Multi-GPU acceleration of a DGTD method for modeling human exposure to electromagnetic waves

Tristan Cabel, Joseph Charles, Stephane Lanteri

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Multi-GPU acceleration of a DGTD method for modeling human exposure to electromagnetic waves

T. Cabel, J. Charles and S. Lanteri

N° 7592

April 2011

Domaine 1

apport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7592--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Multi-GPU acceleration of a DGTD method for modeling human exposure to electromagnetic waves

T. Cabel, J. Charles and S. Lanteri

Domaine : Mathématiques appliquées, calcul et simulation
Équipe-Projet NACHOS

Rapport de recherche n° 7592 — April 2011 — 27 pages

Abstract: We present a high performance computing methodology for the simulation of electromagnetic wave propagation in biological tissues and its application to the numerical evaluation of radio frequency absorption in head tissues as they are exposed to radiation from a cellular phone. For this purpose, the system of time-domain Maxwell equations is discretized in space by a discontinuous Galerkin method which is formulated on a tetrahedral mesh and which relies on a high order interpolation of the electromagnetic field components within a mesh element. The semi-discretized equations are then time integrated by a second order leap-frog scheme. The resulting numerical methodology is adapted to modern parallel computing systems with multiple GPU acceleration cards by adopting a hybrid strategy that combines a coarse grain SPMD programming model for inter-GPU parallelization and a fine grain SIMD programming model for intra-GPU parallelization. The performance improvement thanks to multiple-GPU acceleration is demonstrated through large-scale simulations that are performed on a cluster of GPUs using realistic heterogeneous models of head tissues built from medical images.

Key-words: computational electromagnetics, time-domain Maxwell equations, discontinuous Galerkin method, biological tissues, mobile phone radiation.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Accélération multi-GPU d'une méthode DGTD pour la modélisation de l'exposition des personnes aux ondes électromagnétiques

Résumé : On présente une méthodologie numérique adaptée au calcul haute performance pour la simulation de la propagation d'ondes électromagnétiques dans des tissus biologiques et son application à la dosimétrie numérique de l'exposition des tissus de la tête lorsqu'ils sont exposés au rayonnement d'un téléphone mobile. Pour ce faire, le système des équations de Maxwell est discrétisé par une méthode Galerkin discontinue formulée en maillages tétraédriques et reposant sur une approximation polynomiale d'ordre élevé des composantes du champ électromagnétique au sein d'un élément. Les équations semi-discrétisées sont intégrées en temps au moyen d'un schéma saute-mouton du second ordre. La méthodologie numérique ainsi développée est adaptée aux architectures de calcul haute performance modernes comprenant des cartes accélératrices de type GPU, en adoptant une stratégie hybride combinant un modèle de programmation SPMD *gros grain* pour la parallélisation inter-GPU et un modèle SIMD *grain fin* pour la parallélisation intra-GPU. Les bénéfices en termes de performance résultant de l'accélération multi-GPU sont démontrés par la réalisation de simulations sur un système multi-GPU en utilisant des modèles géométriques réalistes des tissus de la tête construits à partir d'images médicales.

Mots-clés : électromagnétisme numérique, équations de Maxwell en domaine temporel, méthode Galerkin discontinue, tissus biologiques, téléphione mobile.

1 Introduction

Nowadays, numerical modeling is increasingly used and progressively becoming a mandatory path for the study of the interaction of electromagnetic fields with biological tissues. This is in particular the case for the evaluation of the SAR (Specific Absorption Rate) which is a measure of the rate at which electric energy is absorbed by the tissues when exposed to a radio-frequency electromagnetic field. The SAR is defined as the power absorbed per mass of tissue and has units of watts per kilogram. Numerical SAR calculations are intensively considered for dosimetry studies of the exposure of human tissues to microwave radiations from wireless communication systems [1]-[10]-[6] to cite but a few of many examples. These studies are useful for assessing the possible thermal effects (temperature rise in tissues resulting from electric energy dissipation) as well as for compliance testing to regulatory limits. SAR calculations are also relevant for certain medical applications such as for the design of microwave hyperthermia systems [3]-[7]-[19], and for the design of micro-antennas to be implanted inside the human body [14].

Despite the high complexity both in terms of heterogeneity and geometrical features of tissues, the great majority of numerical dosimetry studies have been conducted using the widely known Finite Difference Time Domain (FDTD) method due to Yee [22]. In this method, the whole computational domain is discretized using a structured (Cartesian) grid. Due to the possible straightforward implementation of the algorithm and the availability of computational power, FDTD is currently the leading method for numerical assessment of human exposure to electromagnetic waves. In the particular case of mobile phone radiation, the FDTD method is applied to heterogeneous discretized models of human head tissues built from medical images. Thus, the grid generation process is highly simplified since the voxel based image can be used at a minimal effort as the computational grid for the FDTD method. However it is well known that albeit being highly flexible and second-order accurate in a homogeneous medium, the Yee scheme suffers from serious accuracy degradation when used to model curved objects or when treating material interfaces.

In an attempt to offer an alternative numerical dosimetry methodology which allows for a realistic modeling of geometrical features and tissue interfaces, we consider here the use of a discontinuous Galerkin method formulated on non-uniform tetrahedral meshes. So-called DGTD (Discontinuous Galerkin Time Domain) methods for solving the time-domain Maxwell equations have been proposed by several authors [13]-[9]-[5]. DGTD methods based on discontinuous finite element spaces, easily handle elements of various types and shapes, irregular non-conforming meshes [8], and even locally varying polynomial degree, and hence offer great flexibility in the mesh design. They also lead to (block-) diagonal mass matrices and therefore yield fully explicit, inherently parallel methods when coupled with explicit time stepping. Moreover, continuity is weakly enforced across mesh interfaces by adding suitable bilinear forms (often referred as numerical fluxes) to the standard variational formulations. Despite the achievements so far, it seems that a major limitation to a wider adoption of DGTD methods for large-scale applications in various physical domains and especially for electromagnetic wave propagation problems, is their excessive overhead in terms of computational time and memory occupancy. Not surprisingly, several ongoing research efforts on discontinuous Galerkin methods aim at improving their efficiency both from the numerical and computational point of view. The present work is a contribution to this problematic and we concentrate here on the exploitation of massively parallel computing systems based on graphical processing unit (GPU) acceleration cards.

For more than two decades, the computer industry has witnessed an ever increasing drive of performance improvement. Speeding up processor frequency has considerably improved the processing capabilities given in giga floating point operations per second (GFlops), until physical limits of semiconductor based microelectronics have become a major design concern. In the recent years, design issues for more capable microprocessors have evolved towards the development of multi-core CPUs effectively multiplying the potential performance with several separate processing cores on the same chip, used concurrently. One other particularly noteworthy approach is the use of many-core devices such as Graphical Processing Units (GPUs), accelerating computations orchestrated by a conventional CPU program. Efforts to exploit GPUs, for non-graphical applications has been underway since 2003 and has evolved into programmable and massively parallel computational units with very high memory bandwidth. From this time to the present days a review of research works aiming at harnessing GPUs for the acceleration of partial differential equations solvers would hardly fit into one page. In particular, the development of GPU enabled high order methods is a rapidly growing field especially for what concern computational fluid dynamics applications. Focusing on contributions that are dealing with wave propagation problems, GPUs have been considered for the first time for computational electromagnetics and computational geoseismics applications respectively by Klöckner *et al.* [15] and by Komatitsch *al.* [18]-[17]-[16]. The present work shares several concerns with [15] which describes the development of a GPU enabled DG method formulated on an unstructured tetrahedral mesh for the discretization of hyperbolic systems of conservation laws. Computational results are presented for the solution of the system of 3D time-domain Maxwell equations. As it is the case with the DGTD method at the heart of our study, the one disussed in [15], the local approximation of the unknown field relies on a high order nodal interpolation method which is a key feature in view of exploiting the processing capabilities of the GPU architecture. The main design differences are found in the adopted numerical flux (Klöckner *et al.* consider an upwind flux while we make use of a centered flux) and time-stepping scheme (a 4 steps Runge-Kutta scheme is considered in [15] while we rely on a second order leap-frog scheme). In [15], the CUDA adaptation of the numerical kernels of the studied DG method is detailed, emphasizing the data and computation layout leading to highly tuned implementations of these kernels. Single precision performance on a Nvidia GTX 280 GPU exceeds 250 GFlops for a 9th order interpolation while the speedup relatively to a CPU calculation is maximized for a 4th order interpolation (the observed speedup is 65 in this case). A recent evolution of this work is presented in Gödel *et al.* [12] where the authors discuss the adaptation of a multirate time stepping based DG method for solving the time-domain Maxwell equations on a multiple GPU system. Load balancing issues linked to the local time stepping scheme are treated by a weighted graph partitioning. Performance results are presented for simulations conducted on a system with 4 GPUs but the weak and strong scalability of the resulting hybrid CPU-GPU DG method are not studied. Another widely adopted high order method, namely the spectral element method, is considered in [18]-[17]-[16] for the simulation of seismic wave propagation. In [16], the authors discuss the implementation of the SPECFEM3D software on a large cluster of Nvidia GT200 GPUs using the CUDA environment and non-blocking message-passing using MPI. A similar hybrid SIMD-MIMD parallelization strategy is considered in the present study. Speedups as high as 20 are obtained between the CUDA+MPI version and the highly optimized C+MPI original version of the SPECFEM3D software.

The sequel of the paper is organized as follows: in section 2 we state the initial and boundary value problem to be solved; in section 3 the DGTD method is briefly outlined; the algorithmic adaptation

for obtaining a GPU enabled implementation of this method based on CUDA is discussed in section 4; section 5 is devoted to the application of the resulting numerical methodology to the calculation of the SAR distribution in head tissues, as well as to the analysis of the weak and strong scalability properties on a cluster of GPUs; finally, section 6 concludes this work.

