 1–18. Springer International Publishing, 2014.
[12] A. Catella, V. Dolean, and S. Lanteri. An implicit discontinuous Galerkin time-domain method for two-dimensional electromagnetic wave propagation. *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, 29(3):602–625, 2010.
[13] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, editors. *Discontinuous Galerkin methods*, volume 11 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2000. Theory, computation and applications, Papers from the 1st International Symposium held in Newport, RI, May 24–26, 1999.
[14] D. Coulette, E. Franck, Ph. Helluy, M. Mehrenberger, and L. Navoret. Palindromic Discontinuous Galerkin Method. In *Springer Proceedings in Mathematics & Statistics*, pages 171–178. Springer International Publishing, 2017.
[15] D. Coulette, E. Franck, Ph. Helluy, M. Mehrenberger, and L. Navoret. High-order implicit palindromic discontinuous Galerkin method for kinetic-relaxation approximation. *Comput. & Fluids*, 190:485–502, 2019.
[16] A. Crestetto and Ph. Helluy. Resolution of the Vlasov-Maxwell system by PIC discontinuous Galerkin method on GPU with OpenCL. *ESAIM Proc.*, 38:257–274, 2012.
[17] J. Diaz and M. J. Grote. Energy conserving explicit local time stepping for second-order wave equations. *SIAM J. Sci. Comput.*, 31(3):1985–2014, 2009.
[18] V. Dolean, H. Fahs, L. Fezoui, and S. Lanteri. Locally implicit discontinuous Galerkin method for time domain electromagnetics. *J. Comput. Phys.*, 229(2):512–526, 2010.

