Simulation of near-field plasmonic interactions with a local approximation order discontinuous Galerkin time-domain method

Jonathan Viquerat, Stéphane Lanteri

Simulation of near-field plasmonic interactions with a local approximation order discontinuous Galerkin time-domain method

Jonathan Viquerat[a], Stéphane Lanteri[a,]

[a] *INRIA Sophia Antipolis-Méditerranée research center,* NACHOS *project-team*
06902 Sophia Antipolis Cedex, France

Abstract

During the last ten years, the discontinuous Galerkin time-domain (DGTD) method has progressively emerged as a viable alternative to well established finite-difference time-domain (FDTD) and finite-element time-domain (FETD) methods for the numerical simulation of electromagnetic wave propagation problems in the time-domain. The method is now actively studied for various application contexts including those requiring to model light/matter interactions on the nanoscale. In this paper we further demonstrate the capabilities of the method for the simulation of near-field plasmonic interactions by considering more particularly the possibility of combining the use of a locally refined conforming tetrahedral mesh with a local adaptation of the approximation order.

Keywords: computational electromagnetics, nanophotonics, plasmonics, time-domain Maxwell equations, discontinuous Galerkin method, local adaptivity

1. Introduction

1.1. Generalities about the DGTD method

The DGTD method can be considered as a finite element method where the continuity constraint at an element interface is released. While it keeps almost all the advantages of the finite element method (large spectrum of applications, complex geometries, etc.), the DGTD method has other nice properties which explain the renewed interest it gains in various domains in scientific computing:

- It is naturally adapted to a high order approximation of the unknown field. Moreover, one may increase the degree of the approximation in the whole mesh as easily as for spectral methods but, with a DGTD method, this can also be done locally *i.e.* at the mesh cell level. In most cases, the approximation relies on a polynomial interpolation method but the method also offers the flexibility of applying local approximation strategies that best fit to the intrinsic features of the modeled physical phenomena.
- When the discretization in space is coupled to an explicit time integration method, the DG method leads to a block diagonal mass matrix independently of the form of the local approximation (e.g the type of polynomial interpolation). This is a striking difference with classical, continuous FETD formulations. Moreover, the mass matrix is diagonal if an orthogonal basis is chosen.
- It easily handles complex meshes. The grid may be a classical conforming finite element mesh, a non-conforming one or even a hybrid mesh made of various elements (tetrahedra, prisms, hexahedra, etc.). The DGTD method has been proven to work well with highly locally refined meshes. This property makes the DGTD method more suitable to the design of a *hp*-adaptive solution strategy (i.e. where the characteristic mesh size h and the interpolation degree p changes locally wherever it is needed).
- It is flexible with regards to the choice of the time stepping scheme. One may combine the discontinuous Galerkin spatial discretization with any global or local explicit time integration scheme, or even implicit, provided the resulting scheme is stable.
- It is naturally adapted to parallel computing. As long as an explicit time integration scheme is used, the DGTD method is easily parallelized. Moreover, the compact nature of method is in favor of high computation to communication ratio especially when the interpolation order is increased.

As in a classical finite element framework, a discontinuous Galerkin formulation relies on a weak form of the continuous problem at hand. However, due to the discontinuity of the global approximation, this variational formulation has to be defined at the element level. Then, a degree of freedom in the design of a discontinuous Galerkin scheme stems from the approximation of the boundary integral term resulting from the application of an integration by parts to the element-wise variational form. In the spirit of

Email address: Stephane.Lanteri@inria.fr (Stéphane Lanteri)

finite volume methods, the approximation of this boundary integral term calls for a numerical flux function which can be based on either a centered scheme or an upwind scheme, or a blend of these two schemes.

1.2. DGTD methods for nanophotonics/plasmonics

Numerical modeling of electromagnetic wave propagation in interaction with metallic nanostructures at optical frequencies requires to solve the system of Maxwell equations coupled to appropriate models of physical dispersion in the metal. In general, the Drude and Drude-Lorentz models are adopted although there are practical situations for which these models can fail to describe correcly the behavior of some materials (e.g. transition metals [1]-[2] and graphene [3]). Furthermore at some scales, non-local effects starts to play an important role [4]. The FDTD [5] method is a widely used approach for solving the systems of partial differential equations modeling nanophotonic applications. In this method, the whole computational domain is discretized using a structured (cartesian) grid. In spite of its flexibility and second-order accuracy in a homogeneous medium, the Yee scheme suffers from serious accuracy degradation when used to model curved objects or when treating material interfaces. Indeed, the so-called stair-casing approximation may lead to local zeroth-order and at most first-order accuracy; it may also produce locally non-convergent results [6]. Furthermore, for Maxwell's equations with discontinuous coefficients, the Yee scheme might not be able to capture the possible discontinuity of the solution across the interfaces [6].

Thus, with all its above-listed features, the DGTD method seems to be well suited to the numerical simulation of complex time-domain ectromagntic wave propagation problems. As a matter of fact, the DGTD method for solving the time domain Maxwell equations is increasingly adopted by several physics communities. Concerning nanophotonics, unstructured mesh based DGTD methods have been developed and have demonstrated their potentialities for being considered as viable alternatives to the FDTD method. The most remarkable achievements in the nanophotonics domain since 2009 are due to Busch *et al.* Busch [7]-[8]-[9] has been at the origin of seminal works on the development and application of the DGTD method in this domain. These works not only deal with the extension of the DGTD method with regards to the complex material models and source settings required by applications relevant to nanophotonics and plasmonics [10]-[11]-[2], but also to core contributions aiming at improving the accuracy and the efficiency of the proposed DGTD solvers [12]-[13]-[14]-[15].

Noteworthy, all these studies adopt a diffusive DGTD formulation based on upwind numerical fluxes. We have recently adapted the non-dissipative DGTD method initially introduced in [16] to deal with various local dispersion models for metallic nanostructures. An ADE formulation has been adopted. The resulting ADE-based DGTD method is detailed in [17] where we also study the stability and a priori convergence of the method. We first considered the case of Drude and Drude-Lorentz models and, further extend the proposed ADE-based DGTD method to be able to deal with a generalized dispersion model in which we make use of a Padé approximant to fit an experimental permittivity function. The numerical treatment of such a generalized dispersion model is also presented in [17]. The possibility of using non-conforming hybrid structured/unstructured (cubic/tetrahedral) meshes in the nanophotonics context has been considered in [18]. More recently, a high order diffusive DGTD method formulated on curvilinear tetrahedral meshes has been studied in [19].

1.3. Objectives of the present study

In most of the existing works on the development of high order DGTD methods for the numerical modeling of light/matter interactions on the nanoscale, the formulation of the method is derived assuming a uniform distribution of the polynomial order to the cells of the underlying mesh. However, in the case of a mesh showing large variations in cell size, the time step imposed by the smallest cells can be a serious hindrance when trying to exploit high approximation orders. Indeed, a potentially large part of the CPU time is spent in the update of the physical field inside small cells where high polynomial orders might not be essential, while they are necessary in the larger cells.