2 The continuous problem

We consider the Maxwell equations in three space dimensions for heterogeneous linear isotropic media. The electric field $\vec{E}(\vec{x},t) = {}^t(E_x, E_y, E_z)$ and the magnetic field $\vec{H}(\vec{x},t) = {}^t(H_x, H_y, H_z)$ verify:

$$\epsilon\partial_t\vec{E} - \mathrm{curl}\vec{H} = -\vec{J} \ , \quad \mu\partial_t\vec{H} + \mathrm{curl}\vec{E} = 0, \tag{1}$$

where the symbol ∂_t denotes a time derivative and $\vec{J}(\vec{x},t)$ is a current source term. These equations are set on a bounded polyhedral domain Ω of $\mathbb{R}^3$. The electric permittivity $\epsilon(\vec{x})$ and the magnetic permeability coefficients $\mu(\vec{x})$ are varying in space, time-invariant and both positive functions. The current source term $\vec{J}$ is the sum of the conductive current $\vec{J_\sigma} = \sigma\vec{E}$ (where $\sigma(\vec{x})$ denotes the electric conductivity of the media) and of an applied current $\vec{J_s}$ associated to a localized source for the incident electromagnetic field. Our goal is to solve system (1) in a domain Ω with boundary $\partial\Omega = \Gamma_a \cup \Gamma_m$, where we impose the following boundary conditions: $\vec{n}\times\vec{E} = 0$ on Γ_m, and $\mathcal{L}(\vec{E},\vec{H}) = \mathcal{L}(\vec{E}_{\mathrm{inc}},\vec{H}_{\mathrm{inc}})$ on Γ_a where $\mathcal{L}(\vec{E},\vec{H}) = \vec{n}\times\vec{E} - \sqrt{\frac{\mu}{\varepsilon}}\vec{n}\times(\vec{H}\times\vec{n})$. Here $\vec{n}$ denotes the unit outward normal to $\partial\Omega$ and $(\vec{E}_{\mathrm{inc}},\vec{H}_{\mathrm{inc}})$ is a given incident field. The first boundary condition is called *metallic* (referring to a perfectly conducting surface) while the second condition is called *absorbing* and takes here the form of the Silver-Müller condition which is a first order approximation of the exact absorbing boundary condition. This absorbing condition is applied on Γ_a which represents an artificial truncation of the computational domain.

3 Discretization by a discontinuous Galerkin method

3.1 Space discretization

The domain Ω is triangulated into a set $\mathcal{T}_h$ of tetrahedra τ_i of size h_i with boundary $\partial\tau_i$ such that $h = \max_{\tau_i\in\mathcal{T}_h} h_i$. To each $\tau_i \in \mathcal{T}_h$, we assign a non-negative integer p_i that is the local interpolation degree. For each τ_i, the parameters ϵ_i and μ_i are respectively the local electric permittivity and magnetic permeability of the medium, which are assumed constant inside the element τ_i. For two distinct tetrahedra τ_i and τ_k in $\mathcal{T}_h$, the intersection $\tau_i \cap \tau_k$ is a convex polyhedron a_{ik} which we will call interface. For each internal interface a_{ik}, we denote by $\vec{n}_{ik}$ the unitary normal vector, oriented from τ_i to τ_k. For the boundary interfaces, the index k corresponds to a fictitious element outside the domain. We denote by $\mathcal{F}_h^I$ the union of all interior interfaces of $\mathcal{T}_h$ and by $\mathcal{F}_h^B$ the union of all boundary interfaces of $\mathcal{T}_h$. Finally, we denote by $\mathcal{V}_i$ the set of indices of the elements which are neighbors of τ_i (having an interface in common). In the following, to simplify the presentation, we set $\vec{J} = 0$. For a given partition $\mathcal{T}_h$, we

seek approximate solutions to (1) in the finite element space:

$$V_{p_i}(\mathcal{T}_h) = \{\vec{v} \in L^2(\Omega)^3 : \vec{v}_{|\tau_i} \in (\mathbb{P}_{p_i}[\tau_i])^3, \quad \forall \tau_i \in \mathcal{T}_h\}, \tag{2}$$

where $\mathbb{P}_{p_i}[\tau_i]$ denotes the space of nodal polynomial functions of degree at most p_i inside τ_i. Following the discontinuous Galerkin approach, the local electric and magnetic fields $(\vec{\mathbf{E}}_i, \vec{\mathbf{H}}_i)$ are defined as combinations of linearly independent basis vector fields $\vec{\varphi}_{ij}$. Let $\mathcal{P}_i = \text{Span}(\vec{\varphi}_{ij},\ 1 \leq j \leq d_i)$ where d_i denotes the number of degrees of freedom inside τ_i. The approximate fields $(\vec{\mathbf{E}}_h, \vec{\mathbf{H}}_h)$, defined by $(\forall i, \vec{\mathbf{E}}_{h|\tau_i} = \vec{\mathbf{E}}_i, \vec{\mathbf{H}}_{h|\tau_i} = \vec{\mathbf{H}}_i)$ are allowed to be completely discontinuous across element boundaries. For such a discontinuous field $\vec{\mathbf{U}}_h$, we define its average $\{\vec{\mathbf{U}}_h\}_{ik}$ through any internal interface a_{ik}, as $\{\vec{\mathbf{U}}_h\}_{ik} = (\vec{\mathbf{U}}_{i|a_{ik}} + \vec{\mathbf{U}}_{k|a_{ik}})/2$. Note that for any internal interface a_{ik}, $\{\vec{\mathbf{U}}_h\}_{ki} = \{\vec{\mathbf{U}}_h\}_{ik}$. Because of this discontinuity, a global variational formulation cannot be obtained. However, dot-multiplying (1) by $\vec{\varphi} \in \mathcal{P}_i$, integrating over each single element τ_i and integrating by parts, yields:

$$\left\{ \begin{array}{l} \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \epsilon_i \partial_t \vec{\mathbf{E}} \ = \ \int_{\tau_i} \text{curl}\vec{\varphi} \cdot \vec{\mathbf{H}} - \int_{\partial\tau_i} \vec{\varphi} \cdot (\vec{\mathbf{H}} \times \vec{n}), \\ \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \mu_i \partial_t \vec{\mathbf{H}} = -\int_{\tau_i} \text{curl}\vec{\varphi} \cdot \vec{\mathbf{E}} + \int_{\partial\tau_i} \vec{\varphi} \cdot (\vec{\mathbf{E}} \times \vec{n}). \end{array} \right. \tag{3}$$

In Eq. (3), we now replace the exact fields $\vec{\mathbf{E}}$ and $\vec{\mathbf{H}}$ by the approximate fields $\vec{\mathbf{E}}_h$ and $\vec{\mathbf{H}}_h$ in order to evaluate volume integrals. For integrals over $\partial\tau_i$, a specific treatment must be introduced since the approximate fields are discontinuous through element faces, leading to the definition of a *numerical flux*. We choose to use a fully centered numerical flux, i.e., $\forall i, \forall k \in \mathcal{V}_i$, $\vec{\mathbf{E}}_{|a_{ik}} \simeq \{\vec{\mathbf{E}}_h\}_{ik}$, $\vec{\mathbf{H}}_{|a_{ik}} \simeq \{\vec{\mathbf{H}}_h\}_{ik}$. The metallic boundary condition on a boundary interface $a_{ik} \in \Gamma_m$ (k in the element index of the fictitious neighboring element) is dealt with *weakly*, in the sense that traces of fictitious fields $\vec{\mathbf{E}}_k$ and $\vec{\mathbf{H}}_k$ are used for the computation of numerical fluxes for the boundary element τ_i. More precisely, we set $\vec{\mathbf{E}}_{k|a_{ik}} = -\vec{\mathbf{E}}_{i|a_{ik}}$ and $\vec{\mathbf{H}}_{k|a_{ik}} = \vec{\mathbf{H}}_{i|a_{ik}}$. Similarly, the absorbing boundary condition is taken into account through the use of a fully upwind numerical flux for the evaluation of the corresponding boundary integral over $a_{ik} \in \Gamma_a$ (see [9] for more details). Evaluating the surface integrals in (3) using the centered numerical flux, and re-integrating by parts yields:

$$\left\{ \begin{array}{l} \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \epsilon_i \partial_t \vec{\mathbf{E}}_i \ = \frac{1}{2} \int_{\tau_i} (\text{curl}\vec{\varphi} \cdot \vec{\mathbf{H}}_i + \text{curl}\vec{\mathbf{H}}_i \cdot \vec{\varphi}) \\ \qquad\qquad\qquad\qquad - \dfrac{1}{2} \displaystyle\sum_{k \in \mathcal{V}_i} \int_{a_{ik}} \vec{\varphi} \cdot (\vec{\mathbf{H}}_k \times \vec{n}_{ik}), \\ \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \mu_i \partial_t \vec{\mathbf{H}}_i = -\frac{1}{2} \int_{\tau_i} (\text{curl}\vec{\varphi} \cdot \vec{\mathbf{E}}_i + \text{curl}\vec{\mathbf{E}}_i \cdot \vec{\varphi}) \\ \qquad\qquad\qquad\qquad + \dfrac{1}{2} \displaystyle\sum_{k \in \mathcal{V}_i} \int_{a_{ik}} \vec{\varphi} \cdot (\vec{\mathbf{E}}_k \times \vec{n}_{ik}). \end{array} \right. \tag{4}$$

