ESAIM: PROCEEDINGS, December 2012, Vol. 38, p. 257-274
F. Coquel, M. Gutnic, P. Helluy, F. Lagoutière, C. Rohde, N. Seguin, Editors

RESOLUTION OF THE VLASOV-MAXWELL SYSTEM BY PIC DISCONTINUOUS GALERKIN METHOD ON GPU WITH OPENCL

Anaïs Crestetto[1] and Philippe Helluy[2]

Abstract. We present an implementation of a Vlasov-Maxwell solver for multicore processors. The Vlasov equation describes the evolution of charged particles in an electromagnetic field, solution of the Maxwell equations. The Vlasov equation is solved by a Particle-In-Cell method (PIC), while the Maxwell system is computed by a Discontinuous Galerkin method. We use the OpenCL framework, which allows our code to run on multicore processors or recent Graphic Processing Units (GPU). We present several numerical applications to two-dimensional test cases.

Introduction

The system of Vlasov-Maxwell equations describes the evolution of charged particles in an electromagnetic field. It has many applications in plasma physics. The Vlasov equation can be approximated by several methods. In this work, we use the popular Particle-In-Cell (PIC) method [2], [17]. The distribution function f representing the charged particles is approximated by a sum of Dirac masses located at the positions and velocities of some macro-particles. The macro-particles move according to the equations of motion under the action of electromagnetic forces. The electromagnetic field is solution of the Maxwell equations. In this work, we use an upwind Discontinuous Galerkin finite element method [14], coupled with a divergence cleaning technique [15].

Our objective is to apply the Vlasov-Maxwell model to the simulation of a diode. In such a device, an intense beam of electrons flows from the cathode to the anode. It is used for instance to generate X-rays in medical imaging systems. The geometry of the diode is naturally axisymmetrical. However this leads to several practical numerical difficulties that we will address in a forthcoming work. Thus we only consider numerical results in the pure two-dimensional case.

Even in a two-dimensional setup, the system of Vlasov-Maxwell has five variables (two positions, two velocities and the time) and the computations may be heavy. In order to reduce the computation time, we develop a code that runs on a GPU (Graphics Processing Unit) or a multicore processor, using OpenCL (Open Computing Language). The parallelism of the Discontinuous Galerkin method is obvious and the method is well adapted to GPU. On the other hand, the PIC method is more tricky to implement efficiently. We use a radix sorting of the particles in order to improve the parallelism [10].

The plan of this paper is as follows. First, we recall the two-dimensional Vlasov-Maxwell model. Then, we describe the Discontinuous Galerkin method that we use for approximating the electromagnetic field. The third part is devoted to the description of the implementation on GPU. Finally, we present numerical results and performance tests.

[1] INRIA Nancy-Grand Est & IRMA, Université de Strasbourg
[2] IRMA, Université de Strasbourg

Article published online by EDP Sciences and available at http://www.esaim-proc.org or http://dx.doi.org/10.1051/proc/201238014

1. Vlasov-Maxwell model

In this section, we recall the two-dimensional Maxwell equations. We consider only the Transverse Electric (TE) mode. It is important, for numerical reasons, to ensure the charge conservation in the numerical method. This can be achieved by a technique of divergence cleaning, that we then describe. Finally, we recall the Vlasov equation and the PIC kinetic method that we use to solve it.

1.1. Maxwell equations

In a three-dimensional setup, the Maxwell equations are

$$\begin{aligned} \partial_t \boldsymbol{E} - c^2 \mathbf{rot}\ \boldsymbol{B} &= -\frac{\boldsymbol{J}}{\varepsilon_0}, && (1)\\ \partial_t \boldsymbol{B} + \mathbf{rot}\ \boldsymbol{E} &= 0, && (2)\\ \mathrm{div}\ \boldsymbol{E} &= \frac{\rho}{\varepsilon_0}, \quad \text{at } t = 0, && (3)\\ \mathrm{div}\ \boldsymbol{B} &= 0, \quad \text{at } t = 0, && (4)\\ + \quad \text{initial conditions} \quad + & \quad \text{boundary conditions}, && (5) \end{aligned}$$

where $\boldsymbol{E}(\boldsymbol{x},t)$ is the electric field, $\boldsymbol{B}(\boldsymbol{x},t)$ the magnetic field, $\boldsymbol{J}(\boldsymbol{x},t)$ the current density, $\rho(\boldsymbol{x},t)$ the charge density, $\boldsymbol{x}$ the position, t the time. In the following, we consider dimensionless equations: the speed of light in vacuum c and the permittivity of free space ε_0 are taken equal to 1.

1.1.1. *The two dimensional transverse electric mode in cartesian geometry*

When the electromagnetic field does not depend on the third space variable, we can consider a two-dimensional problem called the Tranverse Electric (TE) mode. In cartesian coordinates

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \tag{6}$$

the fields are reduced to

$$\boldsymbol{E} = \begin{pmatrix} E_x \\ E_y \\ 0 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0 \\ 0 \\ B_z \end{pmatrix} \quad \text{and} \quad \boldsymbol{J} = \begin{pmatrix} J_x \\ J_y \\ 0 \end{pmatrix}. \tag{7}$$

We suppose that they do not depend on the z variable. Having, for a vector

$$\boldsymbol{u} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}, \quad \mathbf{rot}\ \boldsymbol{u} = \begin{pmatrix} \partial_y u_z - \partial_z u_y \\ \partial_z u_x - \partial_x u_z \\ \partial_x u_y - \partial_y u_x \end{pmatrix} \quad \text{and} \quad \mathrm{div}\ \boldsymbol{u} = \partial_x u_x + \partial_y u_y + \partial_z u_z, \tag{8}$$

the equations of Ampère (1) and Faraday (2) become

$$\begin{aligned} \partial_t E_x - \partial_y B_z &= -J_x, && (9)\\ \partial_t E_y + \partial_x B_z &= -J_y, && (10)\\ \partial_t B_z + \partial_x E_y - \partial_y E_x &= 0. && (11) \end{aligned}$$

We can introduce the unknown vector $\boldsymbol{W}$ and the source term vector $\boldsymbol{S}$ by

$$\boldsymbol{W} = \begin{pmatrix} E_x \\ E_y \\ B_z \end{pmatrix}, \quad \boldsymbol{S} = \begin{pmatrix} -J_x \\ -J_y \\ 0 \end{pmatrix} \tag{12}$$

and a matrix $\boldsymbol{M}$ of boundary conditions, to obtain the following boundary value problem

$$\begin{aligned} \partial_t \boldsymbol{W} + \boldsymbol{A}^i \partial_i \boldsymbol{W} &= \boldsymbol{S}, && \text{on } \Omega \times [0,T], && (13)\\ \boldsymbol{M}\boldsymbol{W} &= \boldsymbol{M}\boldsymbol{W}_{inc}, && \text{on } \partial\Omega \times [0,T], && (14)\\ \boldsymbol{W}(\boldsymbol{x},0) &= \boldsymbol{W}_0(\boldsymbol{x}), && \text{at } t=0, && (15) \end{aligned}$$

where $\boldsymbol{A}^i\partial_i$ stands for $\boldsymbol{A}^1\partial_1 + \boldsymbol{A}^2\partial_2 = \boldsymbol{A}^1\partial_x + \boldsymbol{A}^2\partial_y$ (sum on repeated indices),

$$\boldsymbol{A}^1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \boldsymbol{A}^2 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \tag{16}$$

$\boldsymbol{W}_{inc}$ is an incident field and $\boldsymbol{W}_0$ the initial condition.

(13) is considered in the sense of distributions. Thus, if the solution $\boldsymbol{W}$ admits a surface of discontinuity Σ, oriented by a normal vector $\boldsymbol{n}$, we have

$$\left[\boldsymbol{A}^i n_i \boldsymbol{W}\right] = 0, \tag{17}$$

where $[Q]$ is the jump of Q through Σ. The system of first order PDE (13) is linear and hyperbolic. It is also known as a Friedrichs system. It has a zero eigenvalue, which is also related to the charge conservation (3). The existence and uniqueness of such boundary value problems have been studied by Lax and Philips [13] (see also [8], for instance). In a few words, the condition

$$\forall\ \boldsymbol{W} \in \text{Ker}\ \boldsymbol{M}, \quad \boldsymbol{A}^i n_i \boldsymbol{W} \cdot \boldsymbol{W} \geq 0 \quad \text{on} \quad \partial\Omega \tag{18}$$

implies the uniqueness of the solution. For its existence, we have to verify the following property:

$$\dim \text{Ker}\ \boldsymbol{M} \text{ equals the number of non-negative eigenvalues of } \boldsymbol{A}^i n_i, \text{ with multiplicity.} \tag{19}$$

1.1.2. *Divergence cleaning*

Theoretically, if the Gauss law is verified initially

$$\text{div}\ \boldsymbol{E} = \rho \quad \text{at} \quad t = 0, \tag{20}$$

and if the charge conservation holds

$$\partial_t \rho + \text{div}\ \boldsymbol{J} = 0 \quad \forall\ t, \tag{21}$$

then the Gauss law remains true at any time. But this property may be not satisfied numerically. It is not always necessary to correct this problem, but in some cases, it leads to instabilities or large errors between the exact solution and the computed one.