To overcome this limitation, several strategies can be considered. The first one consists in replacing the explicit scheme by an implicit scheme. This, however, is at the expense of the resolution of a large linear system at each time step [20]. In [21], the authors consider a hybrid formulation, where only the smallest cells are treated *via* an implicit scheme, while keeping an explicit time integration for the rest of the tesselation, thus limiting the time step constraint. Although this strategy provides very interesting results in terms of CPU speedup, maintaining high order time integration is difficult. Another possiblity is to exploit local time stepping combined to an explicit scheme: based on the size of the elements, the mesh is divided in regions, each of which being assigned an appropriate timestep for an explicit time integration (see [22]-[23]-[24]-[25]-[15]). As shown for example in [23], high order convergence in time can be preserved with such explicit local time stepping strategies. However, it seems difficult to ensure a good load balance in the case of a parallel implementation, given that the natural repartition is based on a time step criterion, instead of a parallel-related one.

A complementary strategy which is considered here relies on the use of non-uniform distribution of the polynomial order in the framework of a global time step DGDT method. By imposing low orders in small cells and high orders in large cells, it is possible to significantly alleviate both the global number of degrees of freedom and the time step restriction with a minimal impact on the method accuracy. Strategies exploiting locally adaptive (LA) formulations usually combine both h- and p-adaptivity in or-

der to concentrate the computational effort in the areas of high field variations. Here, the adopted point of view is quite different: starting from a given mesh and an uniform distribution of the polynomial order $\mathbb{P}_k$, the LA strategy exploits all the polynomial orders $\mathbb{P}_p$ with $p \leq k$ to obtain a solution of similar accuracy with a reduced computational cost.

In section 2, a DGTD formulation is presented that account for a variable polynomial order. In section 3, the resulting DGTD is tested for convergence, and an extended performance study is provided, both for sequential and parallel executions. Finally, in section 6, the CPU gains are evaluated on two configurations relevant to nanophotonics.

2. DGTD formulation

2.1. Problem statement

For the sake of simplicity, we consider the non-dispersive time-domain Maxwell equations for this presentation. The extension of the formulation presented below to the case of a local dispersion model is straightforward. The continuous, normalized Maxwell system is

$$\begin{aligned} \mu_r \frac{\partial \mathbf{H}}{\partial t} &= -\nabla \times \mathbf{E}, \\ \varepsilon_r \frac{\partial \mathbf{E}}{\partial t} &= \quad \nabla \times \mathbf{H} - \mathbf{J}. \end{aligned} \tag{1}$$

2.2. Notations

Let $\Omega \subset \mathbb{R}^3$ be a bounded convex domain, and $\mathbf{n}$ the unitary outward normal to its boundary $\partial\Omega$. Let Ω_h be a discretization of Ω, relying on a quasi-uniform triangulation $\mathcal{T}_h$ verifying $\mathcal{T}_h = \bigcup_{i=1}^{N} T_i$, where $N \in \mathbb{N}^*$ is the number of mesh elements, and $(T_i)_{i\in[\![1,N]\!]}$ the set of simplices. The internal faces of the discretization are denoted $a_{ik} = T_i \bigcap T_k$ if T_i and T_k are adjacent cells, and $\mathbf{n}_{ik}$ is defined as the unit normal vector to the face a_{ik}, oriented from T_i toward T_k. For each cell T_i, $\mathcal{V}_i$ is the set of indices $\{k \in [\![1, N]\!] \,|\, T_i \bigcap T_k \neq \emptyset\}$. We define the following approximation space V_h

$$V_h = \left\{ v \in \left(L^2(\Omega_h)\right)^3, v_{|T_i} \in (\mathbb{P}_p(T_i))^3 \,\forall\, T_i \in \mathcal{T}_h \right\},$$

where $\mathbb{P}_p(T_i)$ is the space of polynomials of maximum degree p on T_i. The semi-discrete fields, seeked in space V_h, are hereafter denoted $(\mathbf{H}_h, \mathbf{E}_h, \mathbf{J}_h)$, and on each cell T_i the restrictions $(\mathbf{H}_i, \mathbf{E}_i, \mathbf{J}_i) = (\mathbf{H}_h|_{T_i}, \mathbf{E}_h|_{T_i}, \mathbf{J}_h|_{T_i})$ are defined. A set of scalar basis functions $(\phi_{ik})_{1\leqslant k\leqslant d_i}$ is defined for each T_i, where d_i is the number of degrees of freedom (d.o.f.) per dimension. Additionally, to each scalar basis function, the three vectors $\boldsymbol{\phi}^{\boldsymbol{v}}_{\boldsymbol{ik}}$ are associated

$$\boldsymbol{\phi}^{\mathbf{1}}_{\boldsymbol{ik}} = \begin{bmatrix} \phi_{ik} \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{\phi}^{\mathbf{2}}_{\boldsymbol{ik}} = \begin{bmatrix} 0 \\ \phi_{ik} \\ 0 \end{bmatrix}, \boldsymbol{\phi}^{\mathbf{3}}_{\boldsymbol{ik}} = \begin{bmatrix} 0 \\ 0 \\ \phi_{ik} \end{bmatrix}.$$

One now seeks the approximations $\mathbf{E}_h$ and $\mathbf{H}_h$ of $\mathbf{E}$ and $\mathbf{H}$ in space V_h. The contribution of each cell is therefore defined as $\mathbf{E}_i = \mathbf{E}_h|_{T_i}$. Here, one must notice that, for a 3D system, $\mathbf{E}_i$ is actually a vector that has 3 components

$$\mathbf{E}_i = \begin{bmatrix} E_i^x \\ E_i^y \\ E_i^z \end{bmatrix},$$

each of which is locally expanded on the chosen set of basis functions

$$E_i^v = \sum_{j=1}^{d_i} E_{ij}^v \phi_{ij}, v \in \{x, y, z\}. \tag{2}$$

Therefore, for practical purpose, one defines three vectors of d_i components

$$\overline{\mathbf{E}}_i^{\,v} = \begin{bmatrix} E_{i1}^v \\ \vdots \\ E_{id_i}^v \end{bmatrix}, v \in \{x, y, z\},$$

as well as the following $3d_i$ components vector

$$\overline{\mathbf{E}}_{\,i} = \begin{bmatrix} (E_{ij}^x)_{1\leqslant j\leqslant d_i} \\ (E_{ij}^y)_{1\leqslant j\leqslant d_i} \\ (E_{ij}^z)_{1\leqslant j\leqslant d_i} \end{bmatrix}.$$

2.3. Discretization in space

We consider the system of time-domain Maxwell equations (1) with no current source term. The starting point is the following weak formulation

$$\begin{aligned} \int_{T_i} \mu_r \frac{\partial \mathbf{H}_i}{\partial t} \cdot \boldsymbol{\phi}^{\boldsymbol{v}}_{\boldsymbol{ik}} + \int_{T_i} \mathbf{E}_i \cdot \nabla \times \boldsymbol{\phi}^{\boldsymbol{v}}_{\boldsymbol{ik}} = \\ \sum_{l\in\mathcal{V}_i} \int_{a_{il}} (\mathbf{E}_* \times \mathbf{n}_{il}) \cdot \boldsymbol{\phi}^{\boldsymbol{v}}_{\boldsymbol{ik}}, \\ \int_{T_i} \varepsilon_r \frac{\partial \mathbf{E}_i}{\partial t} \cdot \boldsymbol{\phi}^{\boldsymbol{v}}_{\boldsymbol{ik}} - \int_{T_i} \mathbf{H}_i \cdot \nabla \times \boldsymbol{\phi}^{\boldsymbol{v}}_{\boldsymbol{ik}} = \\ - \sum_{l\in\mathcal{V}_i} \int_{a_{il}} (\mathbf{H}_* \times \mathbf{n}_{il}) \cdot \boldsymbol{\phi}^{\boldsymbol{v}}_{\boldsymbol{ik}}. \end{aligned}$$