Eq. (4) can be rewritten in terms of scalar unknowns. Inside each element, the fields are re-composed according to $\vec{\mathbf{E}}_i = \displaystyle\sum_{1 \leq j \leq d} E_{ij} \vec{\varphi}_{ij}$ and $\vec{\mathbf{H}}_i = \displaystyle\sum_{1 \leq j \leq d} H_{ij} \vec{\varphi}_{ij}$ and let us now denote by $\mathbf{E}_i$ and $\mathbf{H}_i$ respectively

the column vectors $(E_{il})_{1\leq l\leq d_i}$ and $(H_{il})_{1\leq l\leq d_i}$. Then, (4) is equivalent to:

$$\left\{\begin{array}{rcl} M_i^{\epsilon}\dfrac{d\mathbf{E}_i}{dt} & = & K_i\mathbf{H}_i - \displaystyle\sum_{k\in\mathcal{V}_i} S_{ik}\mathbf{H}_k, \\ M_i^{\mu}\dfrac{d\mathbf{H}_i}{dt} & = & -K_i\mathbf{E}_i + \displaystyle\sum_{k\in\mathcal{V}_i} S_{ik}\mathbf{E}_k, \end{array}\right. \tag{5}$$

where the symmetric positive definite mass matrices M_i^{η} (η stands for ϵ or μ), the symmetric stiffness matrix K_i (both of size $d_i \times d_i$) and the symmetric interface matrix S_{ik} (of size $d_i \times d_k$) are given by:

$$\begin{array}{rcl} (M_i^{\eta})_{jl} & = & \eta_i \displaystyle\int_{\tau_i} {}^t\vec{\varphi}_{ij}\cdot\vec{\varphi}_{il}, \\ (K_i)_{jl} & = & \dfrac{1}{2}\displaystyle\int_{\tau_i} {}^t\vec{\varphi}_{ij}\cdot\mathrm{curl}\vec{\varphi}_{il} + {}^t\vec{\varphi}_{il}\cdot\mathrm{curl}\vec{\varphi}_{ij}, \\ (S_{ik})_{jl} & = & \dfrac{1}{2}\displaystyle\int_{a_{ik}} {}^t\vec{\varphi}_{ij}\cdot(\vec{\varphi}_{kl}\times\vec{n}_{ik}). \end{array}$$

3.2 Time discretization

The set of local system of ordinary differential equations for each τ_i (5) can be formally transformed in a global system. To this end, we suppose that all electric (resp. magnetic) unknowns are gathered in a column vector $\mathbb{E}$ (resp. $\mathbb{H}$) of size $d_g = \displaystyle\sum_{i=1}^{N_t} d_i$ where N_t stands for the number of elements in $\mathcal{T}_h$. Then system (5) can be rewritten as:

$$\left\{\begin{array}{rcl} \mathbb{M}^{\epsilon}\dfrac{d\mathbb{E}}{dt} & = & \mathbb{K}\mathbb{H} - \mathbb{A}\mathbb{H} - \mathbb{B}\mathbb{H} + \mathbb{C}_E\mathbb{E}, \\ \mathbb{M}^{\mu}\dfrac{d\mathbb{H}}{dt} & = & -\mathbb{K}\mathbb{E} + \mathbb{A}\mathbb{E} - \mathbb{B}\mathbb{E} + \mathbb{C}_H\mathbb{H}, \end{array}\right. \tag{6}$$

where we have the following definitions and properties:

- $\mathbb{M}^{\epsilon}, \mathbb{M}^{\mu}$ and $\mathbb{K}$ are $d_g \times d_g$ block diagonal matrices with diagonal blocks equal to $M_i^{\epsilon}, M_i^{\mu}$ and K_i respectively. $\mathbb{M}^{\epsilon}$ and $\mathbb{M}^{\mu}$ are symmetric positive definite matrices, and $\mathbb{K}$ is a symmetric matrix.
- $\mathbb{A}$ is also a $d_g \times d_g$ block sparse matrix, whose non-zero blocks are equal to S_{ik} when $a_{ik} \in \mathcal{F}_h^I$. Since $\vec{n}_{ki} = -\vec{n}_{ik}$, it can be checked that $(S_{ik})_{jl} = (S_{ki})_{lj}$ and then $S_{ki} = {}^tS_{ik}$; thus $\mathbb{A}$ is a symmetric matrix.
- $\mathbb{B}$ is a $d_g \times d_g$ block diagonal matrix, whose non-zero blocks are equal to S_{ik} when $a_{ik} \in \mathcal{F}_h^B \cap \Gamma_m$. In that case, $(S_{ik})_{jl} = -(S_{ik})_{lj}$; thus $\mathbb{B}$ is a skew-symmetric matrix.
- $\mathbb{C}_E$ and $\mathbb{C}_H$ are $d_g \times d_g$ block diagonal matrices associated to boundary integral terms for $a_{ik} \in \mathcal{F}_h^B \cap \Gamma_a$.

Consequently, if we set $\mathbb{S} = \mathbb{K} - \mathbb{A} - \mathbb{B}$, the system (6) rewrites as:

$$\mathbb{M}^{\epsilon}\frac{d\mathbb{E}}{dt} = \mathbb{S}\mathbb{H} + \mathbb{C}_E\mathbb{E} \quad , \quad \mathbb{M}^{\mu}\frac{d\mathbb{H}}{dt} = -\,{}^t\mathbb{S}\mathbb{E} + \mathbb{C}_H\mathbb{H}. \tag{7}$$

The semi-discrete system (7) is time integrated using a second-order leap-frog scheme as:

$$\left\{\begin{array}{rcl} \mathbb{M}^{\epsilon}\left(\dfrac{\mathbb{E}^{n+1}-\mathbb{E}^{n}}{\Delta t}\right) & = & \mathbb{S}\mathbb{H}^{n+\frac{1}{2}} + \mathbb{C}_E\mathbb{E}^{n}, \\ \mathbb{M}^{\mu}\left(\dfrac{\mathbb{H}^{n+\frac{3}{2}}-\mathbb{H}^{n+\frac{1}{2}}}{\Delta t}\right) & = & -\,{}^t\mathbb{S}\mathbb{E}^{n+1} + \mathbb{C}_H\mathbb{H}^{n+\frac{1}{2}}. \end{array}\right. \tag{8}$$

The resulting fully explicit DGTD-$\mathbb{P}_{p_i}$ method is analyzed in [9] where it is shown that, when $\Gamma_a = \emptyset$, the method is non-dissipative, conserves a discrete form of the electromagnetic energy and is stable under the CFL-like condition:

$$\Delta t \leq \frac{2}{\alpha}, \quad \text{with} \quad \alpha = \parallel (\mathbb{M}^{-\mu})^{\frac{1}{2}}\ {}^t\mathbb{S}\ (\mathbb{M}^{-\epsilon})^{\frac{1}{2}} \parallel, \tag{9}$$

where $\|.\|$ denotes the canonical matrix norm and the matrix $(\mathbb{M}^{-\eta})^{\frac{1}{2}}$ is the inverse square root of $\mathbb{M}^{\eta}$.

3.3 Numerical treatment of biological propagation media

Human tissues are dispersive materials and thus require a specific treatment when modeling time-domain problems. However, in this study, we do not take into aacount the dispersive characteristic of the propagation media and simply consider them as conductive as it is done in the great majority of numerical dosumetry studies. Then, a conductive current $\vec{J_\sigma} = \sigma\vec{E}$ has to be taken into account in (1). It is straightforward to verify that the space discretization of this current term in the framework of the discontinuous Galerkin formulation described in section 3.1 leads to the introduction of the term $-M_i^{\epsilon}\mathbf{E}_i$ in the right hand side of the first equation of (5). Then this term is time integrated as $-M_i^{\epsilon}\left(\mathbf{E}_i^{n} + \mathbf{E}_i^{n+1}\right)/2$ meaning that both the left and right hands side terms of (8) are affected by the discretization of this conductive current.

4 Implementation on GPU clusters

We discuss in this section the design principles that we have adopted for the implementation of the DGTD-$\mathbb{P}_{p_i}$ method described in section 3. Prior to do so, we briefly review the main aspects of GPU computing as well as basic features of the CUDA programming model that are used afterward.

4.1 Generalities about GPUs

Many-core GPUs are currently capable of delivering over 1 TFlops of single precision performance versus 100 GFlops for the multi-core CPUs. This performance gap is mainly due to the architectural differences between the two types of processors (see Fig. 1). CPU cores continue to be optimized for single threaded performance at the expense of parallel execution. Large CPU cache memories are provided to reduce

the instruction and data access latencies of large complex applications; but as additional thread contexts share the limited resources of cache capacity and external memory bandwidth, throughput and power efficiency decrease. In contrast, the design philosophy of the GPUs is motivated by the ability to optimize for the execution throughput of massive numbers of threads. The hardware takes advantage of a large number of execution threads to find work to do when some of them are waiting for long latency memory accesses, thus minimizing the control logic required for each execution thread. Small cache memories are provided to help control the bandwidth requirements of these applications so multiple threads that access the same memory data do not need to all go to the dynamic random access memory (DRAM).

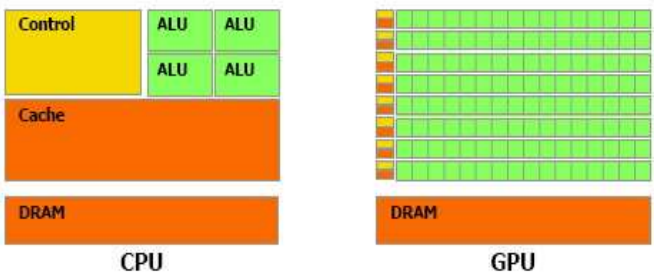


Figure 1: GPU vs CPU design philosophies.