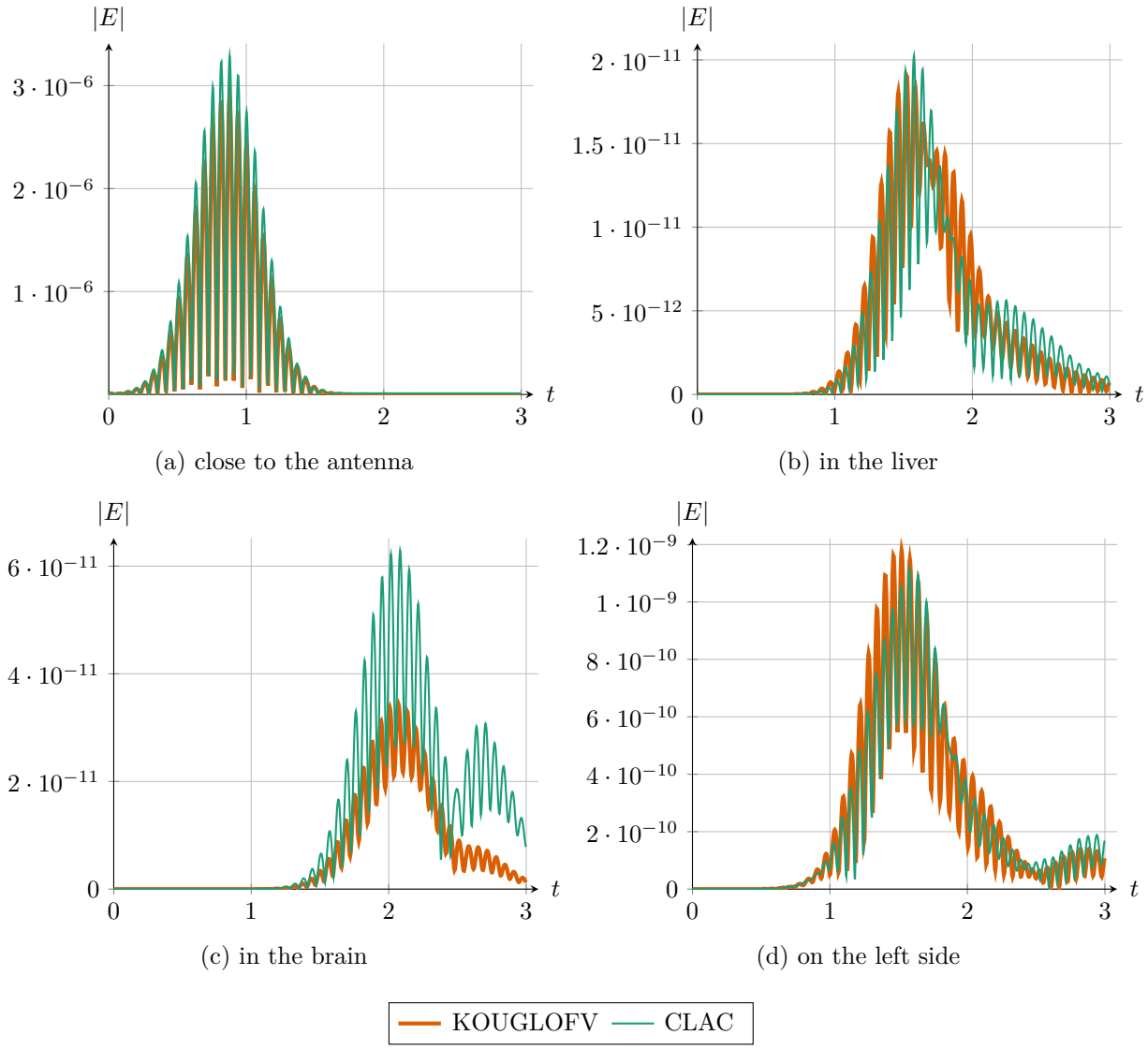

FIGURE 6.11. Numerical approximation of $|E|$ ($\mathrm{V\,m^{-1}}$) at four points within the mesh: comparison between the solutions given by the two codes.

[19] F. Drui, E. Franck, Ph. Helluy, and L. Navoret. An analysis of over-relaxation in a kinetic approximation of systems of conservation laws. *CR Mécanique*, 347(3):259–269, 2019.

[20] F. Dubois. Simulation of strong nonlinear waves with vectorial lattice Boltzmann schemes. *Int. J. Modern Phys. C*, 25(12):1441014, 2014.

[21] M. Dumbser, F. Fambri, M. Tavelli, M. Bader, and T. Weinzierl. Efficient Implementation of ADER Discontinuous Galerkin Schemes for a Scalable Hyperbolic PDE Engine. *Axioms*, 7(3):63, 2018.

[22] M. Dumbser, M. Käser, and E. F. Toro. An arbitrary high-order Discontinuous Galerkin method for elastic waves on unstructured meshes - V. Local time stepping and p-adaptivity. *Geophys. J. Int.*, 171(2):695–717, 2007.

[23] A. Ecer, N. Gopalaswamy, H. U. Akay, and Y. P. Chien. Digital filtering techniques for parallel computation of explicit schemes. *Int. J. Comput. Fluid Dyn.*, 13(3):211–222, 2000.

[24] L. Fezoui, S. Lanteri, S. Lohrengel, and S. Piperno. Convergence and stability of a discontinuous Galerkin time-domain method for the 3D heterogeneous Maxwell equations on unstructured meshes. *ESAIM Math. Model. Numer. Anal.*, 39(6):1149–1176, 2005.

[25] Md. Gaffar and D. Jiao. An Explicit and Unconditionally Stable FDTD Method for Electromagnetic Analysis. *IEEE Trans. Microw. Theory Techn.*, 62(11):2538–2550, 2014.

[26] Md. Gaffar and D. Jiao. Alternative Method for Making Explicit FDTD Unconditionally Stable. *IEEE Trans. Microw. Theory Techn.*, 63(12):4215–4224, 2015.