We recall that the i subscript refers to the cell T_i of the underlying tetrahedral mesh where a $\mathbb{P}_p$ polynomial approximation of the field components is used. Exploiting the local field expansions from (2), one can cast the first volumic integral of the above system as the following matrix-vector products

$$\int_{T_i} \varepsilon_r \frac{\partial \mathbf{E}_i}{\partial t} \cdot \boldsymbol{\phi}^{\boldsymbol{x}}_{\boldsymbol{ik}} = \left(\mathbb{M}_i^{\varepsilon_r} \frac{\partial \overline{\mathbf{E}}_i^{\,x}}{\partial t} \right)_k,$$

where $\mathbb{M}_i^{\varepsilon_r}$ is the mass matrix, of dimension $d_i \times d_i$

$$\left(\mathbb{M}_i^{\varepsilon_r}\right)_{jk} = \int_{T_i} \varepsilon_r \phi_{ij} \phi_{ik}, \text{ with } (j,k) \in [\![1, d_i]\!]^2.$$

In a similar fashion, the curl integral can be cast as

$$\int_{T_i} \mathbf{H}_i \cdot \nabla \times \boldsymbol{\phi}_{\boldsymbol{ik}}^{\boldsymbol{x}} = \left(\mathbb{K}_i \times \overline{\mathbf{H}}_i\right)_k^x.$$

Here, the three stiffness matrices were introduced

$$\left(\mathbb{K}_i^v\right)_{jk} = \int_{T_i} \phi_{ij} \frac{\partial \phi_{ik}}{\partial v} \quad \text{for} \quad v \in \{x, y, z\},$$

with $(j,k) \in [\![1, d_i]\!]^2$. From the latter definition, we define the general $3d_i \times d_i$ stiffness matrix that will be used in the final system

$$\overline{\mathbb{K}}_i = \begin{bmatrix} \mathbb{K}_i^x \\ \mathbb{K}_i^y \\ \mathbb{K}_i^z \end{bmatrix}.$$

The flux integral, however, must undergo a different treatment. For the sake of simplicity, the centered flux is considered, the generalization to other fluxes being straightforward. Consider a neighbor cell T_l of T_i with a $\mathbb{P}_m$ polynomial expansion. The flux integral on their common face from the T_i side for the x component is

$$\begin{aligned} &\int_{a_{il}} (\mathbf{H}_* \times \mathbf{n}_{il}) \cdot \boldsymbol{\phi}_{\boldsymbol{ik}}^{\boldsymbol{x}} = \int_{a_{il}} \left(H_*^y n_{il}^z - H_*^z n_{il}^y\right) \phi_{ik} \\ &= \int_{a_{il}} \left(\frac{H_i^y + H_l^y}{2} n_{il}^z - \frac{H_i^z + H_l^z}{2} n_{il}^y \right) \phi_{ik}. \end{aligned} \tag{3}$$

Consider the following expansions for H_i^y and H_l^y, and their counterparts for H_i^z and H_l^z

$$H_i^y = \sum_{q=1}^{p} H_{iq}^y \phi_{iq} \quad \text{and} \quad H_l^y = \sum_{r=1}^{m} H_{lr}^y \phi_{lr}. \tag{4}$$

Plugging (4) in (3) leads to

$$\begin{aligned} &\int_{a_{il}} (\mathbf{H}_* \times \mathbf{n}_{il}) \cdot \boldsymbol{\phi}_{\boldsymbol{ik}}^{\boldsymbol{x}} \\ &= \int_{a_{il}} \frac{n_{il}^z}{2} \left(\sum_{q=1}^{p} H_{iq}^y \phi_{iq} \phi_{ik} + \sum_{r=1}^{m} H_{lr}^y \phi_{lr} \phi_{ik} \right) \\ &- \int_{a_{il}} \frac{n_{il}^y}{2} \left(\sum_{q=1}^{p} H_{iq}^z \phi_{iq} \phi_{ik} + \sum_{r=1}^{m} H_{lr}^z \phi_{lr} \phi_{ik} \right) \\ &= \frac{1}{2} \sum_{q=1}^{p} \left(H_{iq}^y n_{il}^z - H_{iq}^z n_{il}^y \right) \int_{a_{il}} \phi_{iq} \phi_{ik} \\ &+ \frac{1}{2} \sum_{r=1}^{m} \left(H_{lr}^y n_{il}^z - H_{lr}^z n_{il}^y \right) \int_{a_{il}} \phi_{lr} \phi_{ik} \\ &= \frac{1}{2} \sum_{q=1}^{p} \mathbf{H}_{iq} \times \mathbf{n}_{il} \int_{a_{il}} \phi_{iq} \phi_{ik} \\ &+ \frac{1}{2} \sum_{r=1}^{m} \mathbf{H}_{lr} \times \mathbf{n}_{il} \int_{a_{il}} \phi_{lr} \phi_{ik} \\ &= \left(\mathbb{S}_{il} \left(\overline{\mathbf{H}}_{*,i} \times \mathbf{n}_{il} \right)\right)_k^x + \left(\mathbb{S}_{il}^* \left(\overline{\mathbf{H}}_{*,l} \times \mathbf{n}_{il} \right)\right)_k^x. \end{aligned} \tag{5}$$

We introduce the *flux* or *interface* matrices

$$\left(\mathbb{S}_{il}\right)_{jk} = \int_{a_{il}} \phi_{ij} \phi_{ik} \quad \text{and} \quad \left(\mathbb{S}_{il}^*\right)_{rk} = \int_{a_{il}} \phi_{lr} \phi_{ik}. \tag{6}$$

The derivation (3) easily extends to the other components, as well as other flux choices. We recall one of the most spread flux formulation

$$\begin{aligned} \mathbf{E}_* &= \frac{1}{Y_i + Y_l} \left(\{Y\mathbf{E}\}_{il} + \alpha \mathbf{n} \times [\![\mathbf{H}]\!]_{il} \right), \\ \mathbf{H}_* &= \frac{1}{Z_i + Z_l} \left(\{Z\mathbf{H}\}_{il} - \alpha \mathbf{n} \times [\![\mathbf{E}]\!]_{il} \right), \end{aligned} \tag{7}$$

where $\{A\}_{il} = A_i + A_l$ is twice the mean value of A across the interface, $[\![A]\!]_{il} = A_l - A_i$ is the jump of A across the interface, and Y and Z are respectively the inductance and impedance of the considered materials. $\alpha \in [0,1]$ is a tunable parameter that allows to vary between the centered flux for $\alpha = 0$, to a fully upwind flux for $\alpha = 1$.