4.2 CUDA programming model

CUDA (Compute Unified Device Architecture) [20] is a parallel computing architecture developed by Nvidia, which enables non-graphics computations on GPUs. The typical architecture of GT200 series GPUs which is considered in this study is built around a scalable array of multithreaded Streaming Multiprocessors (SMs), which consist of eight Scalar Processor (SP) cores, one multithreaded Instruction Unit (MT), two Special Function Units (SFU) performing floating point functions as well as transcendental functions, one Double Precision Unit (DP), 16 KB 32-bit on-chip registers and 16 KB on-chip shared memory (see Fig. 2). The global, constant, texture and local memory are optimized for different memory usages and reside on the off-chip device memory, accessible by all SPs. A GPU such as the one in the Tesla C1060 system has 240 cores, processes 933 Gflops in single precision floating point arithmetic, and can run thousands of threads per application. However dramatic increases in computing performance can only be achieved after extensive optimization and tuning, by rewriting processing intensive parts with parallelization techniques. To a CUDA programmer, the computing system consists of a host, which is a traditional central processing unit (CPU), and one or more devices, viewed as massively parallel processors equipped with a large number of arithmetic execution units. A CUDA program is a unified source code encompassing both host and device code. The data parallel functions, called kernels, typically generate a large number of GPU threads to exploit data parallelism in a Single Instruction Multiple Data (SIMD) fashion, because all of these threads execute the same code during a parallel phase. Launching a kernel for GPU execution is similar to calling the kernel function, except that the programmer needs to

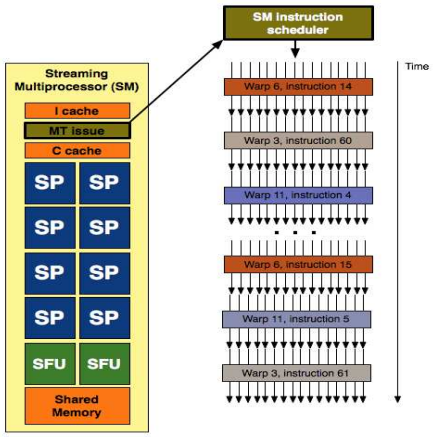


Figure 2: Streaming Multiprocessor architecture and scheduler.

specify the space of GPU threads that execute it, called a grid. A grid contains multiple thread blocks having the same number of threads, organized in a two-dimensional space, executing the same kernel. Each thread block is, in turn, organized as a three-dimensional array of threads with a total size up to 512 threads. A unique thread ID is assigned to each thread, computed from both thread indexes, allowing all threads in a grid to distinguish themselves from each other and to identify the appropriate portion of the data to process. Threads in the same thread block may coordinate their activities by using shared memory, and by synchronizing at a barrier using a specific intrinsic function. The programmer needs to allocate memory on the device and transfer pertinent data from the host memory to the allocated device memory. Similarly, after device execution, the programmer needs to transfer result data from the device memory back to the host memory and free up the device memory that is no longer needed.

Simple CUDA kernels generally achieve only a small fraction of the potential speed of the underlying hardware due to the fact that global memory, which is typically implemented with dynamic random access memory (DRAM) of a GPU, tends to have long access latencies (hundred of clock cycles) and finite access bandwidth. Although having many threads available for execution can theoretically tolerate long memory access latencies, one can easily run into a situation where traffic congestion in the global memory access paths prevents all but a few threads from making progress, thus rendering some of the streaming multiprocessors idle. In order to circumvent such congestion, CUDA provides a number of additional methods for accessing memory that can remove the majority of data requests to the global memory (see [20] for more details). The constant memory supports short latency, high bandwidth and read only access by the device. The texture memory supports read only access by the device and offers different addressing modes, as well as data filtering, for some specific data formats. The constant and texture memory are also accessible from the host. Registers and shared memory are on-chip memories. Variables that reside in these types of memory can be accessed at very high speed in a highly parallel manner. Registers are allocated to individual threads; each thread can read and/or write its own registers very fast with almost no delay. A kernel function typically uses registers to hold frequently accessed

variables that are private to each thread. Shared memory is allocated to thread blocks; all threads in a block can access variables in the shared memory locations allocated to the block but the shared memory cannot be shared between other blocks.

4.3 CUDA adaptation of the DGTD method

We first note that the main computational kernels of the DGTD-$\mathbb{P}_{p_i}$ method considered in this study are the volume and surface integrals involved in the weak formulation (4). Moreover, we limit ourselves to a uniform order method i.e. $p \equiv p_i$ is the same for all the elements of the mesh, and we present experimental results for the values $p = 1, 2, 3, 4$. At the discrete level, these local computations translate into the matrix-vector products appearing in (5). The discrete equations for updating the electric and magnetic fields are composed of the same steps and only differ by the fields they are applied to. They both involve the same kernels that we will refer to in the sequel as `intVolume` (computation of volume integrals), `intSurface` (computation of surface integrals) and `updateField` (update of field components).

4.3.1 Common paradigm and definitions

As advised in the section 5.3 of the CUDA programming guide [20], all our kernels follow the following paradigm:

1. Load data from device memory to shared memory
2. Synchronize with all the other threads of the block so that each thread can safely read shared memory locations that were populated by different threads
3. Process the data in shared memory
4. Synchronize again to make sure that shared memory has been updated with the results
5. Write the results back to device memory

In our implementation, some useful elementary matrices, such as the mass matrix computed on the reference element, are stored in constant memory because they are small and are accessed following constant memory patterns. For the sequel, we introduce the following notations:

NBTET is the number of tetrahedra that are treated by a block of threads. It depends of the chosen interpolation order and it is taken to be a multiple of 16 because of the way one load and write data to and from device memory.

NDL is the number of degrees of freedom (d.o.f) in an element τ_i for each field component, for a given interpolation order.

NDF is the number of d.o.f on a face a_{ik} for each field component, for a given interpolation order.

4.3.2 Volume integral kernel : `intVolume`

This kernel operates on each d.o.f of a tetrahedron. Since the number of d.o.f increases with the interpolation order, resources needed by this kernel (registers and shared memory) also raise. Consequently, we wrote two versions of this kernel: one kernel for $p = 1$ and 2, and the other one for $p = 3$ and 4. However, these two versions have some common features. First, each thread computes one d.o.f of one tetrahedron. The second common feature is the data stored in shared memory, which are some geometrical quantities associated to a tetrahedron, and the field and the flux balance components. The last common feature is the number of tetrahedra operated by a block (i.e.NBTET). The main difference is that when in the low order version a block computes all the d.o.f (NDL) of the NBTET tetrahedra, the high order volume kernel only computes a certain number of d.o.f of the NBTET tetrahedra. Consequently, in the latter case, two or three instances of the kernel are necessary to compute all the d.o.f of all the tetrahedra. This approach induces a drawback because we have to load field data in two or three kernels instead of one. Indeed, the dimension of a block is NBTET*NDL which leads to blocks of more than 512 threads for high interpolation orders which is not possible in CUDA. However, there is also a benefit because computing a lower number of d.o.f in a kernel allows us to use less shared memory in the buffer storing field data and less registers in a kernel. Consequently, the occupation of the graphic card increases.

4.3.3 Surface Integral kernel : `intSurface`

For this kernel, one thread works on one surface d.o.f of one tetrahedron. Similarly to the `intVolume` kernel, two versions of this kernel have been implemented. For the low order version, a thread will apply the influence of its d.o.f to the four faces of its tetrahedron whereas for the high order version of this kernel, a thread will only work on one face of its tetrahedron. So, for the low order version, a block will compute four faces of NBTET tetrahedra instead of one face of NBTET tetrahedra for the high order version. Therefore, the high order version needs to launch four kernels instead of one for the low order version. Here, we work on the surface d.o.f (NDF) but fields components are store using the volume d.o.f (NDL) so we need to use a permutation matrix to link these different local numberings of these d.o.f. Moreover, a face of a tetrahedron is also shared by another tetrahedron and the corresponding field values are needed in the computation of the elementary flux. Consequently, we cannot load field data in a coalesced way and we have to use texture memory. Field values are loaded before each face computation. Nevertheless, the high order version has a memory drawback compared to the lower one. Indeed, because there are four launches of the function (one for each face), data are written four times to the flux table instead of one in the low order version.

4.3.4 Update kernel : `updateField`

There are four update kernels. First of all, update kernels are a bit different according to the field they are working on (electric or magnetic). Since in this case a thread works on one d.o.f of a tetrahedron, the dimension of a block is NBTET*NDL. Consequently, as for the `intVolume` kernel, we need a special version for the higher interpolation orders in order to avoid exceeding the maximum number of threads per block. In the high order version, we adopt an approach where a thread deals with two different d.o.f of a tetrahedron which allows a block to compute all the d.o.f for NBTET tetrahedra. This approach is

less efficient for the lower interpolation orders. The two versions of the electric field update kernels need only one shared memory table. Indeed, in the first step, the flux computed by the previous kernels is loaded in this table, used to do some computations and then stored in a register. Therefore, the shared memory table is no longer used at the end of this part. In the second step, we load the previous values of the electric field in it in a coalesced way. In a third step, we update the value of the field in the shared memory, and in the last step, we write the new value of the field in the global memory.

The update of the magnetic field follows essentially the same pattern as the update of the electric field, except for the computation of the discrete energy. Indeed, to compute the discrete energy, values of updated electric field and magnetic field are needed. Thus, it is reasonable to compute the energy simultaneously with the update of the magnetic field. However, this works fine only for the lower interpolation orders because this solution needs another shared memory table to store electric field values and some additional registers. For the high order version, there are two kernels, one that only performs the update of the magnetic field and one that computes the energy. When we want to compute the energy, the kernel updating the magnetic field writes the new value of the field in the flux table instead of the magnetic field table. This allows us to have the magnetic field value of the current (in the flux table) and of the previous time step (in the magnetic field table) and when the energy is computed, the energy kernel updates the magnetic field table. Finally, we use the reduction algorithm presented in the CUDA SDK 3.0 to computes energy.