Munz *et al.* presented in [15] (and also in [6] for the MHD equations) a simple method for correcting the numerical divergence errors in Maxwell solvers. It improves the precision of the Gauss law numerically. The hyperbolic correction (see [15]) consists in introducing in the Maxwell system an additional velocity $\chi \geq 0$ and

a Lagrange multiplier ϕ, so that the system becomes

$$\begin{aligned}
\partial_t E_x - \partial_y B_z + \chi \partial_x \phi &= -J_x, && (22)\\
\partial_t E_y + \partial_x B_z + \chi \partial_y \phi &= -J_y, && (23)\\
\partial_t B_z + \partial_x E_y - \partial_y E_x &= 0, && (24)\\
\partial_t \phi + \chi \left(\partial_x E_x + \partial_y E_y\right) &= \chi \rho. && (25)
\end{aligned}$$

The mixed hyperbolic-parabolic correction (see [6]) consists in replacing (25) by

$$\partial_t \phi + \chi \left(\partial_x E_x + \partial_y E_y\right) = \chi \rho - \frac{\chi}{\chi_p} \phi, \tag{26}$$

with χ_p an other factor > 0. The new system is hyperbolic, with non-zero velocities. The effect of the parameter χ is to propagate the divergence errors up to the boundary at the velocity χ. The choice of boundary conditions is thus essential to obtain an effective correction. The effect of the parameter χ_p is to produce a zero order damping source term. This parameter cannot be to small for avoiding stiff source terms when one uses an explicit time solver. When $\chi = 0$, we recover the Maxwell system. The correction is activated when we take $\chi > 0$ and $\chi_p > 0$ ($\chi_p \to \infty$ leads to the hyperbolic correction). Then, the new vector of unknown and the new source term are

$$\boldsymbol{W} = \begin{pmatrix} E_x \\ E_y \\ B_z \\ \phi \end{pmatrix}, \quad \boldsymbol{S} = \begin{pmatrix} -J_x \\ -J_y \\ 0 \\ \chi\rho - \frac{\chi}{\chi_p}\phi \end{pmatrix} \tag{27}$$

and we have now

$$\boldsymbol{A}^1 = \begin{pmatrix} 0 & 0 & 0 & \chi \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \chi & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}^2 = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & \chi \\ -1 & 0 & 0 & 0 \\ 0 & \chi & 0 & 0 \end{pmatrix}. \tag{28}$$

1.2. **Vlasov equation and Particle-In-Cell method**

For the kinetic simulation of charged particles in an electromagnetic field, we couple the Maxwell equations with the Vlasov equation, which is a transport equation written in the $(\boldsymbol{x}, \boldsymbol{v})$ phase space ($\boldsymbol{v}$ denoting the velocity),

$$\partial_t f\left(\boldsymbol{x}, \boldsymbol{v}, t\right) + \boldsymbol{v} \cdot \partial_{\boldsymbol{x}} f\left(\boldsymbol{x}, \boldsymbol{v}, t\right) + \frac{q}{m}\left(\boldsymbol{E} + \boldsymbol{v} \wedge \boldsymbol{B}\right) \cdot \partial_{\boldsymbol{v}} f\left(\boldsymbol{x}, \boldsymbol{v}, t\right) = 0. \tag{29}$$

The distribution function f represents these particles, q denotes their charge and m their mass. A popular method for solving this transport equation is the Particle-In-Cell (PIC) method (see [17] or [2]). It consists in approximating f by a sum of Dirac masses centered at the positions $\boldsymbol{x}_k = (x_k, y_k, 0)^T$ and velocities $\boldsymbol{v}_k = (u_k, v_k, 0)^T$ of N macro-particles (each one of them stands for a set of physical particles) of weight ω_k ($k = 1, \ldots, N$)

$$f\left(\boldsymbol{x}, \boldsymbol{v}, t\right) = \sum_{k=1}^{N} \omega_k \; \delta\left(\boldsymbol{x} - \boldsymbol{x}_k\left(t\right)\right) \delta\left(\boldsymbol{v} - \boldsymbol{v}_k\left(t\right)\right). \tag{30}$$

The Vlasov equation (29) is satisfied if the macro-particles obey the Newton law of motion

$$\begin{aligned}
\frac{d\boldsymbol{x}}{dt} &= \boldsymbol{v}, && (31)\\
\frac{d\boldsymbol{v}}{dt} &= \frac{q}{m}\left(\boldsymbol{E} + \boldsymbol{v} \wedge \boldsymbol{B}\right). && (32)
\end{aligned}$$

The expression of the charge and current densities are given by

$$\rho = q \int_{\boldsymbol{v}} f \, d\boldsymbol{v}, \tag{33}$$

$$\boldsymbol{J} = q \int_{\boldsymbol{v}} \boldsymbol{v} f \, d\boldsymbol{v}, \tag{34}$$

in such a way that the coupling between Maxwell and Vlasov is indeed a two-way coupling.

1.2.1. *PIC method*

A typical PIC algorithm is then the following:

- we compute the current $\boldsymbol{J}$ (it is a mathematical measure) from the particle positions and velocities. We then project this measure on the finite element space used for the Maxwell equations approximation,
- we compute the electromagnetic field on the mesh following the Discontinuous Galerkin method,
- if it is necessary (it depends on the test case), we create particles,
- we compute $\boldsymbol{E}\left(\boldsymbol{x}_k\left(t\right),t\right)$ and $\boldsymbol{B}\left(\boldsymbol{x}_k\left(t\right),t\right)$ the electric and magnetic fields at the position of each macro-particle k,
- macro-particles are then moved following the equations of motion (31), (32), using $\boldsymbol{E}\left(\boldsymbol{x}_k\left(t\right),t\right)$ and $\boldsymbol{B}\left(\boldsymbol{x}_k\left(t\right),t\right)$.

1.2.2. *Emission of particles*

In some test cases, for example for the coil case described later, particles are initially in the domain and their number is constant. However, in a physical diode, electrons are extracted from the cathode by the electric field. We thus have to describe the particle emission process. We denote by $\boldsymbol{n}$ the outward normal vector to $\partial\Omega$. We also note $\boldsymbol{n} = (n_x, n_y, n_z)^T = (n_1, n_2, n_3)^T$. When particles are electrons, the charge $q < 0$, the weight $\omega_k > 0$, $k = 1, \ldots, N$ and the normal component of the electric field $\boldsymbol{E} \cdot \boldsymbol{n}$ has to be positive for extracting electrons.

In each emitting cell L touching the cathode C, the emission law is the following:

- if $\frac{\boldsymbol{E}\cdot\boldsymbol{n}}{q} < 0$ on $\partial L \cap C$, we compute the charge defect

$$Q_L = -q_L + \int_{\partial L \setminus C} \boldsymbol{E} \cdot \boldsymbol{n}, \tag{35}$$

  where

$$q_L = q \sum_{\boldsymbol{x}_k \in L} \omega_k \tag{36}$$

  is the charge in the cell L before the emission,
- if $\frac{Q_L}{q} > 0$, we create n_L particles in L: their positions are chosen randomly in the cell (or in a thin layer in the cell, close to the boundary), their velocities are zero (or numerically close to zero) and their weights are

$$\omega_k = \frac{1}{q}\frac{Q_L}{n_L}, \quad k = 1, \ldots, n_L. \tag{37}$$

  After the emission, we have thus

$$q_L = \int_{\partial L \setminus C} \boldsymbol{E} \cdot \boldsymbol{n}, \tag{38}$$

  which is compatible with the Gauss law $\operatorname{div} \boldsymbol{E} = \rho$ and tends to impose the condition $\boldsymbol{E} \cdot \boldsymbol{n} = 0$ on the cathode. The choice of n_L depends on the mesh discretization and on the total number of particles that we want to reach in the domain, for good precision. In our simulations, we often take $n_L = 10$.

Moreover, we detect the particles leaving the domain: we do not move them anymore. As time advances, we can also replace them by new particles in order to gain memory storage space.