To summarize, the flux integral is cut in two parts: (i) the part corresponding to local information, which is integrated *via* the *regular* flux matrix $\mathbb{S}_{il}$, and (ii) the part corresponding to the neighbor information, which is integrated *via* the $\mathbb{S}_{il}^*$ matrix. In the case of a conformous interface (*i.e.* $p = m$), $\mathbb{S}_{il}^* = \mathbb{S}_{il}$. However, for a $\mathbb{P}_p - \mathbb{P}_m$ interface, $\mathbb{S}_{il}^*$ is a non-conforming rectangular matrix of size $s(p) \times s(m)$ (we recall that $s(p) = \frac{(p+1)(p+2)}{2}$). From this point, the remaining of the derivation is similar to the standard case (see for example [19]). The final semi-discrete scheme is

$$\begin{aligned} \overline{\mathbb{M}}_i^{\mu_r} \frac{\partial \overline{\mathbf{H}}_i}{\partial t} &= -\overline{\mathbb{K}}_i \times \overline{\mathbf{E}}_i + \sum_{l \in \mathcal{V}_i} \overline{\mathbb{S}}_{il} \left(\overline{\mathbf{E}}_{*,i} \times \mathbf{n}_{il} \right) \\ &\quad + \sum_{l \in \mathcal{V}_i} \overline{\mathbb{S}}_{il}^* \left(\overline{\mathbf{E}}_{*,l} \times \mathbf{n}_{il} \right), \\ \overline{\mathbb{M}}_i^{\varepsilon_r} \frac{\partial \overline{\mathbf{E}}_i}{\partial t} &= \quad \overline{\mathbb{K}}_i \times \overline{\mathbf{H}}_i - \sum_{l \in \mathcal{V}_i} \overline{\mathbb{S}}_{il} \left(\overline{\mathbf{H}}_{*,i} \times \mathbf{n}_{il} \right) \\ &\quad - \sum_{l \in \mathcal{V}_i} \overline{\mathbb{S}}_{il}^* \left(\overline{\mathbf{H}}_{*,l} \times \mathbf{n}_{il} \right), \end{aligned} \tag{8}$$

with the following definitions

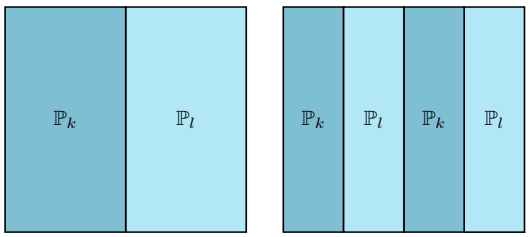
$$\begin{aligned} \left(\mathbb{S}_{il}\right)_{jk} &= \int_{a_{il}} \phi_{ij} \phi_{ik} \\ \left(\mathbb{S}_{il}^*\right)_{rk} &= \int_{a_{il}} \phi_{lr} \phi_{ik}. \end{aligned} \tag{9}$$

2.4. Time integration

In this work, we use the optimized 14 stages fourth-order low-storage Runge-Kutta scheme developped by Niegemann *et al.* [14], hereafter designated as LSRK4 scheme. Its stability region is optimized for the upwind DG eigenspectrum, and is thus an efficient time-integration scheme for our needs. The reader can refer to the aforementionned reference for more details.

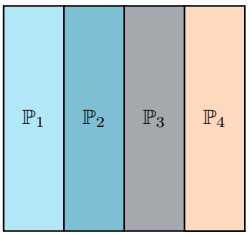
3. Numerical study of the *h*-convergence

In this section, we perform a preliminary validation of the LA-DGTD (Locally Adaptive DGTD) method by studying its numerical convergence. For this purpose, we consider the test problem of the propagation of an eigenmode in a cubic PEC cavity. Four meshes of increasing refinement, M1 to M4, are used. Unless stated otherwise, a fully upwind flux is used, coupled to the LSRK4 time scheme. In the present case, the used meshes are uniform, and the mesh cells all have the same size. To begin, we separate the computational domain into two vertical stripes of equal sizes, with different values of the polynomial order (see figure 1(a)). The obtained results are given in table 1. As expected for a $\mathbb{P}_k - \mathbb{P}_l$ configuration, the asymptotic h-convergence order is $\min(k, l) + 1$.



(a) Test configuration for h-convergence

(b) Influence of the number of non-conformeous interfaces



(c) Influence of the type of non-conformeous interfaces

Figure 1: **Order distribution for the h-convergence validation**. To test convergence, the domain is arbitrarily cut in two halves, each part receiving a different order (figure 1(a)). In order to assess the impact of non-conformeous interfaces, the domain is cut in four equal stripes, thus doubling the number of $\mathbb{P}_k - \mathbb{P}_l$ faces while keeping the same number of tetrahedron per order (figure 1(a)). Another test is conducted by setting a different order in each quarter (figure 1(c)).

In order to evaluate the impact of non-conforming interfaces, the computational domain is now distributed into four equal size stripes (see figure 1(b)). Hence, the total number of tetrahedra for each value of the interpolation order remains the same as in figure 1(a), but the number of interfaces is twice as high. As can be seen in table 1, for the coarsest meshes the higher amount of non-conforming faces yields a slightly higher (but still acceptable) L^2 error. However, this error overhead vanishes for refined meshes. In a last numerical test, another four stripes division of the computational domain is used, where each part receives a different order (see figure 1(c)). As in the previous cases, and since all the cells are of same size, the numerical error is driven by the lowest approximation order.

4. Order distribution strategy

Starting with a given unstructured mesh, it is clear that the distribution of the interpolation order to the mesh cells will have a major impact on the obtained accuracy, as well as on the time required to obtain the numerical solution. Let us assume that the solution is obtained on the given mesh with a homogeneous polynomial order $\mathbb{P}_k$. The point is here to see how, with a good distribution of polynomial orders $\mathbb{P}_l$ with $l \leq k$, a solution of similar accuracy can be obtained at a lower computational effort. At first glance, it seems that configurations including small geometrical details, or small gaps between two structures, could benefit from such a strategy. For this reason, for any given mesh, we define the following quantity

$$\overline{h} = \frac{h_{\max}}{h_{\min}},$$

which represents the *heterogeneity* in terms of cell size of the mesh. In the following, Δt_i represents the time step corresponding to the cell T_i, computed following the formula ..., while Δt_i^p represents the effective time step obtained if cell T_i is discretized with a $\mathbb{P}_p$ polynomial expansion

$$\Delta t_i^p = \mathrm{CFL}(p)\Delta t_i,$$

where Δt_i is the maximal acceptable timestep for cell T_i, computed as

$$\Delta t_i = c_p \frac{V_{T_i}}{A_{T_i}}, \tag{10}$$

where V_{T_i} and A_{T_i} are respectively the volume and the area of cell T_i, and c_p is an order-dependant constant. The normalized time step includes a rough estimate of the computational charge induced by the polynomial order, and is defined as

$$\overline{\Delta t}_i^{\,p} = \frac{\Delta t_i^p}{n(p)} = \frac{\mathrm{CFL}(p)\Delta t_i}{n(p)},$$

where $n(p)$ is the number of degrees of freedom inside a $\mathbb{P}_p$ cell. Finally, we define p_{min} and p_{max} the minimal and maximal user-authorized orders in the mesh. We also add the following constraint: non-conforming faces cannot connect cells with an order jump higher than one (the allowed configurations are presented on figure 2). Indeed, it is preferable to restrain the number and size of matrices in memory in order to improve data locality. Additionaly, it leads to a robust distribution strategy, as will be shown hereafter.