[27] P. Gerhard, Ph. Helluy, and V. Michel-Dansac. Unconditionally stable and parallel Discontinuous Galerkin solver. *Comput. Math. Appl.*, 112:116–137, 2022.

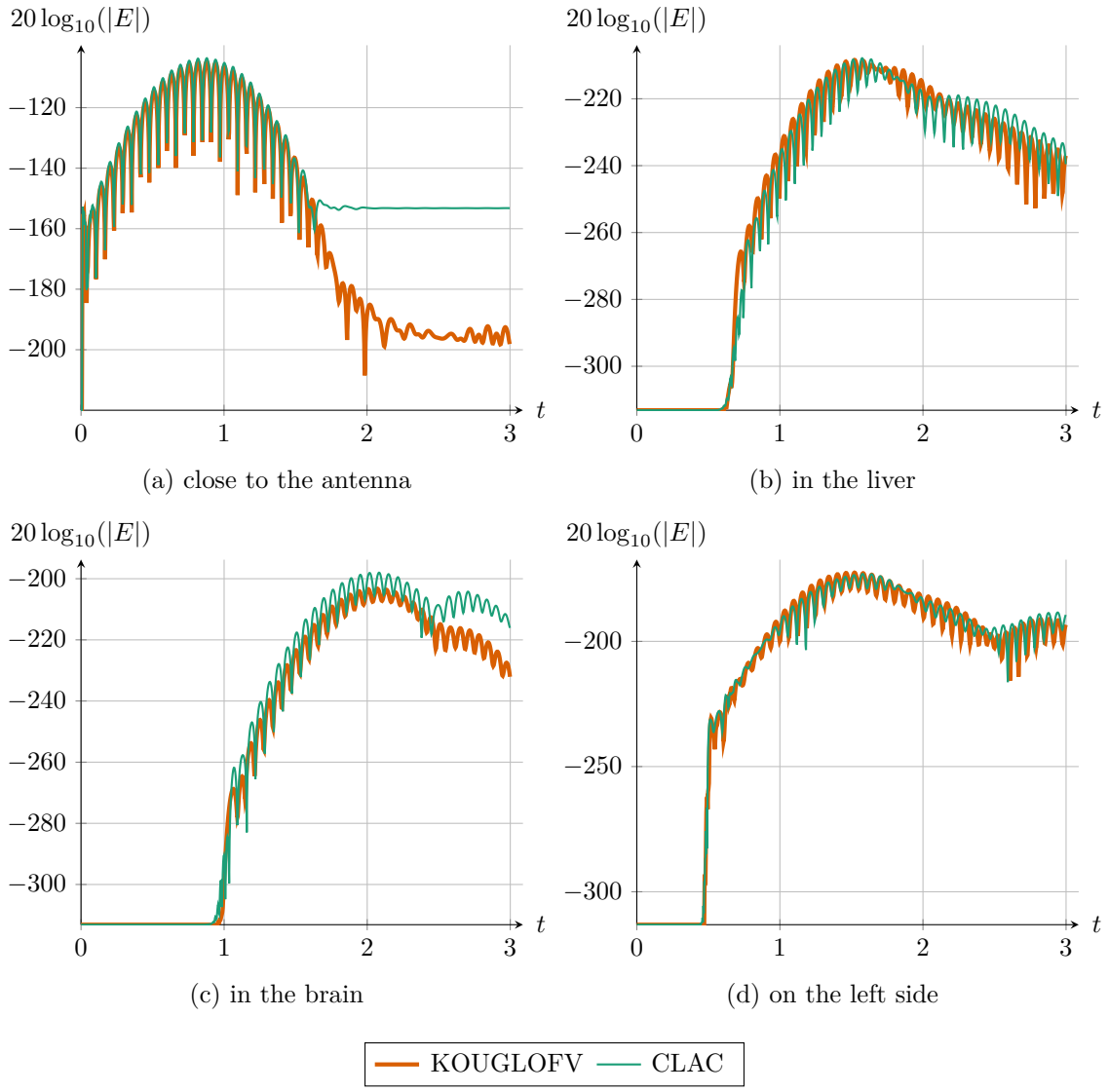

FIGURE 6.12. Numerical approximation of $|E|$ at four points within the mesh: comparison between the solutions given by the two codes. Here, the results are presented in dB(V/m), corresponding to taking 20 times the base-10 logarithm of $|E|$.