2. Discontinuous Galerkin method and boundary conditions

For solving the Maxwell equations, we use the upwind Discontinuous Galerkin (or DG in short) method. In this section, we first recall the weak formulation. We then explain how to implement some boundary conditions (Silver-Müller, metallic boundary). The Discontinuous Galerkin method has been introduced by Reed and Hill in [16] and Lesaint and Raviart in [14] for neutron transport problems. It has then been analyzed and extended by several authors. For theoretical aspects we refer for instance to [11]. The extension to nonlinear system of conservation laws has been extensively studied by Cockburn and coworkers, see for instance [4]. For applications to the Maxwell system, see (for instance) [3, 5, 7] and included references.

2.1. Weak formulation

We consider the boundary value problem given by (13), (14), (15)

$$\begin{aligned} \partial_t \boldsymbol{W} + \boldsymbol{A}^i \partial_i \boldsymbol{W} &= \boldsymbol{S}, && \text{on } \Omega \times [0,T], && (39)\\ \boldsymbol{M}\boldsymbol{W} &= \boldsymbol{M}\boldsymbol{W}_{inc}, && \text{on } \partial\Omega \times [0,T], && (40)\\ \boldsymbol{W}(\boldsymbol{x},0) &= \boldsymbol{W}_0(\boldsymbol{x}), && \text{at } t=0, && (41) \end{aligned}$$

where the open set Ω denotes the computational domain and $\partial\Omega$ its boundary. We consider a partition of Ω into a finite number of disjoint open sets $(\Omega_\ell)_{\ell\in I}$ such that $\overline{\Omega} = \cup_\ell \overline{\Omega_\ell}$. We suppose that on each Ω_ℓ the restriction of $\boldsymbol{W}$ to Ω_ℓ is in $(H^1(\Omega_\ell))^4$. This property defines the set $\mathcal{H}$ in which we search $\boldsymbol{W}$. The regularity domain of $\boldsymbol{W}$ is $R = \cup_\ell \Omega_\ell$ and its set of discontinuities is $D = \cup_\ell \partial\Omega_\ell$. D is oriented by a unit normal vector $\boldsymbol{n} = (n_x, n_y)^T$. We suppose that this normal vector is outward to Ω on $\partial\Omega \cap D$. We denote by $\boldsymbol{W}_L$ the value of $\boldsymbol{W}$ on the side $-\boldsymbol{n}$ and $\boldsymbol{W}_R$ its value on the side $\boldsymbol{n}$ of the discontinuity. We consider test functions that have the same discontinuities as the unknown $\boldsymbol{W}$.

With these notations and by taking a vectorial test functions $\boldsymbol{\psi}$, we can derive the following weak formulation: find $\boldsymbol{W}$ in $\mathcal{H}$ such that for all $\boldsymbol{\psi}$ in $\mathcal{H}$, we have

$$\begin{aligned} \int_R \partial_t \boldsymbol{W}\cdot\boldsymbol{\psi} + \int_R \left(\boldsymbol{A}^i \partial_i \boldsymbol{W}\right)\cdot\boldsymbol{\psi} + \int_{D\cap\Omega} \left(\boldsymbol{A}^i n_i^+ \left(\boldsymbol{W}_R - \boldsymbol{W}_L\right)\right)\cdot\boldsymbol{\psi}_R + \left(\boldsymbol{A}^i n_i^- \left(\boldsymbol{W}_R - \boldsymbol{W}_L\right)\right)\cdot\boldsymbol{\psi}_L \\ + \int_{\partial\Omega} \left(\boldsymbol{M}\boldsymbol{W}_L\right)\cdot\boldsymbol{\psi}_L = \int_R \boldsymbol{S}\cdot\boldsymbol{\psi} + \int_{\partial\Omega} \left(\boldsymbol{M}\boldsymbol{W}_{inc}\right)\cdot\boldsymbol{\psi}_L, \end{aligned} \tag{42}$$

where $\boldsymbol{A}^i n_i^+$ is the positive part of $\boldsymbol{A}^i n_i$, and $\boldsymbol{A}^i n_i^-$ is the negative part, such that $\boldsymbol{A}^i n_i^+ + \boldsymbol{A}^i n_i^- = \boldsymbol{A}^i n_i$. For obtaining $\boldsymbol{A}^i n_i^+$ (respectively $\boldsymbol{A}^i n_i^-$) we simply cancel the negative (respectively positive) eigenvalues of $\boldsymbol{A}^i n_i$. The expressions of $\boldsymbol{A}^i n_i$, $\boldsymbol{A}^i n_i^+$ and $\boldsymbol{A}^i n_i^-$ are given by

$$\boldsymbol{A}^i n_i = \begin{pmatrix} 0 & 0 & -n_y & \chi n_x \\ 0 & 0 & n_x & \chi n_y \\ -n_y & n_x & 0 & 0 \\ \chi n_x & \chi n_y & 0 & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{A}^i n_i^s = \frac{1}{2} \begin{pmatrix} s\frac{n_y^2+\chi n_x^2}{r} & s\frac{n_y n_x(\chi-1)}{r} & -n_y & \chi n_x \\ s\frac{n_y n_x(\chi-1)}{r} & s\frac{n_x^2+\chi n_y^2}{r} & n_x & \chi n_y \\ -n_y & n_x & sr & 0 \\ \chi n_x & \chi n_y & 0 & s\chi r \end{pmatrix}, \tag{43}$$

s being the sign $\pm$ and $r = \sqrt{n_x^2 + n_y^2}$. It is easy to check that, thanks to these definitions, the weak formulation does not depend on the chosen orientation of D.

Let us now choose one given Ω_ℓ. Without loss of generality, we can suppose that the normal vector $\boldsymbol{n}$ on D is outward to Ω_ℓ. By taking $\boldsymbol{\psi}$ whose support is contained in Ω_ℓ, we obtain

$$\begin{aligned}\int_{\Omega_\ell} \partial_t \boldsymbol{W}\cdot\boldsymbol{\psi} + \int_{\Omega_\ell} \left(\boldsymbol{A}^i\partial_i \boldsymbol{W}\right)\cdot\boldsymbol{\psi} + \int_{R\cap\partial\Omega_\ell} \left(\boldsymbol{A}^i n_i^- \left(\boldsymbol{W}_R - \boldsymbol{W}_L\right)\right)\cdot\boldsymbol{\psi}_L \\ + \int_{\partial\Omega\cap\partial\Omega_\ell} (\boldsymbol{M}\boldsymbol{W}_L)\cdot\boldsymbol{\psi}_L = \int_{\Omega_\ell} \boldsymbol{S}\cdot\boldsymbol{\psi} + \int_{\partial\Omega\cap\partial\Omega_\ell} (\boldsymbol{M}\boldsymbol{W}_{inc})\cdot\boldsymbol{\psi}_L,\end{aligned} \tag{44}$$

which is equivalent to the so called conservative formulation

$$\begin{aligned}\int_{\Omega_\ell} \partial_t \boldsymbol{W}\cdot\boldsymbol{\psi} - \int_{\Omega_\ell} \boldsymbol{W}\cdot\left(\boldsymbol{A}^i\partial_i\boldsymbol{\psi}\right) + \int_{R\cap\partial\Omega_\ell} \left(\boldsymbol{A}^i n_i^+ \boldsymbol{W}_L + \boldsymbol{A}^i n_i^- \boldsymbol{W}_R\right)\cdot\boldsymbol{\psi}_L \\ + \int_{\partial\Omega\cap\Omega_\ell} \left(\left(\boldsymbol{M} + \boldsymbol{A}^i n_i\right)\boldsymbol{W}_L\right)\cdot\boldsymbol{\psi}_L = \int_{\Omega_\ell} \boldsymbol{S}\cdot\boldsymbol{\psi} + \int_{\partial\Omega\cap\partial\Omega_\ell} (\boldsymbol{M}\boldsymbol{W}_{inc})\cdot\boldsymbol{\psi}_L.\end{aligned} \tag{45}$$

The numerical DG approximation consists then in restricting the weak formulation to a finite dimensional subspace of $\mathcal{H}$. We can, for instance, consider that the approximated fields are polynomial in each open set Ω_ℓ. The choice of the Ω_ℓ corresponds then to the construction of an unstructured mesh of Ω. An advantage of the DG method is that this mesh is not necessarily conforming, but we will not use this feature here.