The first step of the algorithm consists in computing the local time steps Δt_i and sorting them by ascending order. The cell with the smaller Δt receives order p_{min} and we compute its normalized time step $\overline{\Delta t}_1^{\,p_{min}}$. Two temporary variables, p_{loc} and $\overline{\Delta t}_{\,loc}$, respectively store the

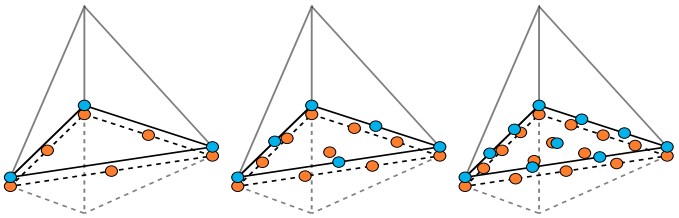


Figure 2: **Authorized interfaces in the local order of approximation implementation**. Order jumps are limited to one, yielding three types of interfaces for $(p_{min}, p_{max}) = (1, 4)$.

Table 1: **Error levels and convergence rates of the cubic cavity case** for mixed orders of approximation on meshes of increasing refinement. In the case of mixed orders $\mathbb{P}_k - \mathbb{P}_l$, 1 refers to a domain cut in two halves (see figure 1(a)), and 2 to a domain cut in four stripes (see figure 1(b)). The case $\mathbb{P}_1 - \mathbb{P}_4$ corresponds to a domain cut in four quarters, as depicted on figure 1(c). All simulations were run with upwind fluxes and LSRK4 time integration.

		M1		**M2**	
		$\|\mathbf{E} - \mathbf{E}_h\|$	r	$\|\mathbf{E} - \mathbf{E}_h\|$	r
$\mathbb{P}_1$	–	2.87×10^{-1}	–	6.05×10^{-2}	2.25
$\mathbb{P}_2$	–	1.47×10^{-2}	–	1.36×10^{-3}	3.43
$\mathbb{P}_3$	–	9.24×10^{-4}	–	5.87×10^{-5}	3.98
$\mathbb{P}_4$	–	9.45×10^{-5}	–	3.11×10^{-6}	4.92
$\mathbb{P}_1 - \mathbb{P}_2$	1	2.65×10^{-1}	–	3.38×10^{-2}	2.97
	2	2.07×10^{-1}	–	3.55×10^{-2}	2.54
$\mathbb{P}_2 - \mathbb{P}_3$	1	8.50×10^{-3}	–	9.21×10^{-4}	3.21
	2	8.70×10^{-3}	–	9.16×10^{-4}	3.25
$\mathbb{P}_3 - \mathbb{P}_4$	1	6.73×10^{-4}	–	4.41×10^{-5}	3.93
	2	6.81×10^{-4}	–	4.47×10^{-5}	3.93
$\mathbb{P}_1 - \mathbb{P}_4$	–	2.65×10^{-1}	–	3.38×10^{-2}	2.97

		M3		**M4**	
		$\|\mathbf{E} - \mathbf{E}_h\|$	r	$\|\mathbf{E} - \mathbf{E}_h\|$	r
$\mathbb{P}_1$	–	8.66×10^{-3}	2.80	1.46×10^{-3}	2.57
$\mathbb{P}_2$	–	1.75×10^{-4}	2.96	2.19×10^{-5}	3.00
$\mathbb{P}_3$	–	3.72×10^{-6}	3.98	2.33×10^{-7}	4.00
$\mathbb{P}_4$	–	1.98×10^{-7}	3.97	1.15×10^{-8}	4.11
$\mathbb{P}_1 - \mathbb{P}_2$	1	6.15×10^{-3}	2.46	1.42×10^{-3}	2.12
	2	6.30×10^{-3}	2.49	1.43×10^{-3}	2.14
$\mathbb{P}_2 - \mathbb{P}_3$	1	1.18×10^{-4}	2.96	1.48×10^{-5}	3.00
	2	1.17×10^{-4}	2.97	1.48×10^{-5}	2.98
$\mathbb{P}_3 - \mathbb{P}_4$	1	2.80×10^{-6}	3.98	1.76×10^{-7}	4.00
	2	2.85×10^{-6}	3.97	1.79×10^{-7}	3.99
$\mathbb{P}_1 - \mathbb{P}_4$	–	6.15×10^{-3}	2.46	1.42×10^{-3}	2.12

current order assigned to the cells and the current restrictive normalized time step. For a given cell, the normalized time step for increased order $p_{loc} + 1$ is compared to the current limiting normalized time step $\overline{\Delta t}_{loc}$. In the case where the first is higher than the second, switching to the higher order is assumed to have a limited impact on the final time step. Hence, p_{loc} is increased by one, and $\overline{\Delta t}_{loc}$ is updated. The procedure is summarized in algorithm 1, whose performances are assessed in next section.

5. Order distribution strategy

5.1. Single-core speedup

To evaluate the gains provided by the LA-DGDT strategy, we consider the three meshes M1, M2 and M3 shown on figure 3. These meshes are obtained by the tesselation of a cubic PEC cavity corresponding to the test problem considered previously, a local refinement being imposed on one side of the box. The characteristics of these meshes are summarized in table 2. For each mesh, the cavity mode is computed sequentially for 5 periods. As before, CPU time and maximum L^2 error are stored. The results obtained for homogeneous and mixed orders are presented in table 3.

Table 2: **Characteristics of the locally refined cubic cavity meshes.** n_s is the number of vertices, n_t the number of tetrahedrons and r is the ratio between the largest and the smallest cells in the mesh.

	M1	**M2**	**M3**
n_s	427	1429	11975
n_t	1513	5042	42652
r	10.2	100.9	1000.8

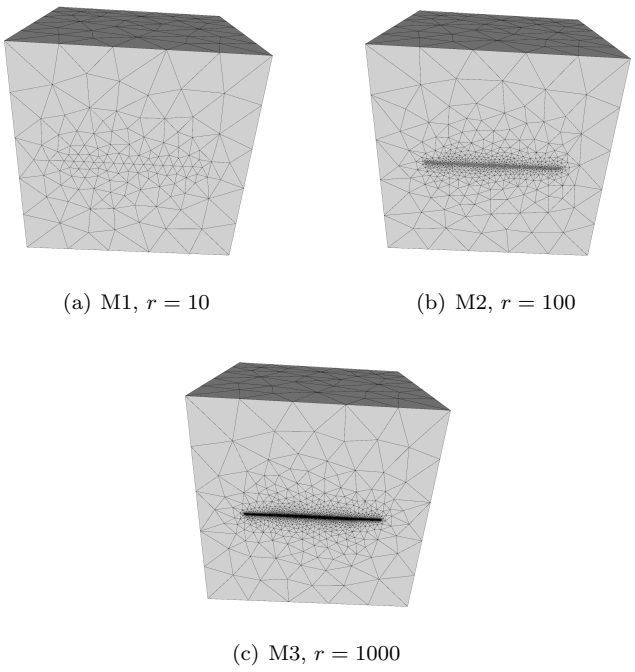
First, we note that the memory occupation has not been optimized in the present implementation of the LA-DGTD strategy. Hence, we can expect that mixed order computations will require the same memory occupation than homogeneous order ones. For mixed order solutions, the speedup of a $\mathbb{P}_k - \mathbb{P}_l$ computation is obtained by comparing its CPU time with that of a full $\mathbb{P}_{\max(k,l)}$ computation. For the three considered meshes, mixing two polynomial orders leads to speedups ranging from 1.5 to 2.2. One can remark that there exists a relation between

Algorithm 1 Polynomial order distribution

for $i \leftarrow 1, n_t$ **do** ▷ Compute timestep for each cell
  Compute Δt_i
end for