5 Numerical and performance results

5.1 Tetrahedral mesh based geometric models of head tissues

The DGTD-$\mathbb{P}_{p_i}$ method described previously assumes that the computational domain is discretized using tetrahedral elements. In this study, we aim at exploiting this numerical method for the calculation of the SAR induced in head tissues. A first step is thus to construct compatible geometrical models of the head tissues. Starting from magnetic resonance images of the Visible Human 2.0 project [21], head tissues are segmented and surface triangulations of a selected number of tissues are obtained. Different strategies can be used in order to obtain a smooth and accurate segmentation of head tissues and interface triangulations as well. The strategy adopted in this work consists in using a variant of Chew's algorithm [4], based on Delaunay triangulation restricted to the interface, which allows to control the size and aspect ratio of interface triangles [2]. Example of triangulations of the skin, skull and brain are shown on Fig. 3. In a second step, these triangulated surfaces together with a triangulation of the artificial boundary (absorbing boundary) of the overall computational domain are used as inputs for the generation of volume meshes. The GHS3D tetrahedral mesh generator [11] is used to mesh the volume domains between the various interfaces. The exterior of the head must also be meshed, up to a certain distance and the computational domain is artificially bounded by a sphere surface corresponding to the boundary Γ_a on which the Silver-Müller absorbing boundary condition is imposed. Moreover, a simplified mobile phone model (metallic box with a quarter-wave length mounted on the top surface) is included and placed in vertical position close to the right ear (see Fig. 4). The surface of this metallic box defines the boundary Γ_m. Overall, the geometrical models considered here consist of four tissues (skin, skull, CSF - Cerebro Spinal Fluid and brain).

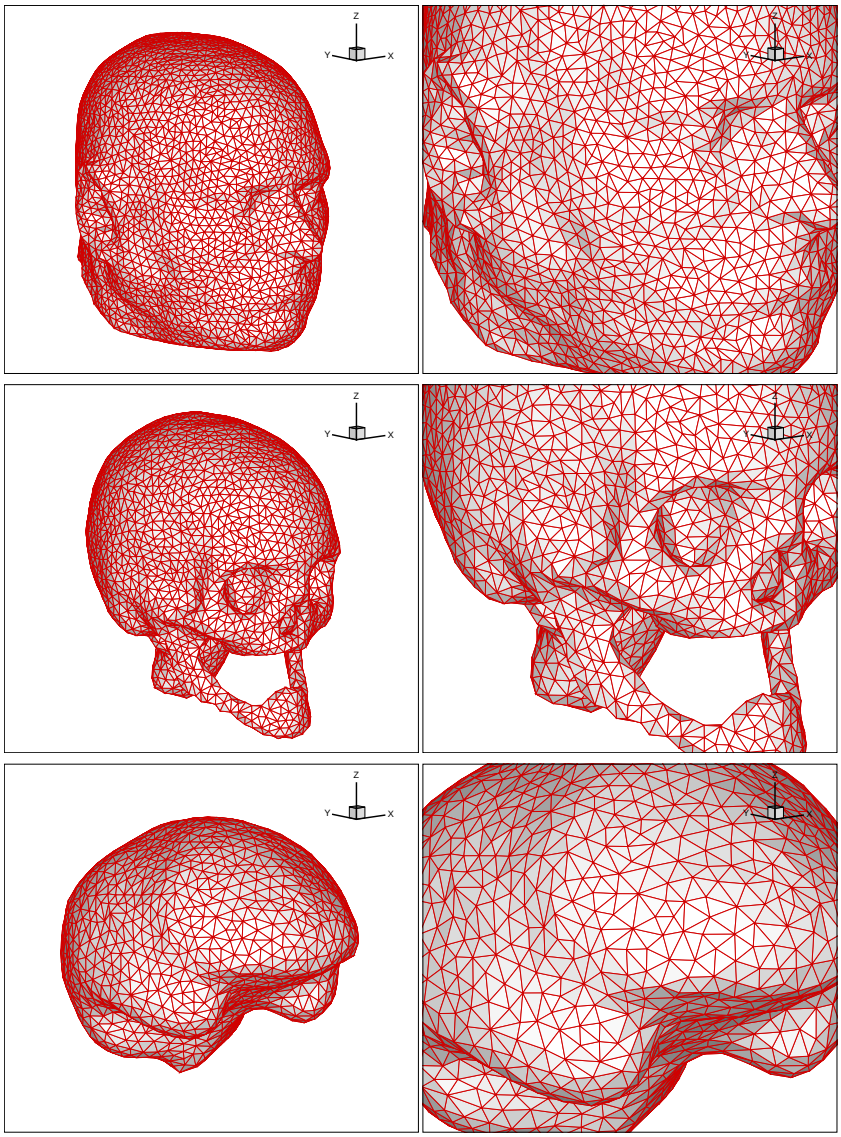


Figure 3: Surface meshes of the skin, skll and brain.

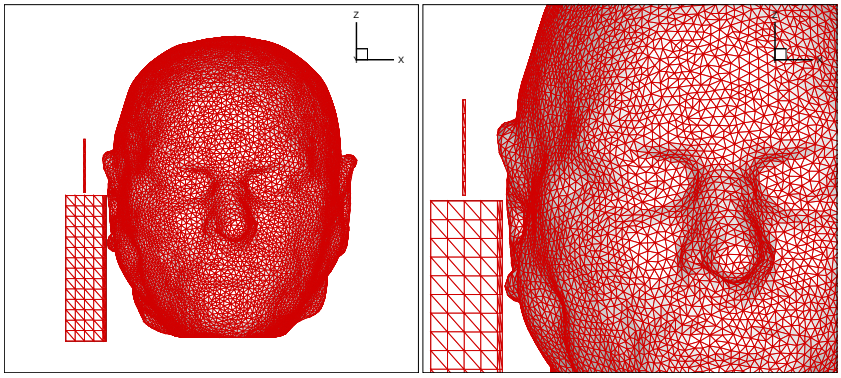


Figure 4: Simplified mobile phone model and its positioning.

5.2 Numerical results

All the numerical experiments reported here are concerned with the propagation of an electromagnetic wave emitted by a dipolar source localized (and centered) between the lower tip of the antenna and the top surface of the metallic box defining the mobile phone, and is modeled by a current of the form ($\vec{x}_d$ is the localization point of the source):

$$\vec{J}_s(\vec{x},t) = \delta(\vec{x}-\vec{x}_d)f(t)\vec{e}_z, \tag{10}$$

where $f(t)$ is a sinusoidally varying temporal signal. This source current is easily introduced and discretized according to the discontinuous Galerkin formulation discussed in subsection 3.1. The physical simulation time has been fixed to 5 periods of the temporal signal of (10). A discrete Fourier transform of the components of the electric field is computed during the last period of the simulation. The characteristics of the tissues are summarized in Table 1 where the values of the electrical permittivity correspond to a frequency F=1800 MHz. We consider a sequence of three unstructured tetrahedral meshes whose

Table 1: Exposure of head tissues to a localized source radiation: electromagnetic characteristics of tissues.

Tissue	ε_r	λ (mm)	σ (S/m)	ρ (Kg/m^3)
Skin	43.85	26.73	1.23	1100.0
Skull	15.56	42.25	0.43	1200.0
CSF	67.20	20.33	2.92	1000.0
Brain	43.55	25.26	1.15	1050.0

characteristics are summarized in Table 2. For these meshes, the artificial boundary Γ_a is a spherical surface approximately located one wavelength away from the skin. The tetrahedral meshes are globally non-uniform and the quantities L_{min}, L_{max} and L_{avg} in Table 2 respectively denote the minimum,

maximum and average lengths of mesh edges. Fig. 5 shows the contour lines of the amplitude of the

Table 2: Characteristics of the fully unstructured tetrahedral meshes of head tissues.

Mesh	# elements	L_{min} (mm)	L_{max} (mm)	L_{avg} (mm)
M1	815,405	1.00	28.14	10.69
M2	1,862,136	0.65	23.81	6.89
M3	7,894,172	0.77	22.75	3.21

electric field on the skin, skull and brain surfaces for a numerical solution computed with mesh M2 using the DGTD-$\mathbb{P}_2$ method. Contour lines of the local SAR normalized to the maximum value of the local SAR and of the local SAR normalized to the total emitted power, plotted in selected XoZ and XoY planes, are shown on Fig. 6 and Fig. 7 for calculations based on the coarsest mesh (i.e.mesh M1), and on Fig. 8 and Fig. 9 for calculations based on the finest mesh (i.e.mesh M3). In order to discuss these results, we consider that the numerical solution computed with mesh M3 using the DGTD-$\mathbb{P}_1$ method defines a reference solution. Patterns of the contour lines for calculations respectively performed with mesh M1 using the DGTD-$\mathbb{P}_3$ method and with mesh M3 using the DGTD-$\mathbb{P}_1$ method are very similar. However variations certainly exist locally since the ranges of the plotted values differ (especially for the local SAR normalized to the total emitted power). In Table 3, the quantity given parenthetically is the difference with the reference value (i.e., the one associated to mesh M3 and the DGTD-$\mathbb{P}_1$ method) and the corresponding error level. We note here that the relative error tends to increase when switching from $p = 1$ to $p = 2$ for a given discretization mesh. Again, this should not be interpreted as a counter effect of an increase of the approximation order since this relative error is evaluated on the basis of a solution computed on the finest mesh which is constructed from high resolution triangulations of the tissue interfaces (see the figures in section 5.1). Rather, we can conclude that the discretization of the geometrical features of tissues has a greater impact on accuracy than the interpolation order in the DGTD-$\mathbb{P}_p$ method (first column of Table 3).

Table 3: Calculations with the DGTD-$\mathbb{P}_p$ method. Maximum value of the normalized local SAR.