2.2. **Energy estimation**

In order to study the stability of the method, we make an energy estimation without source terms: $\boldsymbol{S} = 0$ and $\boldsymbol{W}_{inc} = 0$. By taking $\boldsymbol{\psi} = \boldsymbol{W}$ in (42), we obtain

$$\begin{aligned}\partial_t \int_R \frac{1}{2}\boldsymbol{W}^2 + \int_R \frac{1}{2}\partial_i\left(\left(\boldsymbol{A}^i\boldsymbol{W}\right)\cdot\boldsymbol{W}\right) + \int_{D\cap\Omega} \left(\boldsymbol{A}^i n_i^+ \left(\boldsymbol{W}_R - \boldsymbol{W}_L\right)\right)\cdot\boldsymbol{W}_R + \left(\boldsymbol{A}^i n_i^- \left(\boldsymbol{W}_R - \boldsymbol{W}_L\right)\right)\cdot\boldsymbol{W}_L \\ + \int_{\partial\Omega} (\boldsymbol{M}\boldsymbol{W}_L)\cdot\boldsymbol{W}_L = 0.\end{aligned} \tag{46}$$

The energy $\mathcal{E}$ is defined by the integral $\mathcal{E} = \int_R \frac{1}{2}\boldsymbol{W}^2$. Stokes theorem and simple calculations give us

$$\partial_t\mathcal{E} + \int_{D\cap\Omega} \frac{1}{2}\left(\mid \boldsymbol{A}^i n_i \mid [\boldsymbol{W}]\right)\cdot[\boldsymbol{W}] + \int_{\partial\Omega} \left(\left(\frac{1}{2}\boldsymbol{A}^i n_i + \boldsymbol{M}\right)\boldsymbol{W}_L\right)\cdot\boldsymbol{W}_L = 0, \tag{47}$$

$[\boldsymbol{W}]$ being the jump of $\boldsymbol{W}$ on the discontinuity and $\mid \boldsymbol{A}^i n_i \mid$ denoting $\boldsymbol{A}^i n_i^+ - \boldsymbol{A}^i n_i^-$ is a positive matrix. A sufficient condition for the decrease of the energy is thus

$$\frac{1}{2}\boldsymbol{A}^i n_i + \boldsymbol{M} > 0 \tag{48}$$

(but not necessarily symmetric). This property also implies the stability and the uniqueness of the solution (it implies (18)). However the existence of the solution is not ensured by the energy estimate. We have also to consider the more subtle additional condition, obtained from (19),

$$\dim \text{Ker } \boldsymbol{A}^i n_i^- = \dim \text{Ker } \boldsymbol{M}. \tag{49}$$

This condition ensures that the spatial part of the Maxwell operator is maximal monotone and allows us to apply the Hille-Yosida theorem. The theory is proved in [13] (and also explained in [8], for instance). Moreover, (49) implies

$$\text{Ker } \boldsymbol{A}^i n_i \subset \text{Ker } \boldsymbol{M}. \tag{50}$$

A nice feature of the divergence correction model is that Ker $\boldsymbol{A}^i n_i = 0$ for $\chi > 0$. This will permit us to envisage boundary conditions that would not be allowed for the Maxwell system ($\chi = 0$). For instance it is now mathematically possible to impose a given value to the normal component of the electric field $\boldsymbol{E} \cdot \boldsymbol{n}$ on the boundary.

2.3. **Boundary conditions**

The implementation of boundary conditions is of particular importance for the stability of the method. According to the test cases that we want to consider, we will focus on the following boundary conditions:

- Silver-Müller,
- symmetry,
- perfectly conducting wall or metallic boundary.

2.3.1. *Silver-Müller boundary condition*

To impose an incident field $\boldsymbol{W}_{inc}$ on a boundary of the domain and to let the waves leave the domain, we can consider the Silver-Müller boundary condition

$$\boldsymbol{M}\boldsymbol{W} = \boldsymbol{M}\boldsymbol{W}_{inc}. \tag{51}$$

The corresponding matrix

$$\boldsymbol{M} = -\boldsymbol{A}^i n_i^- \tag{52}$$

leads to

$$\frac{1}{2}\boldsymbol{A}^i n_i + \boldsymbol{M} = \frac{1}{2} \mid \boldsymbol{A}^i n_i \mid . \tag{53}$$

Thus, the Lax-Phillips conditions (48), (49) are satisfied. In addition, this condition induces some damping of the divergence errors at the corresponding part of the boundary.

2.3.2. *Symmetry condition*

On a boundary of Ω corresponding to an axis of symmetry $y = 0$, the condition is

$$\boldsymbol{N}\left(\boldsymbol{W} - \boldsymbol{W}_{sym}\right) = 0, \quad \text{with} \quad \boldsymbol{N} = -\boldsymbol{A}^i n_i^- \tag{54}$$

and

$$\boldsymbol{W}_{sym} = \begin{pmatrix} E_{x,L} \\ -E_{y,L} \\ -B_{z,L} \\ \phi_L \end{pmatrix} = \boldsymbol{S}\boldsymbol{W} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{W}. \tag{55}$$

We then obtain the boundary condition

$$\boldsymbol{M}\boldsymbol{W} = 0 \quad \text{with} \quad \boldsymbol{M} = \boldsymbol{N}\left(\boldsymbol{I} - \boldsymbol{S}\right), \tag{56}$$

where $\boldsymbol{I}$ is the identity matrix. The Lax-Philipps conditions are also satisfied.

2.3.3. *Perfectly conducting wall*

For a metallic boundary of Ω, we consider the perfectly conducting boundary condition, which can be written

$$\boldsymbol{E} \wedge \boldsymbol{n} = \boldsymbol{0}. \tag{57}$$

Because we have introduced a divergence correction, we also have to give a boundary condition for the additional variable ϕ. This condition has no physical meaning. It has to be chosen in order to introduce a good numerical damping of the divergence errors. We also have to verify that it is compatible with the evident solution $\phi = 0$

of the corrected Maxwell equations, because in this case, the corrected equations and the physical Maxwell equations are equivalent. We can take

$$\boldsymbol{M} = \begin{pmatrix} 0 & 0 & 0 & -\chi n_x \\ 0 & 0 & 0 & -\chi n_y \\ n_y & -n_x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{58}$$

such that

$$\{\boldsymbol{W}, \boldsymbol{E} \wedge \boldsymbol{n} = \boldsymbol{0} \text{ and } \phi = 0\} = Ker\left(\boldsymbol{M}\right) \tag{59}$$

and $\boldsymbol{M}$ respects the dissipation and maximality conditions (48), (49). This boundary condition induces a zero energy flux at the boundary. The divergence errors are not damped by this boundary conditions but instead reflected inside the computational domain. This might not be a problem if the errors are absorbed by the source term and at other parts of the boundary.

3. Implementation on GPU

In this section, we first present in a few words the architecture of a Graphics Processing Unit (GPU) and the library that we have chosen, namely the Open Computing Language (OpenCL), for programming such devices. We then explain the consequences of such an architecture on our parallelized implementation. The consequences are different if we consider the Discontinuous Galerkin method [12] or the Particle-In-Cell algorithm [1].

3.1. Computing on GPU with OpenCL

A modern GPU is schematically made of global memory (typically a few Gigabytes) and compute units (typically a few tens). Each compute unit contains processing elements (typically 8) and local memory (a few kilobytes). The same program can be executed on all the processing elements at the same time. The access to the memory follows these three rules:

- all the processing elements have access to the global memory,
- the processing elements have only access to the local memory of their compute unit,
- the access to the local memory is much faster than the access to the global memory.

The programing framework we have chosen is OpenCL. It includes a library of C functions in order to drive the GPU and a C-like language to write the programs (called "kernels") that will be executed on the processing elements. Virtually, it allows to have as many compute units (called "work-groups") and processing elements (called "work-items") as needed: the threads are sent to the GPU, thanks to a mechanism of command queues, on the physical compute units and processing elements. Moreover, this language is portable. It can be executed on different graphics cards and on a multicore (or single-core) CPU. It is also possible to manage several devices in the same program, to share the work between a CPU and several GPU for instance.

3.2. DG implementation

The domain Ω is made of quadrangular cells (with possibly curved boundaries), which correspond to the Ω_ℓ, linked to a reference square $[0,1] \times [0,1]$ by a given transformation function. We expand the fields with respect to a basis of second order polynomial functions. More precisely, in each cell $L = \Omega_\ell$, the approximated field is given by (using the sum of repeated indices convention)

$$\boldsymbol{W}_L\left(\boldsymbol{x}, t\right) \simeq \psi_{L,j}\left(\boldsymbol{x}\right) \boldsymbol{w}_{L,j}\left(t\right), \quad \{\psi_{L,j}\} \text{ linearly independent polynomials of } P_2\left(\mathbb{R}^2\right). \tag{60}$$

When restricted to that finite dimensional space, the formulation (42) becomes: $\forall L, t$, find $\boldsymbol{W}_L(\boldsymbol{x}, t) \simeq \psi_{L,j}(\boldsymbol{x})\, \boldsymbol{w}_{L,j}(t)$ such that $\forall \psi_L = \psi_{L,j}$

$$\begin{aligned} &\int_L \psi_L \partial_t \boldsymbol{W}_L - \int_L (\boldsymbol{A}^i \partial_i \psi_L) \boldsymbol{W}_L + \int_{\partial L \cap \Omega} \psi_L (\boldsymbol{A}^i n_i^+ \boldsymbol{W}_L + \boldsymbol{A}^i n_i^- \boldsymbol{W}_R) \\ &\quad + \int_{\partial L \cap \partial\Omega} \psi_L \left((\boldsymbol{M} + \boldsymbol{A}^i n_i) \boldsymbol{W}_L \right) = \int_L \psi_L \boldsymbol{S} + \int_{\partial L \cap \partial\Omega} \psi_L \boldsymbol{M} \boldsymbol{W}_{inc}. \end{aligned} \tag{61}$$

We end up with a system of ordinary differential equations to solve for the $\boldsymbol{w}_{L,j}(t)$. The initial condition is obtained by the L^2 projection of the exact initial condition on the approximation space. We perform numerical integration with 4 Gauss points in each cell L or edge and in each space direction.