Sort cells by ascending Δt_i
$p(1) \leftarrow p_{\min}$
$\overline{\Delta t}_{\text{loc}} \leftarrow \overline{\Delta t}_1^{p_{\min}}$
$p_{\text{loc}} \leftarrow p_{\min}$

for $i \leftarrow 2, n_t$ **do** ▷ Go over cells by ascending order of Δt_i
  if $p_{\text{loc}} + 1 > p_{\max}$ **then** ▷ Check that we do not exceed $p_{\max}$
    $p(i) \leftarrow p_{\max}$
  else
    Compute $\overline{\Delta t}_i^{p_{\text{loc}}+1}$
    if $\overline{\Delta t}_i^{p_{\text{loc}}+1} > \overline{\Delta t}_{\text{loc}}$ **then** ▷ Check if it is worth changing order
      $\overline{\Delta t}_{\text{loc}} \leftarrow \overline{\Delta t}_i^{p_{\text{loc}}+1}$ ▷ Update the limiting timestep
      $p(i) \leftarrow p_{\text{loc}} + 1$
      $p_{\text{loc}} \leftarrow p_{\text{loc}} + 1$ ▷ Update the current order
    else
      $p(i) \leftarrow p_{\text{loc}}$
    end if
  end if
end for



(a) M1, $r = 10$ (b) M2, $r = 100$

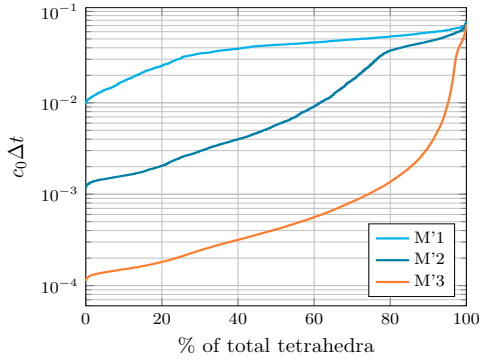
(c) M3, $r = 1000$

Figure 3: **Meshes for the cubic cavity mode with local refinement** on one side.

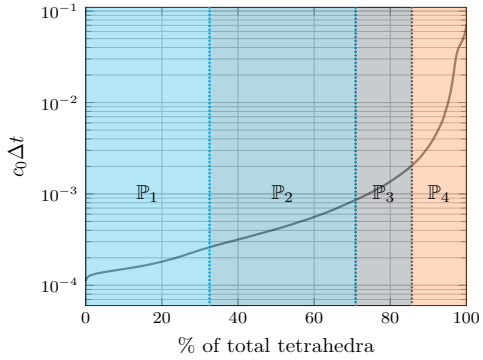
r and the obtained speedup. An interesting point is that the obtained L^2-errors are less than 1% higher than those of the homogeneous polynomial order computations. Mixing three orders does not provide any improvement for the M1 mesh, *i.e.* the distribution algorithm did not map the highest polynomial order to a cell of the mesh. This can be easily understood by looking at the compared Δt distribution of the three meshes (see figure 4). Since the algorithm imposes the lowest order for the cell of smallest Δt, a certain range for the Δt is required to exploit the highest polynomial orders. For example, the $\mathbb{P}_1 - \mathbb{P}_4$ distribution is shown on the same figure for mesh M3. For meshes M2 and M3, however, very interesting gains are obtained, with speedups ranging from 3 to 4.5. In this case, it seems that a higher r implies a higher benefit from the LA-DGTD strategy based on the distribution algorithm 1. Finally, mixing polynomial orders 1 to 4 roughly provides a speedup of 6, while increasing the global L^2 error by less than 1%.

5.2. Parallel load balancing

Parallel computing is a mandatory path for reducing the cost of simulations of complex electromagnetic wave propagation problems such as those pertaining to nanophotonics and plasmonics. In this section, we present the results obtained when trying to balance the computational load for a parallel implementation of the LA-DGTD method. The MeTiS [26] graph partitioning tool is used to split the computational domain in subdomains, each of which is then mapped on a core of parallel system. The communications between the cores are handled *via* the MPI



(a) Δt distribution for meshes M1, M2 and M3



(b) Order distribution for mesh M3

Figure 4: **Compared time-step distribution in M1, M2 and M3** (left), and order distribution for the $\mathbb{P}_1 - \mathbb{P}_4$ case on M3 (right).

standard. At the end of each computation, each CPU core returns its own CPU time, excluding the time spent in the MPI communication routines. Hence, a good load balance between the cores will manifest as nearly-identical CPU times for all cores. To reach this result, the partitioning algorithms of the MeTiS package can be provided with a weight w_i for each cell T_i. During the partitioning, this weight is taken into account so that the total weight of the various subdomains are as close as possible. Here, the following definition of the weight is used

$$w_i = n(p_i) + \sum_{l \in \mathcal{V}_i} \max(s(p_i), s(p_l)) \qquad (11)$$

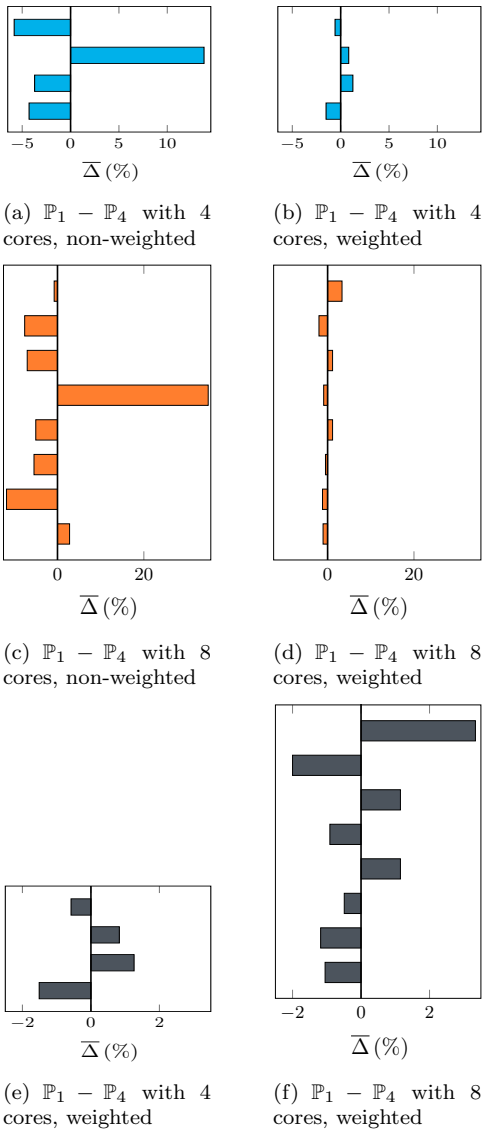
This weighted partitioning is applied to the test problem considered in section 5.1 using the M3 mesh and running 100 time iterations. Given the small number of tetrahedra in the mesh, the study is limited to 4 and 8 subdomains partitions. For a simpler comparison between the two partitionings, the relative deviation $\overline{\Delta}$ (in %) to the mean value is computed for each core. First, the effect of the weighting is assessed by comparing relative deviations obtained from weighted and non-weighted partitions. The results for $\mathbb{P}_1 - \mathbb{P}_4$ approximation with 4 and 8 CPUs are shown on figure 5. As can be seen, the use of a weighting pattern is mandatory to preserve good parallel performances. For both 4 and 8 cores, applying the weighted partitionning results in a maximal relative imbalance lower than 5%. To further explore the behavior of the algorithm in parallel, the CPU load balances with 4 and 8 cores are re-plotted with matching scales. As expected, increasing the number of cores also increases the maximal imbalance between cores, though in reasonable bounds.