Mesh	Method
-	DGTD-$\mathbb{P}_1$
M1	3.365 W/Kg (0.463, 12.1 %)
M2	3.734 W/Kg (0.094, 2.4 %)
M3	3.828 W/Kg (reference value)
-	DGTD-$\mathbb{P}_2$
M1	3.269 W/Kg (0.559, 14.6 %)
M2	3.586 W/Kg (0.242, 6.3 %)
-	DGTD-$\mathbb{P}_3$
M1	3.283 W/Kg (0.545, 12.3 %)

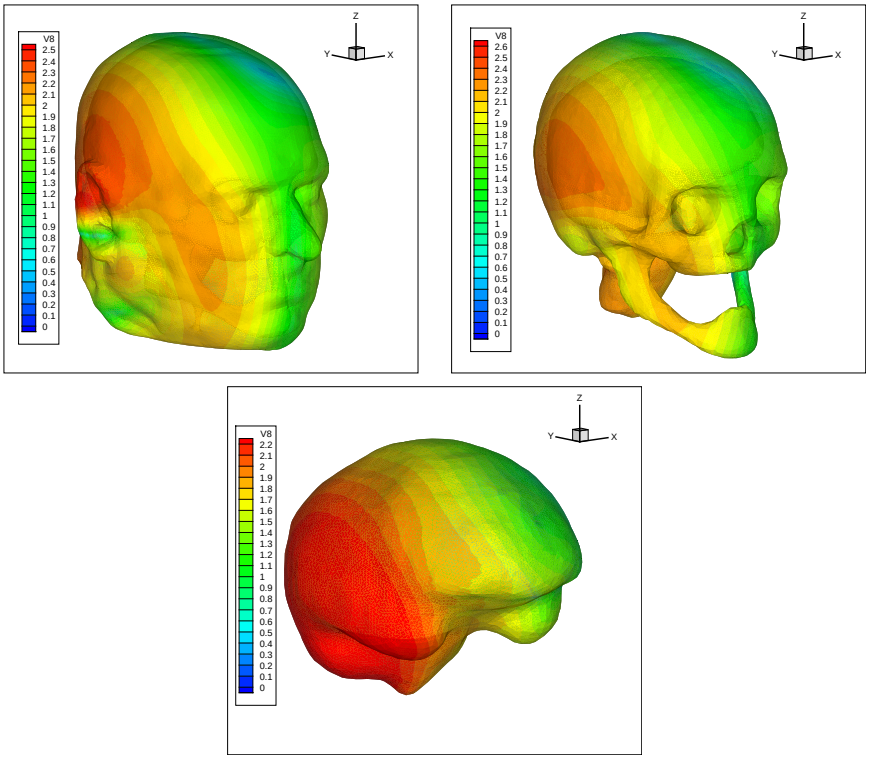


Figure 5: Calculations with the DGTD-$\mathbb{P}_2$ method. Mesh M2: contour lines of the amplitude of the electric field in log scale.

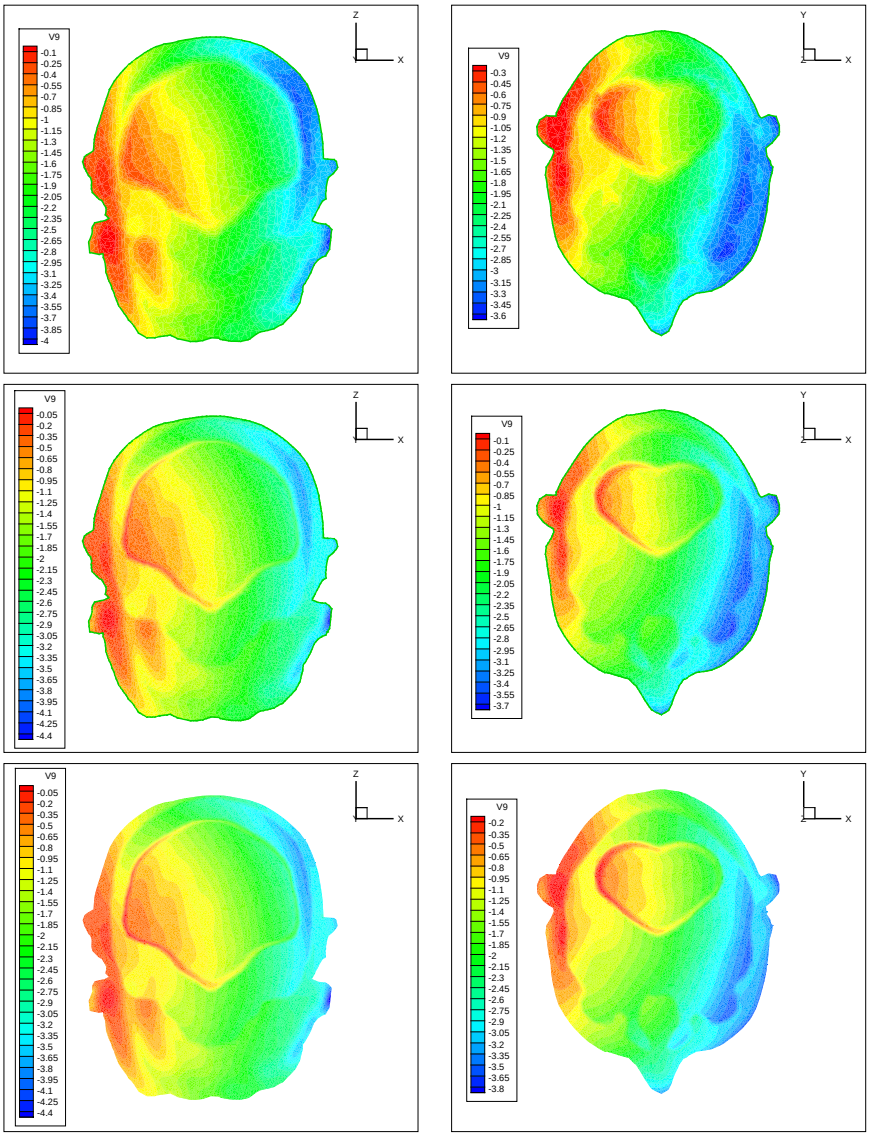


Figure 6: Calculations with the DGTD-$\mathbb{P}_1$ (top), DGTD-$\mathbb{P}_2$ (middle) and DGTD-$\mathbb{P}_3$ (bottom) methods. Mesh M1: contour lines of local SAR over maximum local SAR in log scale (selected cut planes).

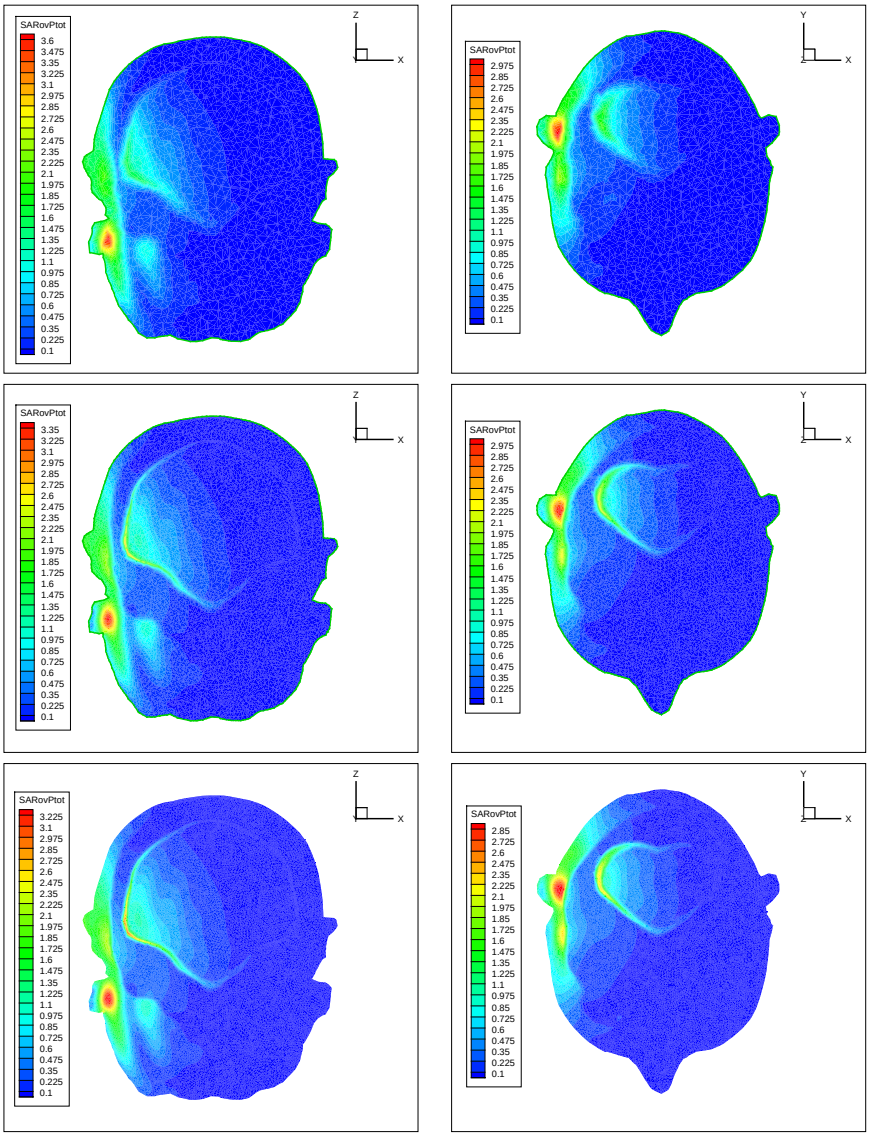


Figure 7: Calculations with the DGTD-$\mathbb{P}_1$ (top), DGTD-$\mathbb{P}_2$ (middle) and DGTD-$\mathbb{P}_3$ (bottom) methods. Mesh M1: contour lines of normalized local SAR (selected cut planes).