The initialization of the Discontinuous Galerkin method is done on the CPU. For example, we compute and invert the cell local mass matrices on the CPU. All the data are then sent to the GPU.

In order to compute the time derivatives of the unknowns $\boldsymbol{w}_{L,j}(t)$ we compute separately the edge terms and the surface terms.

- For the flux terms
$$\boldsymbol{A}^i n_i^+ \boldsymbol{W}_L + \boldsymbol{A}^i n_i^- \boldsymbol{W}_R, \tag{62}$$
we associate to each Gauss point of each edge a work-item and execute the OpenCL kernel that computes the flux. We store the fluxes in the global memory of the GPU.
- For the surface terms
$$-\int_L (\boldsymbol{A}^i \partial_i \psi_L) \boldsymbol{W}_L - \int_L \psi_L \boldsymbol{S}, \tag{63}$$
we associate a work-item to each basis function of each element. We also collect and integrate the previously computed fluxes.
- Finally, in the same loop we apply locally the previously stored inverted mass matrices. The mass matrix comes from the term
$$\int_L \psi_{i,L} \partial_t \boldsymbol{W}_L = \left(\int_L \psi_{j,L} \psi_{i,L} \right) \boldsymbol{w}'_{L,j}(t). \tag{64}$$
A decisive advantage of the DG method is that the mass matrix is block diagonal.

Finally, we use the second order Heun scheme for the time integration

$$\boldsymbol{W}^\star = \boldsymbol{W}^n + \Delta t \partial_t \boldsymbol{W}^n, \tag{65}$$

$$\boldsymbol{W}^{n+1} = \frac{1}{2} \left(\boldsymbol{W}^n + \boldsymbol{W}^\star + \Delta t \partial_t \boldsymbol{W}^\star \right) = \boldsymbol{W}^n + \frac{\Delta t}{2} \left(\partial_t \boldsymbol{W}^n + \partial_t \boldsymbol{W}^\star \right). \tag{66}$$

The parallelism in this step is obvious because the Heun scheme is applied componentwise. In order to obtain more stability and to take bigger time step, we can also use a third or fourth order Runge-Kutta scheme.

3.3. PIC implementation

For solving the Vlasov equation, we use the previously described PIC method. Each macro-particle k stands for a set of physical electrons or ions, which gives it a positive weight ω_k. In the following, we will call particles these macro-particles.

3.3.1. *Easy parallelism*

If there are particles initially (it depends on the test case), we initialize their positions, velocities and weights on the CPU and send these data to the GPU. Then, we associate one work-item to each particle in order to:

- compute the forces acting on them (the right-hand sides of (31) and (32)): we find the new location of each particle assuming that it does not cross more than one cell during one time step. We obtain $\boldsymbol{E}(\boldsymbol{x}_k(t), t)$ and $\boldsymbol{B}(\boldsymbol{x}_k(t), t)$ by computing the fields at the particle position $\boldsymbol{x}_k$,
- move them by using the same second order Heun scheme: $\boldsymbol{W}$ is replaced by $\boldsymbol{x}_k$ and $\boldsymbol{v}_k$ in (65), (66), $\partial_t \boldsymbol{x}_k$ and $\partial_t \boldsymbol{v}_k$ are given by (31), (32).

For some test cases, we have to consider emission of particles at the cathode. We associate a work-item to each emitting cell. As explained in 1.2.2, we compute the charge to be added in each emitting cell and initialize the positions, velocities and weights of the newly created particles.

Creation of particles prevents the Gauss law to be verified numerically. This is why we have considered the divergence correction. In order to improve its effect practically, particles are emitted and moved every p time steps of the Maxwell solver: $\Delta t_{particles} = p\Delta t$ ($p = 10$ for example).

3.3.2. *Current computation*

The difficulty appears when computing the current on each cell. Indeed, all the particles of the same cell have to add their contributions to the current of this cell. They may have to modify the same storage space. If they do it at the same time, we can have memory write conflicts and some contributions can be lost. We should require a memory lock for these operations. Such operations are called atomic in computer science. They are very expensive in computational time, because the parallelism is lost. In order to avoid atomic operations, we have to arrange the computations in another way.

The solution we have chosen is to first sort the particles according to their cell numbers [1]. This is done through an optimized radix sort algorithm described in [10]. It is then easy to count the number of particles in each cell. We associate a work-item to each cell L and loop on the particles of L in order to compute and add their contributions to the current. The particles are easily accessed thanks to the sorting. The current contributions are given by

$$\int_L \psi_L \boldsymbol{J} = \sum_{k \in L} \omega_k \psi_L(\boldsymbol{x}_k) \boldsymbol{v}_k. \tag{67}$$

It is also possible to sort the cells according to the number of particles that they contain, in order to improve load balancing. Indeed, thanks to this second sorting, neighboring processors have approximately the same number of particles and do not wait too much for each other.

4. Numerical results

We present now some numerical results. We first verify the order of our Discontinuous Galerkin method. We then look at the computation time and evaluate the performance of the computations on GPU. Finally, we describe three test cases and their results: the coil, the rectangular diode and the diode with hemispherical cathode.

4.1. **Convergence**

To verify the convergence of our method in cartesian geometry, we mesh a disc with rectangular and curved cells (see Figure 1) and propagate the following plane wave in it:

$$\begin{pmatrix} E_x \\ E_y \\ B_z \\ \phi \end{pmatrix} = \begin{pmatrix} 0 \\ \cos(\pi(x-t)) \\ \cos(\pi(x-t)) \\ 0 \end{pmatrix}. \tag{68}$$

We represent on Figure 2 the magnetic field at time $t = 1$ on a domain containing 768 cells. On Figure 3, we plot the error in L2-norm (and in logarithmic scale) obtained with a small CFL number (equal to 0.001) for killing the time integration error, at time $t = 1$. The number of cells goes from 12 to 12288. We obtain a

convergence rate of 3, which confirm the high order of our method. The saturation of the error at the right side of the curve is probably due to the use of single precision in our computation (as of today, GPU are much more efficient on single precision than on double precision computations).

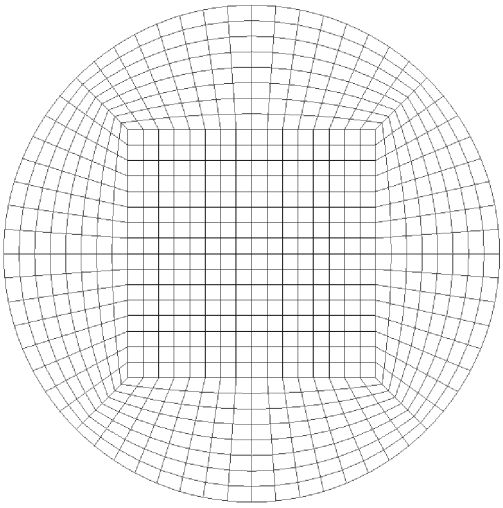

FIGURE 1. Plane wave in a disc: meshing, 768 cells.

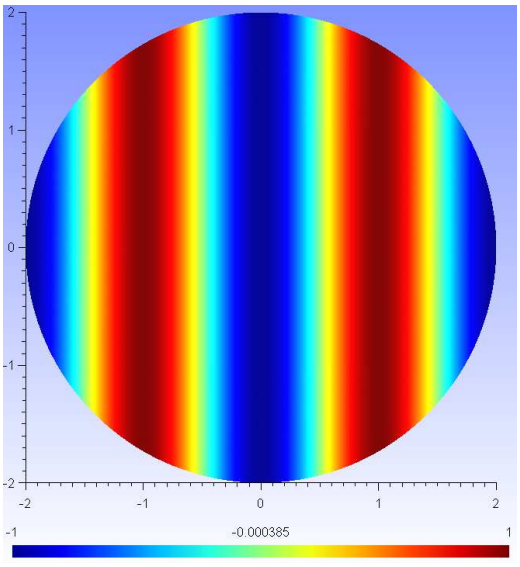

FIGURE 2. Plane wave in a disc: magnetic field at time $t = 1$, 768 cells.