[28] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Internat. J. Numer. Methods Engrg.*, 79(11):1309–1331, 2009.

[29] C. Girard, B. Weber, B. Cirou, and V. Cameo Ponz. SHAPE Project AxesSim - CINES Partnership: HPC for connected Objects. https://prace-ri.eu/wp-content/uploads/AXESSIM-%E2%80%93-CINES-Partnership-HPC-for-connected-Objects.pdf, 2018.

[30] C. Guiffaut, A. Reineix, and B. Pecqueux. New Oblique Thin Wire Formalism in the FDTD Method With Multiwire Junctions. *IEEE T. Antenn. Propag.*, 60(3):1458–1466, 2012.

[31] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods*. Springer New York, 2008.

[32] I. Higueras, N. Happenhofer, O. Koch, and F. Kupka. Optimized strong stability preserving IMEX Runge–Kutta methods. *J. Comput. Appl. Math.*, 272:116–140, 2014.

[33] M. Hochbruck and T. Pažur. Implicit Runge–Kutta Methods and Discontinuous Galerkin Discretizations for Linear Maxwell's Equations. *SIAM J. Numer. Anal.*, 53(1):485–507, 2015.

[34] M. Houillon. *Schémas Galerkin Discontinu optimisés pour les problèmes d'électromagnétisme avec des géométries complexes*. Ph.D. Thesis, Université de Strasbourg, 2020.

[35] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[36] C. A. Kennedy and M. H. Carpenter. Diagonally implicit Runge–Kutta methods for stiff ODEs. *Appl. Numer. Math.*, 146:221–244, 2019.