6. Plasmonic simulations

6.1. Plasmonic nanolens

This section presents the computation of the field enhancement obtained in a plamonic nanolens device. To overcome the limitation of the diffraction limit, it is possible to exploit the focusing effect provided by coupled surface plasmons [27]. A typical nanolens is composed of a chain of metallic nanoparticles (nanospheres being the most common) of decreasing size, aligned with the polarization direction of the incident field (see figure 6). When the nanospheres are of significantly different sizes, the local field enhancement of the first particle is not perturbed by the second one because of its small relative size. As a result, the locally enhanced field of the first particle acts as an incident field for the second particle, resulting in a second enhancement, and so on. Eventually, the strongest enhancement is obtained in the gap between the two smaller particles [28].

Here, we consider a nanolens made of three gold spheres. The geometry is taken from [28]. The respective radii are 45, 15 and 4.5 nanometers, while the spacings between the sphere surfaces are respectively 4.5 and 1.5 nanometers. Gold is described by a Drude model of parameters $\varepsilon_\infty = 3.7362$, $\omega_d = 1.387 \times 10^7$ GHz and $\gamma_d = 4.515 \times 10^4$ GHz. The Drude model is a basic model used to describe the response of free electrons in metal in the THz regime [29]. The nanolens is illuminated *via* a total-field/scattered-field (TF/SF) interface, with a wideband plane wave of central frequency 700 THz, whose polarization is aligned with the natural axis of the lens (here, the $x+$ direction). The domain is terminated with a complex frequency-shifted perfectly-matched layer (CFS-PML) [30]. Finally, a probe point is set at half distance between each pair of spheres. At these positions, the discrete Fourier transform of the field is computed, and the field enhancement $g = \frac{|\hat{E}_x|}{|\hat{E}_i|}$ is deduced. To obtain a proper resolution of the geometry, very small elements must be used on the surface of the smallest sphere, as well as in the smallest gap, while the rest of the geometry (largest sphere, vacuum and PMLs) are meshed with much coarser elements (see mesh on figure 7). As a result, the r factor is here superior to 800. To obtain convergence with an homogeneous order over the whole mesh, $\mathbb{P}_3$ approximation is required. The corresponding numerical solution is obtained in 49 hours 48 minutes on 16 cores, and is taken as a reference. To exploit the LA-DGTD method, $\mathbb{P}_1$ to $\mathbb{P}_3$ approximations are used. The order distribution with respect to time step is shown on figure 8(a). To further understand the behavior of the repartition algorithm, a visual representation is added on



(a) $\mathbb{P}_1 - \mathbb{P}_4$ with 4 cores, non-weighted

(b) $\mathbb{P}_1 - \mathbb{P}_4$ with 4 cores, weighted

(c) $\mathbb{P}_1 - \mathbb{P}_4$ with 8 cores, non-weighted

(d) $\mathbb{P}_1 - \mathbb{P}_4$ with 8 cores, weighted

(e) $\mathbb{P}_1 - \mathbb{P}_4$ with 4 cores, weighted

(f) $\mathbb{P}_1 - \mathbb{P}_4$ with 8 cores, weighted

Figure 5: **Weighted against non-weighted parallel load balance on 4 and 8 cores with M3 mesh**. Each bar corresponds to the relative imbalance of a single CPU to the average value computed over all processors. As can be seen, the weighting restores a good balance of the CPU loads, with no relative imbalance exceeding 5%.

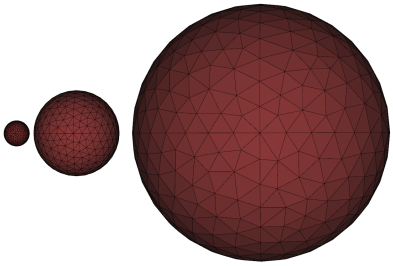


Figure 6: **Nanolens composed of three metallic spheres.**

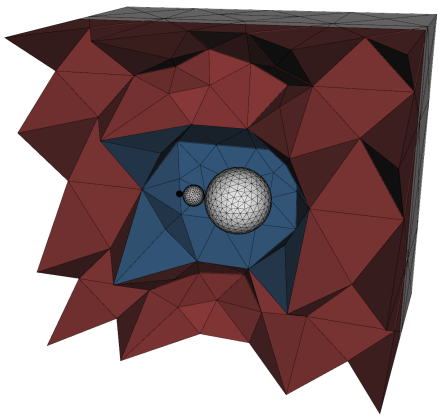


Figure 7: **Mesh setup for a metallic nanolens.** The gray cells correspond to the metallic spheres, the blue cells to vacuum, while the red cells constitute the PML region. For this mesh, the ratio r is above 400.

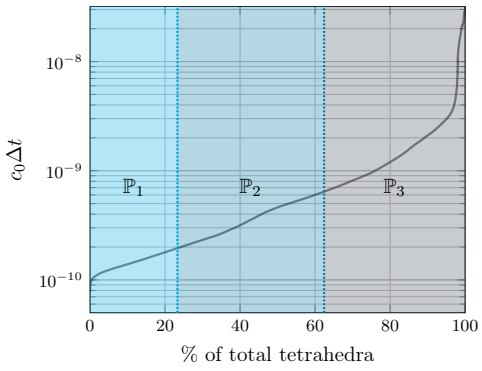
figure 8(b). As expected, first order polynomials are assigned to the elements lying on the surface of the smallest sphere, while second order approximation is used in its vicinity and for the surface of the second sphere. The rest of the mesh is discretized with third order polynomials.

The computed field enhancements are presented on figure 9. As stated in the litterature, particularly intense fields are obtained between the two smallest sphere, where enhancements up to 700 are observed. The $\mathbb{P}_1 - \mathbb{P}_3$ solution is obtained in 19 hours and 17 minutes on 16 cores, hence providing a speedup of 2.6 over the full $\mathbb{P}_3$ solution. The observed error over the frequency range of interest is less than 1 %. Field maps of the E_y component are presented on figure 10 for $\mathbb{P}_1$, $\mathbb{P}_1 - \mathbb{P}_3$ and $\mathbb{P}_3$ polynomial approximations. As expected, the field intensity between the spheres is underestimated for the full $\mathbb{P}_1$ computation, while they are almost identical for $\mathbb{P}_1 - \mathbb{P}_3$ and $\mathbb{P}_3$ approximations.

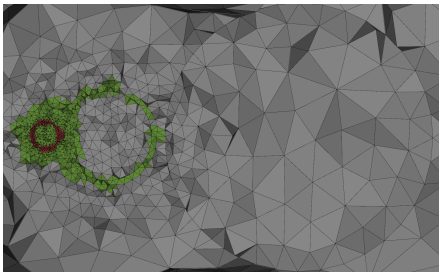
6.2. Bowtie nanoantenna

As a second application, we consider here the computation of the extinction cross section of a metallic bowtie nanoantenna. These structures are actively studied for the very strong field enhancement they provide between the tips of the two triangular nanoparticles (see figure 11), which is known to be inversely proportionnal to the size of the gap. Hence, bowtie nanoantennas are good candidates for surface-enhanced Raman spectroscopy (SERS) applications [31]. Recent advances in lithography techniques allowed the creation of structures with gaps as small as 3 nm [32], while the typical size of the full structure can get close to 200 nm. Additionally, realistic geometries of such antennas include small roundings at the edges and tips, whose typical size is between a few to a few tens of nanometers [33].