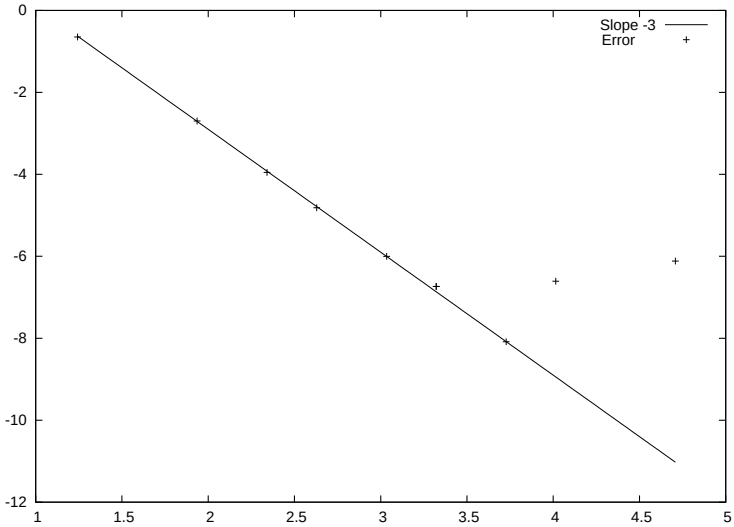

FIGURE 3. Plane wave in a disc: error in L2-norm at time $t = 1$, from 12 to 12288 cells, logarithmic scale in both directions.

4.2. **Performance**

Computing on GPU reduces significantly the computation time. For evaluating the speedup, we compare the time of a simulation on CPU (AMD Phenom II x4 processor), with one core or several cores, and on GPU with different graphic cards. Computations are all done in single (float) precision. The data transfer between the host memory and the GPU memory is not counted, but is only performed at the beginning and at the end of the computation. We consider a plane wave entering in a disc, without particles: we only solve the Maxwell model. We compute the simulation until time $t = 1$, with different values for the CFL (0.05 and 0.01), the mesh containing 3072 or 12288 cells. In Tabular 1 we give the simulation times on CPU (using 1 core and 4 cores) and on two different graphic cards. The ratio between the elapsed time on CPU and the time on GPU, called speedup, is given too. The high speedup confirms that our implementation is efficient on multicore architectures.

Number of cells	Number of iterations	CPU 1 core	CPU 4 cores	ATI Radeon HD 5750	Nvidia GeForce GTX 470	CPU_{1core}/GPU_{Nvidia} (**Speedup**)
3072	329	185 s.	49 s.	4 s.	1 s.	**185**
3072	1644	925 s.	254 s.	16 s.	7 s.	**132**
12288	663	1581 s.	464 s.	24 s.	9 s.	**176**
12288	3312	8133 s.	2455 s.	118 s.	46 s.	**177**

TABLE 1. Calculation times and speedups, without particles.

The PIC method is not so well adapted to the programming on GPU. The performance on computations with particles is not so high but still interesting, as shown in Tabular 2. It corresponds to the emission and the movement of electrons in a rectangular diode, until time $t = 5$ with different values of the cells number and of injected particles (we give the approximated number of particles at final time). The comparison is difficult, because on CPU the sorting of particles is not necessary, but it is done in our code. That is why we only give the speedup between a computation on a CPU with 4 cores (for which the sort is relevant) and a GPU.

Number of cells	Number of iterations	Number of particles	CPU 1 core	CPU 4 cores	ATI Radeon HD 5750	Nvidia GeForce GTX 470	$CPU_{4cores}/GPU_{Nvidia}$ (**Speedup**)
1024	2530	46000	723 s.	446 s.	45 s.	21 s.	**21.2**
1024	2530	92000	914 s.	611 s.	67 s.	31 s.	**19.7**
4096	5060	70000	4450 s.	2319 s.	141 s.	65 s.	**35.7**
4096	5060	110000	4809 s.	2652 s.	242 s.	97 s.	**27.3**

TABLE 2. Calculation times and speedups, with particles.

In order to show the cost of the particles algorithm, we can also look at the speedup between the same computation with and without particles, on the same architecture. We compute Maxwell and Vlasov-Maxwell on the rectangular diode until time $t = 5$. With the Nvidia GeForce GTX 470 card, we obtain the values given in Tabular 3, on a CPU of 4 cores, we obtain Tabular 4.

Number of cells	Number of iterations	Without particles	With $\approx 46000-$ 70000 particles	With/Without	With $\approx 92000-$ 110000 particles	With/Without
1024	2530	4 s.	21 s.	**5.3**	31 s.	**7.8**
4096	5060	26 s.	65 s.	**2.5**	97 s.	**3.7**

TABLE 3. Calculation times on the Nvidia GeForce GTX 470 card, with and without particles.

Number of cells	Number of iterations	Without particles	With ≈ 46000−70000 particles	With/Without	With ≈ 92000−110000 particles	With/Without
1024	2530	137 s.	446 s.	**3.3**	611 s.	**4.5**
4096	5060	1009 s	2319 s.	**2.3**	2652 s.	**2.6**

TABLE 4. Calculation times on a CPU of 4 cores, with and without particles.

For GPU computations, we also describe the cost of each step: Discontinuous Galerkin method (DG), sort of particles (sort) and PIC method (PIC). Results corresponding to a planar diode are given for different parameters in Tabular 5. We move the particles every 5χ time steps.

Number of cells	Number of iterations	Number of particles	χ	DG	sort	PIC
256	1600	133818	1	0.8 s.	12.5 s.	15.4 s.
256	8000	101546	5	4.2 s.	12.5 s.	15.0 s.
256	8000	50769	5	4.1 s.	12.5 s.	7.3 s.
1024	16000	99261	5	24.8 s.	25.1 s.	24.0 s.

TABLE 5. Calculation times of each step on the Nvidia GeForce GTX 470 card.

4.3. **Results of three test cases**

We present now three test cases in cartesian geometry, and their results.

4.3.1. *Coil*

The first studied test case is the simulation of a simple coil. The domain is a disc of radius $R_d = \sqrt{2}$, ions are on a circle of the same center and of a radius $R_c < R_d$, we took here $R_c = 1$. We denote by *in* the inside of the circle of ions and by *out* the outside. The time-independent exact solution writes

$$\begin{pmatrix} E_{x,in} \\ E_{y,in} \\ B_{z,in} \\ \phi_{in} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} E_{x,out} \\ E_{y,out} \\ B_{z,out} \\ \phi_{out} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{69}$$

It is solution of the following boundary value problem

$$\begin{aligned} \partial_t \boldsymbol{W} + \boldsymbol{A}^i \partial_i \boldsymbol{W} &= \boldsymbol{S} && \text{on } \Omega \times [0,T], && (70)\\ \boldsymbol{M}\boldsymbol{W} &= 0 && \text{on } \{(x,y) \text{ s.t. } \sqrt{x^2+y^2} = R_d\} \times [0,T], && (71)\\ \boldsymbol{W}(x,y,0) &= \boldsymbol{W}_{ex}(x,y) && \text{at } t = 0, && (72) \end{aligned}$$

with $\boldsymbol{M}$ the matrix of Silver-Müller boundary conditions (52) and $\boldsymbol{W}_{ex}$ given by (69). The source term is a vector measure given by

$$\boldsymbol{S}(x,y) = \delta_{x^2+y^2=R_c}(x,y)\,(-y,x,0,0)^T. \tag{73}$$

Practically, it is represented by randomly distributed particles on the circle of radius R_c. In this example, the Vlasov-Maxwell coupling is only one-way, because we do not move the particles.

We represent on Figure 4 the magnetic field at time $t = 0.5$, where small perturbations around the exact solution still propagate, and at time $t = 5$ when the numerical stationary state is reached. We have here 3072 cells and 1000 particles. The solution (69) is recovered in a very satisfactory way, except small oscillations on the circle of ions, due to a Gibbs phenomenon: indeed, we represent discontinuous solutions by polynomials.