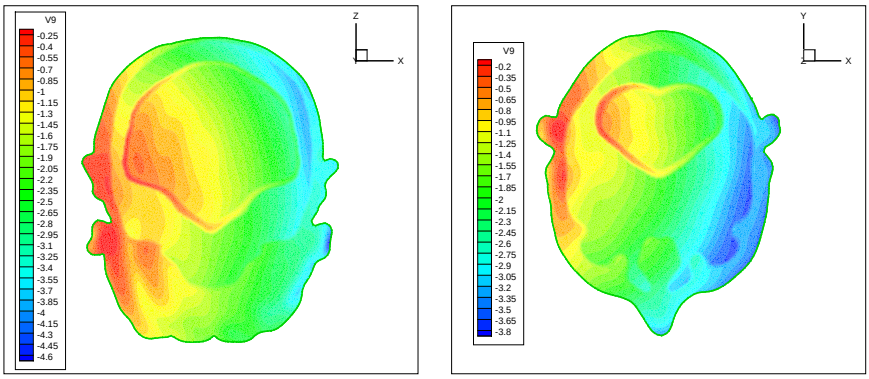


Figure 8: Calculations with the DGTD-$\mathbb{P}_1$ method. Mesh M3: contour lines of local SAR over maximum local SAR in log scale (selected cut planes).

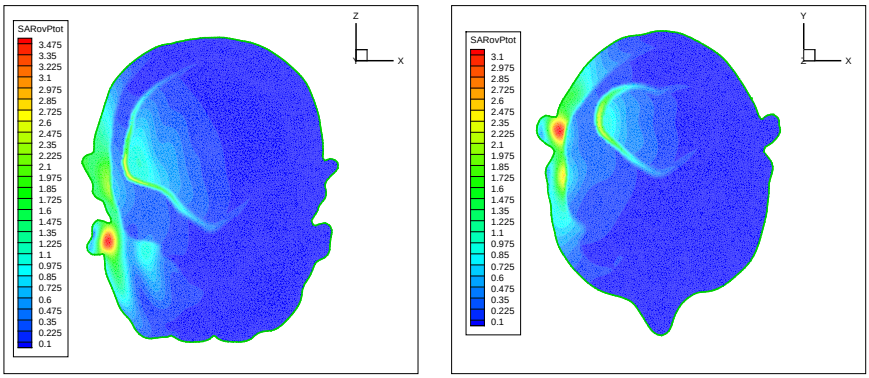


Figure 9: Calculations with the DGTD-$\mathbb{P}_1$ method. Mesh M3: contour lines of normalized local SAR (selected cut planes).

5.3 Performance results

5.3.1 Hardware configuration

Numerical simulations have been performed on the hybrid CPU-GPU cluster of the CCRT (Centre de Calcul Recherche et Technologie) in Bruyères-le-Châtel, France. This cluster comprises 1068 Intel CPU nodes and 48 Tesla S1070 GPU systems. Each Tesla S1070 has four GT200 GPUs and two PCI Express-2 buses (i.e., two GPUs share a PCI Express-2 bus). The GT200 cards have 4 GB of memory, and the memory bandwidth is 102 GB/s. The Tesla systems are connected to BULL Novascale R422 E1 nodes with two quad-core Intel Xeon X5570 Nehalem processors operating at 2.93 GHz. Each node has 24 GB of RAM. The network is a non-blocking, symmetric, full duplex Voltaire InfiniBand double data rate organized as a fat tree.

5.3.2 Weak scalability

We first present results for the assessment of the weak scalability properties of the GPU enabled DGTD-$\mathbb{P}_p$ method. For that purpose, we consider a model test problem which consists in the propagation of a standing wave in a perfectly conducting unitary cubic cavity. For this simple geometry, we make use of regular uniform tetrahedral meshes respectively containing 3,072,000 elements for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods, 1,296,000 elements for the DGTD-$\mathbb{P}_3$ method and 750,000 elements for the DGTD-$\mathbb{P}_4$ method for the experiments involving one GPU. As usual in the context of a weak scalability analysis, the size of each mesh is increased proportionally to the number of computational entities. Moreover, since these meshes are regular discretizations of the cube, it is possible to construct perfectly balanced partitions and this is achieved here by constructing the tetrahedral meshes in parallel (i.e.on a subdomain basis) given a box-wise decomposition of the domain. Timings are given for 1000 iterations of the time stepping loop. We also emphasize that GPU timings (for all the performance results presented here and in the following subsections) include the data structures copy operations from the CPU memory to the GPU device memory prior to the time stepping loop, and vice versa at the end of the time stepping loop. The graphs of Fig. 10 illustrate the almost perfect weak scalability of the GPU enabled DGTD-$\mathbb{P}_p$ method with $p = 1, \ldots, 4$ for up to 128 GPUs. GFlops rates are plotted on the graphs of Fig. 11 while Table 4 summarizes the measured GFlops rates for 1 and 128 GPUs. We note that the DGTD-$\mathbb{P}_3$ method delivers the best performances and that our current implementation strategy for interpolation order 4 does not allow higher throughput as demonstrated in [15] (Fig. 8(a) page 7877). Besides, the lower Gflops rates observed here as compared to those reported in [15] are probably due to the use of a centered numerical flux function for the computation of the boundary integral terms in (3) which offers a lower computation to communication ratio than the one characterizing the upwind numerical flux function adopted in [15].

Table 4: Weak scalability assessment: sustained performance figures.

# GPU	DGTD-$\mathbb{P}_1$	DGTD-$\mathbb{P}_2$	DGTD-$\mathbb{P}_3$	DGTD-$\mathbb{P}_4$
1	63 GFlops	92 GFlops	106 GFlops	94 GFlops
128	8072 GFlops	11844 GFlops	13676 GFlops	12009 GFlops

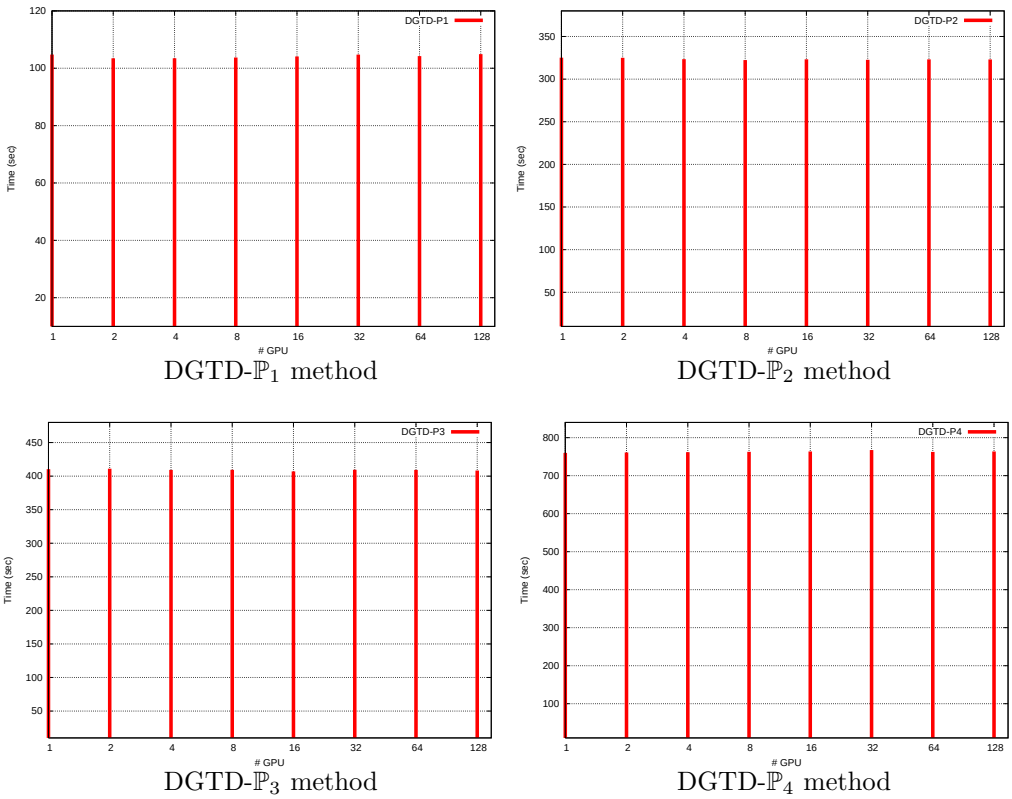
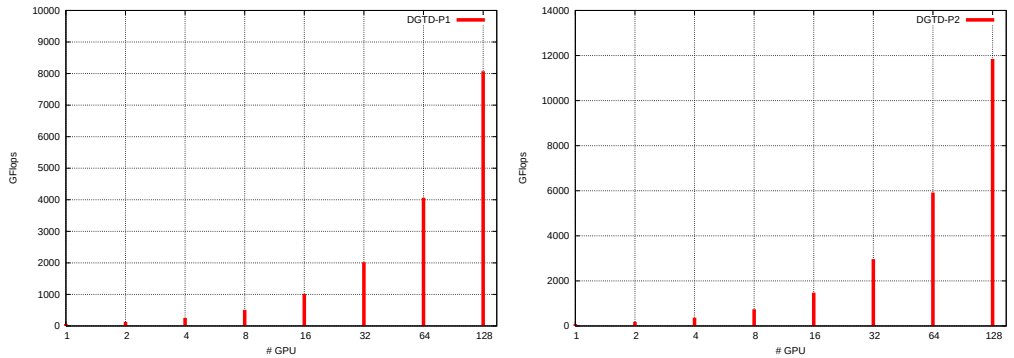


Figure 10: Weak scalability assessment: evolution of the computing time with the number of GPUs.