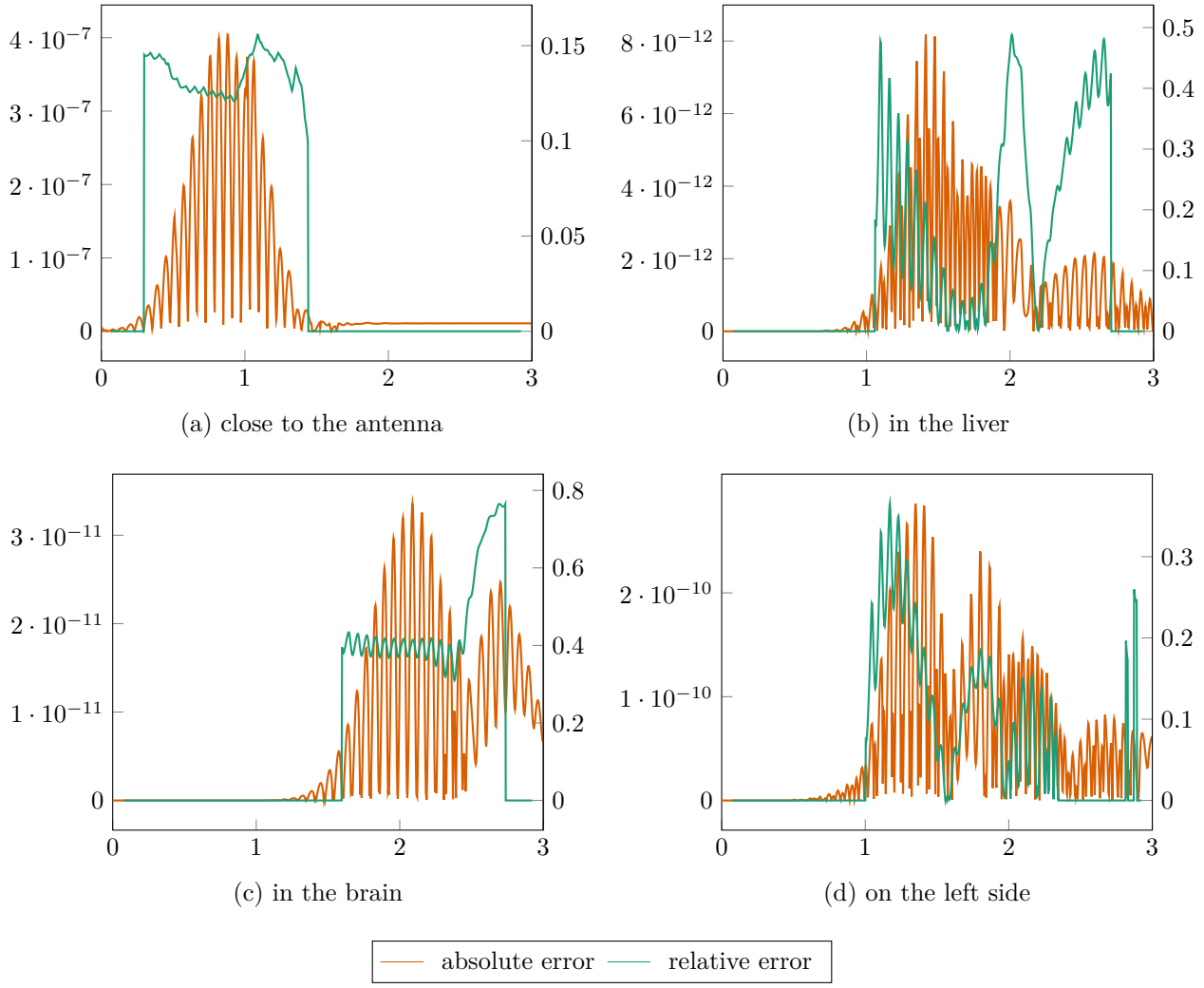

FIGURE 6.13. Absolute (left axes) and relative (right axes) errors between the KOUGLOFV and CLAC solvers, at four points within the mesh.

[37] N. Matsakis and J. Stone. Rayon – A data parallelism library for Rust. https://github.com/rayon-rs/rayon, 2022.
[38] V. Michel-Dansac and A. Thomann. TVD-MOOD schemes based on implicit-explicit time integration. *Appl. Math. Comput.*, 433:127397, 2022.
[39] C.-D. Munz, P. Omnes, R. Schneider, E. Sonnendrücker, and U. Voß. Divergence Correction Techniques for Maxwell Solvers Based on a Hyperbolic Model. *J. Comput. Phys.*, 161(2):484–511, 2000.
[40] S. Müller and Y. Stiriba. Fully Adaptive Multiscale Schemes for Conservation Laws Employing Locally Varying Time Stepping. *J. Sci. Comput.*, 30(3):493–531, 2006.
[41] B. Perthame. Boltzmann type schemes for gas dynamics and the entropy property. *SIAM J. Numer. Anal.*, 27(6):1405–1421, 1990.
[42] X. Shi, J. Lin, and Z. Yu. Discontinuous Galerkin spectral element lattice Boltzmann method on triangular element. *Internat. J. Numer. Methods Fluids*, 42(11):1249–1261, 2003.
[43] B. Weber. *Optimisation de code Galerkin Discontinu sur ordinateur hybride. Application à la simulation numérique en électromagnétisme.* Ph.D. Thesis, Université de Strasbourg, November 2018.
[44] J. Yan and D. Jiao. Explicit and unconditionally stable FDTD method without eigenvalue solutions. In *2016 IEEE MTT-S International Microwave Symposium (IMS)*. IEEE, 2016.
[45] K. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. Antennas Propag.*, 14(3):302–307, 1966.