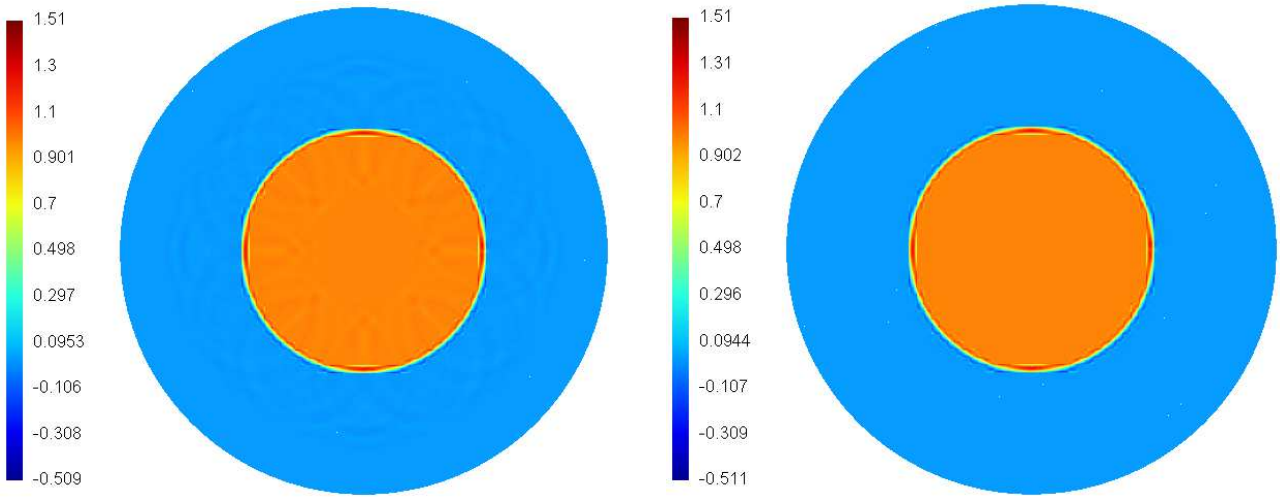

FIGURE 4. Coil case: magnetic field at times $t = 0.5$ (left), and $t = 5$ (right), 1000 particles, 3072 elements.

4.3.2. *Rectangular diode*

As second test case, we consider a planar diode $\Omega = [0, L_x] \times [0, L_y]$ with a cathode $C = \{x = 0\} \times [0, L_y]$ and an anode $A = \{x = L_x\} \times [0, L_y]$. We consider the following boundary conditions:

$$\boldsymbol{M}_1 \boldsymbol{W} = 0 \quad \text{on} \quad (\{x = 0\} \times [0, L_y]) \times [0, T], \tag{74}$$

$$\boldsymbol{M}_2 (\boldsymbol{W} - \boldsymbol{W}_{inc}) = 0 \quad \text{on} \quad (\{x = L_x\} \times [0, L_y]) \times [0, T], \tag{75}$$

with $\boldsymbol{M}_1$ the matrix of perfectly conducting wall given by (58), $\boldsymbol{M}_2$ for a Silver-Müller condition (52) with an incident field

$$\boldsymbol{W}_{inc} = (-1, 0, 0, 0)^T \tag{76}$$

and periodic conditions on $y = 0$ and $y = L_y$.

We represent E_x on Figure 5 at times $t = 1$ and $t = 5$. χ is taken equal to 5 and we move the particles every 25 time steps in order to let the divergence correction act. We see the emission (there is no particle at $t = 0$) and the movement of electrons from the cathode to the anode. We can also remark that on the cathode $E_x = \boldsymbol{E} \cdot \boldsymbol{n} \approx 0$, which is the searched emission condition, also called the Child-Langmuir condition.

For such a planar diode in cartesian geometry, the Child-Langmuir current J_{CL} on the anode for a given potential drop V_0 between the cathode and the anode verifies

$$J_{CL}(L_x) = \frac{4\varepsilon_0}{9{L_x}^2} \sqrt{\frac{2 \mid q \mid}{m}} V_0^{\frac{3}{2}}, \tag{77}$$

where

$$J_{CL} = \int_A \boldsymbol{J} \cdot \boldsymbol{n} \tag{78}$$

and

$$V_0 = V(x = L_x, y_0) - V(x = 0, y_0) = \int_0^{L_x} \partial_x V(x, y_0)\, dx = -\int_0^{L_x} E_x(x, y_0)\, dx, \tag{79}$$

$y_0 \in [0, L_y]$ and V denoting the scalar potential such that $\boldsymbol{E} = -\mathbf{grad} V$. The computed potential drop is given as a function of time on Figure 6 (left). It tends to the value denoted by V_0. The corresponding theoretical J_{CL} is also given on Figure 6 (right) and compared to the computed one.

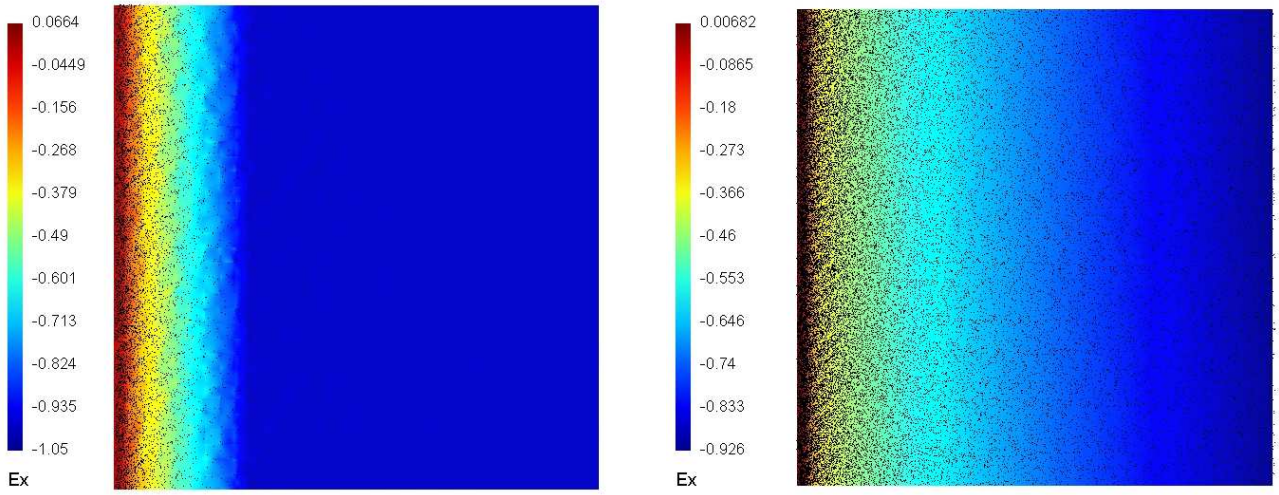

FIGURE 5. Planar diode case: E_x at times $t = 1$ with 8918 particles (left), and $t = 5$ with 44133 particles (right), 1024 elements.

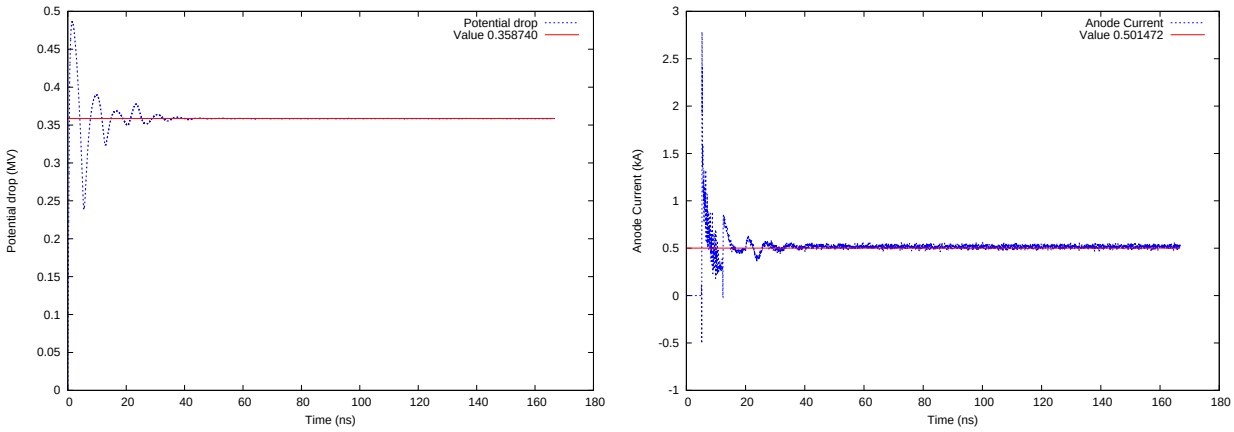

FIGURE 6. Child-Langmuir law: potential drop (left) and Child-Langmuir current (right), computed values (blue) compared to theoretical ones (red).