5.3.3 Strong scalability

We now come back to the wave propagation problems discussed in subsection 5.2 and present strong scalability results for simulations with the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods using the teatrahedral meshes of Table 2. Performances results are presented in Tables 5 to 7. For the coarsest mesh (i.e. mesh M1), the parallel speedup is evaluated for 16 GPUs relatively to the simulation time using one GPU. Although the number of elements of this mesh is well below the size of the mesh considered for the weak scalability analysis (i.e. 3,072,000 elements for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods), superlinear speedups are obtained. However, not surprisingly, the single GPU GFlops rates are lower than the corresponding ones reported in Table 4 (32 instead of 63 for the DGTD-$\mathbb{P}_1$ method, and 60 instead of 92 for the DGTD-$\mathbb{P}_2$ method). For the two other meshes (i.e. M2 and M3), as expected the DGTD-$\mathbb{P}_2$ method is always more scalable than the DGTD-$\mathbb{P}_1$ method because of a more favorable computation to communication ratio. We can reasonably predict that the strong scalability figures will be better



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

DGTD-$\mathbb{P}_3$ method DGTD-$\mathbb{P}_4$ method

Figure 11: Weak scalability assessment: evolution of the GFlops figure with the number of GPUs.

for higher order methods (i.e. for $p \geq 3$) however at the expense of a noticeable increase in the overall simulation time due to a reduced time step to insure stability of the time stepping process.

Table 5: Head tissues exposure to an electromagnetic wave emitted by a mobile phone. Strong scalability assessment: mesh M1. Elapsed time on 16 CPUs: 715 sec (DGTD-$\mathbb{P}_1$ method) and 3824 sec (DGTD-$\mathbb{P}_2$ method).

# GPU	DGTD-$\mathbb{P}_1$			DGTD-$\mathbb{P}_2$		
	Time	GFlops	Speedup	Time	GFlops	Speedup
1	620 sec	32	-	2683 sec	60	-
16	35 sec	566	17.8	145 sec	1110	18.5

Table 6: Head tissues exposure to an electromagnetic wave emitted by a mobile phone. Strong scalability assessment: mesh M2. Elapsed time on 64 CPUs: 519 sec (DGTD-$\mathbb{P}_1$ method) and 2869 sec (DGTD-$\mathbb{P}_2$ method).

# GPU	DGTD-$\mathbb{P}_1$			DGTD-$\mathbb{P}_2$		
	Time	GFlops	Speedup	Time	GFlops	Speedup
16	82 sec	699	-	407 sec	1137	-
32	46 sec	1239	1.8	201 sec	2299	2.0
64	33 sec	1747	2.5	116 sec	4007	3.5

Table 7: Head tissues exposure to an electromagnetic wave emitted by a mobile phone. Strong scalability assessment: mesh M3. Elapsed time on 64 CPUs: 2786 sec (DGTD-$\mathbb{P}_1$ method) and 6057 sec (DGTD-$\mathbb{P}_2$ method).

# GPU	DGTD-$\mathbb{P}_1$			DGTD-$\mathbb{P}_2$		
	Time	GFlops	Speedup	Time	GFlops	Speedup
32	162 sec	146	-	816 sec	2370	-
64	97 sec	2470	1.7	416 sec	4657	2.0
128	69 sec	3469	2.4	257 sec	7522	3.2

6 Conclusion

In this paper we have presented a high performance numerical methodology to simulate electromagnetic wave interaction with biological tissues. This methodology combines a high order discontinuous Galerkin

time domain (DGTD) method for solving the system of Maxwell equations with realistic geometrical models of human organs based on unstructured tetrahedral meshes. It has been applied here to the numerical evaluation of the SAR induced in head tissues when the latter are exposed to an electromagnetic wave emitted by a mobile phone antenna. The underlying electromagnetic wave propagation problems are particularly challenging because of the heterogeneity of the propagation media and the complexity of the shapes of the head tissues. Unstructured mesh based solvers are attractive in this context since they allow for an accurate discretization of the interfaces between tissues where discontinuities of the field components occur. Noteworthy, a complete picture of the assessment of the potential adverse effects of head tissues exposure to mobile phone radiation requires to consider several factors such as the variations of the morphology and the electromagnetic characteristics of the tissues with the age, as well as the the type and positioning of the source (i.e. the mobile phone antenna). Whether such a study is conducted by a multi-parametric analysis or by relying on a strategy for uncertainty analysis, reducing the computational time of a single 3D simulation is highly desirable if not mandatory. In this work, this improvement of the computational performances of the simulation tool has been achieved by adapting the DGTD method to modern parallel computing systems with multiple GPU acceleration units. Acceleration factors ranging from 15 to 25 have been observed between the multiple CPU and multiple GPU simulations. For instance, a simulation using the DGTD-$\mathbb{P}_2$ method acting on a geometrical model consisting of 7,894,172 tetrahedral elements (i.e. involving 473,650,320 degrees of freedom) runs in a about 4 mn on 128 GPUs instead of 1 h 40 mn on a cluster of 128 CPUs. Although the measured floating point rates are only 10% of the theoretical performances of the considered generation of GPU systems and further optimization of our CUDA adaptation is certainly possible, the achieved reduction in the total simulation time will make it possible to tackle more challenging situations such a the exposure of heterogeneous models of the full body (e.g. for studying SAR levels induced in pregnant women), and more complex propagation models such as those characterizing dispersive media which is actually the case with biological tissues.

Acknowledgements. This work was granted access to the HPC resources of CCRT under the allocation 2010-t2010065004 made by GENCI (Grand Equipement National de Calcul Intensif).

References

[1] P. Bernardi, M. Cavagnaro, S. Pisa, and E. Piuzzi. Power absorption and temperature elevations induced in the human head by a dual-band monopole-helix antenna phone. *IEEE Trans. Microw. Theory Tech.*, 49(12):2539–2546, 2001.

[2] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

[3] J-C. Camart, D. Despretz, M. Chivé, and J. Pribetich. Modeling of various kinds of applicators used for microwave hyperthermia based on the FDTD method. *IEEE Trans. Microw. Theory Tech.*, 44(10):1811–1818, 1996.

[4] L.P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *9th Annual ACM Symposium Computational Geometry*, pages 274–280. ACM Press, 1993.

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

[6] E. Conil, A. Hadjem, F. Lacroux, M.F. Wong, and J. Wiart. Variability analysis of SAR from 20 MHz to 2.4 GHz for different adult and child models using finite-difference time-domain. *Phys. Med. Biol.*, 53:151–1525, 2008.

[7] D. Dunn, C.M. Rappaport, and A.J. Terzuoli. FDTD verification of deep-set brain tumor hyperthermia using a spherical microwave source distribution. *IEEE Trans. Microw. Theory Tech.*, 44:1769–1777, 1996.

[8] H. Fahs. Development of a *hp*-like discontinuous Galerkin time-domain method on non-conforming simplicial meshes for electromagnetic wave propagation. *Int. J. Numer. Anal. Mod.*, 6:193–216, 2009.

[9] 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. Num. Anal.*, 39(6):1149–1176, 2005.

[10] O.P. Gandhi, Q.-X. Li, and G. Kang. Temperature rise for the human head for cellular telephones and for peak SARs prescribed in safety guidelines. *IEEE Trans. Microw. Theory Tech.*, 49(9):1607–1613, 2001.

[11] P.-L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundary. *Comput. Methods Appl. Mech. Engrg.*, 92:269–288, 1991.

[12] N. Gödel, N. Nunn, T. Warburton, and M. Clemens. Scalability of higher-order discontinuous Galerkin FEM computations for solving electromagnetic wave propagation problems on GPU clusters. *IEEE. Trans. Magn.*, 46(8):3469–3472, 2010.

[13] J.S. Hesthaven and T. Warburton. Nodal high-order methods on unstructured grids. I. Time-domain solution of Maxwell's equations. *J. Comput. Phys.*, 181(1):186–221, 2002.

[14] J. Kim and Y. Rahmat-Samii. Implanted antennas inside a human body: simulations, designs, and characterizations. *IEEE Trans. Microw. Theory Tech.*, 52(8):1934–1943, 2004.

[15] A. Klöckner, T. Warburton, J. Bridge, and J.S. Hesthaven. Nodal discontinuous Galerkin methods on graphic processors. *J. Comput. Phys.*, 228:7863–7882, 2009.

[16] D. Komatitsch, G. Erlebacher, D. Göddeke, and D. Michéa. High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster. *J. Comput. Phys.*, 229(20):7692–7714, 2010.

[17] D. Komatitsch, D. Göddeke, G. Erlebacher, and D. Michéa. Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs. *Comput. Sci. Res. Dev.*, 25:75–82, 2010.

[18] D. Komatitsch, D. Michéa, and G. Erlebacher. Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA. *J. Parallel Distrib. Comput.*, 69:451–460, 2009.

[19] J.C. Lin, S. Hinrai, C.L. Chiang, W.L. Hsu, J-L. Su, and Y. Computer simulation and experimental studies of SAR distributions of interstitial arrays of sleeved-slot microwave antennas for hyperthermia treatment of brain tumors. *IEEE Trans. Microw. Theory Tech.*, 48(11):2191–2198, 2000.

[20] NVIDIA. Cuda C programming guide version 3.2. NVIDIA Corporation documentation, 2010.

[21] P. Ratiu, B. Hillen, J. Glaser, and D.P. Jenkins. *Medicine Meets Virtual Reality 11 - NextMed: Health Horizon*, volume 11, chapter Visible Human 2.0 - the next generation, pages 275–281. IOS Press, 2003.

[22] K.S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. Antennas Propag.*, AP-16:302–307, 1966.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399