4.3.3. *Diode with hemispherical cathode*

Finally, we study a diode with hemispherical cathode in cartesian geometry. The computational domain Ω is the interior of the diode. The cathode is defined by

$$C = [0, 0.2] \times \{y = 0.1\} \cup \{\sqrt{(x - 0.2)^2 + y^2} = 0.1\}. \tag{80}$$

The anode is

$$A = [0, 0.4] \times \{y = 0.2\} \cup \{x = 0.4\} \times [0, 0.2]. \tag{81}$$

We solve the following boundary value problem

$$\begin{aligned}
\partial_t \boldsymbol{W} + \boldsymbol{A}^i \partial_i \boldsymbol{W} &= \boldsymbol{S} \quad \text{on} \quad \Omega \times [0,T], && (82)\\
\boldsymbol{M}_1 \boldsymbol{W} &= 0 \quad \text{on} \quad C \cup A, && (83)\\
\boldsymbol{M}_2 \left(\boldsymbol{W} - \boldsymbol{W}_{inc}\right) &= 0 \quad \text{on} \quad (\{x=0\} \times [0.1, 0.2]) \times [0,T], && (84)\\
\boldsymbol{M}_3 \boldsymbol{W} &= 0 \quad \text{on} \quad ([0.3, 0.4] \times \{y=0\}) \times [0,T], && (85)\\
\boldsymbol{W}(x,y,0) &= 0 \quad \text{at} \quad t=0, && (86)
\end{aligned}$$

with $\boldsymbol{M}_1$ the matrix of metallic boundary given by (58), $\boldsymbol{M}_2$ for the Silver-Müller condition (52) with an incident field

$$\boldsymbol{W}_{inc} = (0, -7.8, 0, 0)^T \tag{87}$$

and $\boldsymbol{M}_3$ for the symmetric condition (56).

On Figure 7, we represent E_y and the particles at different times: $t = 0.1$, $t = 0.3$, $t = 0.5$ and $t = 2$ (stationary state), with 768 cells. We can see the extraction of electrons from the cathode and their trajectory towards the anode. On the horizontal part of the cathode, we see that the emission condition $E_y = \boldsymbol{E} \cdot \boldsymbol{n} \approx 0$ is respected.

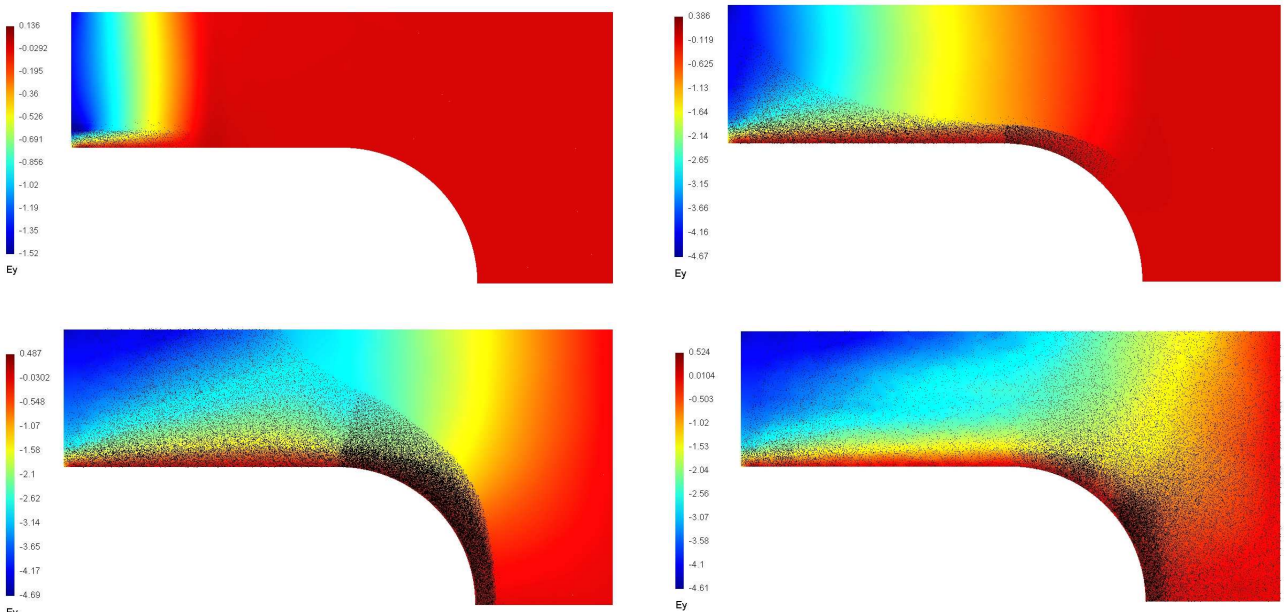

FIGURE 7. Diode case: E_y at times $t = 0.1$ with 1586 particles (top left), $t = 0.3$ with 17420 particles (top right), $t = 0.5$ with 58838 particles (bottom left) and $t = 2$ with 51052 particles (bottom right), 768 elements.

5. CONCLUSION

We have presented the implementation of a Vlasov-Maxwell solver using the OpenCL framework. In order to evaluate the feasibility of scientific computing with OpenCL, we decided to implement a rather complex method: the coupling between a Discontinuous Galerkin scheme and a Particle-In-Cell solver.

The implementation is not necessarily easy but in the end the OpenCL solver can be run on GPU of several brands or on a multicore CPU.

Our solver presents very interesting speedups when compared to CPU computations.

In the implementation, we have discovered that the Discontinuous Galerkin method is well adapted to multicore hardware. The PIC algorithm is more difficult to implement, while the speedups are still interesting.

Our work will now be devoted to more physical validations and to the implementation of the axisymmetrical case.

References

[1] Aubert, D.; Amini, M.; David, R., *A Particle-Mesh Integrator for Galactic Dynamics Powered by GPGPUs*, Lecture Notes in Computer Science **5544**, 874–883, (2009).

[2] Birdsall, C. K.; Langdon, A. B., *Plasma Physics Via Computer Simulation*, Institute of Physics Publishing, Bristol and Philadelphia, (2002).

[3] Bourdel, F.; Mazet, P. A.; Helluy, P., *Resolution of the non-stationary or harmonic Maxwell equations by a discontinuous finite element method. Application to an E.M.I. (electromagnetic impulse) case*, Proceedings of the 10th international conference on computing methods in applied sciences and engineering, (1992).

[4] Cockburn, B.; Hou, S.; Shu, C. W., *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case*, Math. Comp. **54**, (1990).

[5] Cohen, G.; Ferrieres, X.; Pernet, S., *A spatial high-order hexahedral discontinuous Galerkin method to solve Maxwell's equations in time domain*, J. Comput. Phys. **217**, 340–363, (2006).

[6] Dedner, A.; Kemm, F.; Kröner, D.; Munz, C.-D.; Schnitzer, T.; Wesenberg, M., *Hyperbolic Divergence Cleaning for the MHD Equations*, Journal of Computational Physics **175**, 645–673, (2002).

[7] Fezoui, L.; Lanteri, S.; Lohrengel, S.; Piperno, S., *Convergence and stability of a discontinuous Galerkin time-domain method for the 3D heterogeneous Maxwell equations on unstructured meshes*, M2AN Math. Model. Numer. Anal. **39**, 1149–1176, (2005).

[8] Helluy, P., *Résolution numérique des équations de Maxwell harmoniques par une méthode d'éléments finis discontinus*, `http://tel.archives-ouvertes.fr/tel-00657828_v1`, PhD thesis, ENSAE, (1994).

[9] Helluy, P.; Dayma, S., *Convergence of a discontinuous approximation of first-order systems*, C. R. Acad. Sci. Paris Sér. I Math. **319**, 1331–1335, (1994).

[10] Helluy, P., *A portable implementation of the radix sort algorithm in OpenCL*, `http://hal.archives-ouvertes.fr/hal-00596730/fr/`, Technical report, (2011).

[11] Johnson, C.; Pitkäranta, J., *An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation*, Math. Comp. **46**, (1986).

[12] Klöckner, A.; Warburton, T.; Bridge, J.; Hesthaven, J. S., *Nodal discontinuous Galerkin methods on graphics processors.* J. Comput. Phys. **228**, 7863–7882, (2009).

[13] Lax, P. D.; Phillips, R. S., *Local boundary conditions for dissipative symmetric linear differential operators*, Comm. Pure Appl. Math. **13**, 427–455, (1960).

[14] LeSaint, P.; Raviart, P. A., *On a finite element method for solving the neutron transport equation*, Mathematical aspects of finite elements in partial differential equations (C. de Boor, Ed.), Academic Press, 89–145, (1974).

[15] Munz, C.-D.; Omnes, P.; Schneider, R.; Sonnendrücker, E.; Voss, U., *Divergence Correction Techniques for Maxwell Solvers Based on a Hyperbolic Model*, Journal of Computational Physics **161**, 484–511, (2000).

[16] Reed, W. H.; Hill, T. R., *Triangular Mesh Methods for the Neutron Transport Equation*, Los Alamos Scientific Laboratory Report LA-UR-73-479, (1973).

[17] Sonnendrücker, E., *Modèles cinétiques pour la fusion*, Notes du cours de M2, (2008).