In the present case, we consider a pair of 10 nm thick, equilateral prisms of edge length 100 nm, with a spacing gap of 3 nm. The rounding radius is 2 nm, and is uniform



(a) Order distribution for the nanolens mesh



(b) Order selection in the vicinity of the lens

Figure 8: **Polynomial order repartition for the nanolens mesh** with respect to time-step (top), and geometrical repartition (bottom) for the $\mathbb{P}_1 - \mathbb{P}_3$ case. The red elements correspond to $\mathbb{P}_1$ approximation, the green ones to $\mathbb{P}_2$, and the gray ones to $\mathbb{P}_3$.

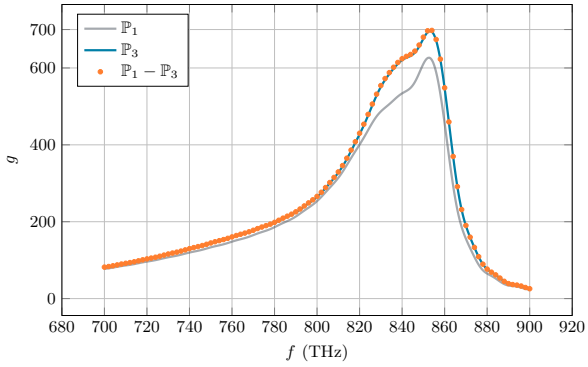
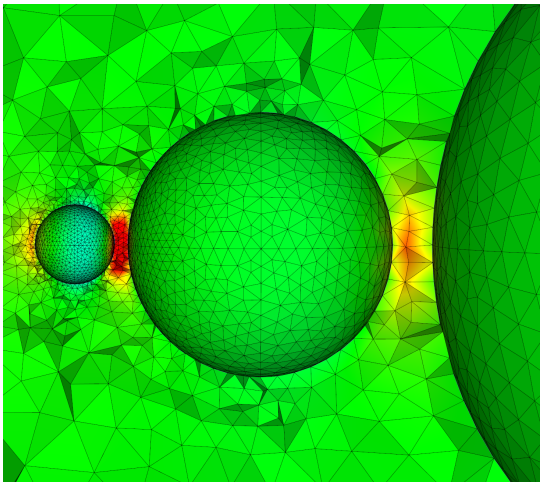
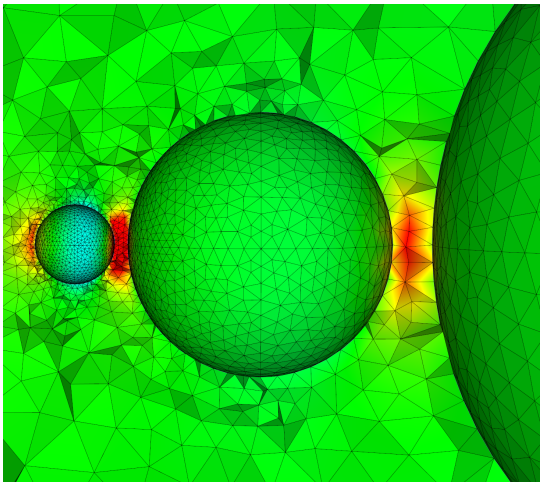


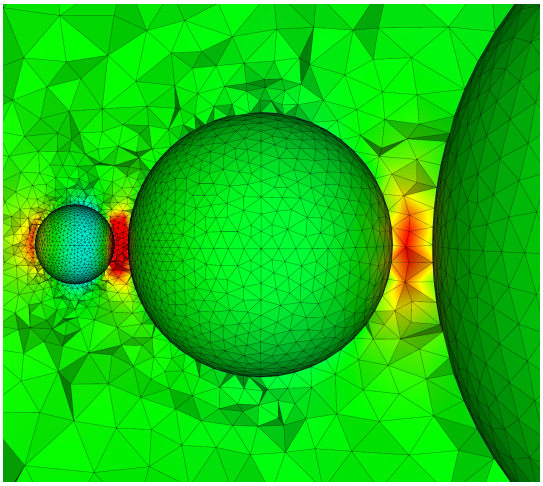
Figure 9: **Field enhancement in the vicinity of the smallest sphere of a self-similar nanolens** obtained with $\mathbb{P}_1$, $\mathbb{P}_3$ and $\mathbb{P}_1 - \mathbb{P}_3$ approximations. Less than 1 % of relative error is observed between full $\mathbb{P}_3$ and $\mathbb{P}_1 - \mathbb{P}_3$ computations, for a speedup factor of 2.6.



(a) $\mathbb{P}_1$



(b) $\mathbb{P}_1 - \mathbb{P}_3$



(c) $\mathbb{P}_3$

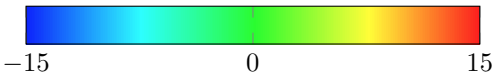


Figure 10: E_y **field map in the nanolens device** at $t = 10$ fs. For the three views, the field values are scaled to $[-15, 15]$.

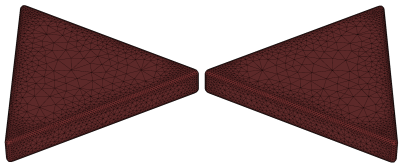


Figure 11: **Bowtie nanoantenna with rounded edges.**

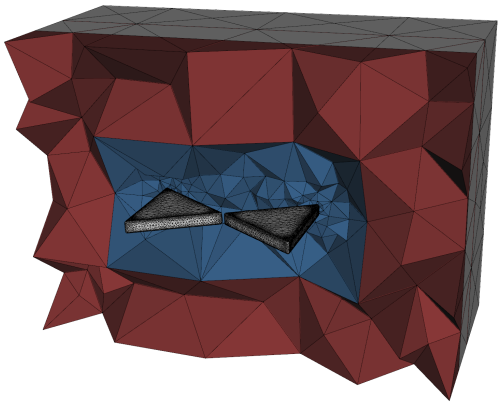
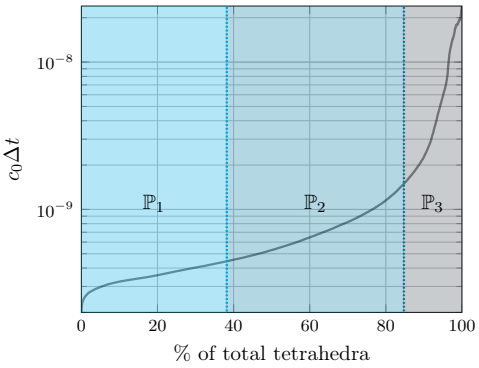
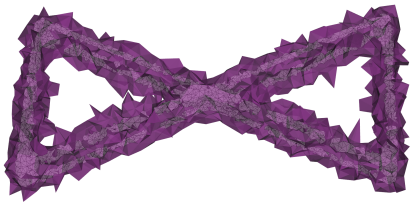


Figure 12: **Mesh setup for a bowtie nanoantenna.** The gray cells correspond to the nanoantenna, the blue cells to vacuum, while the red cells constitute the PML region. For this mesh, the ratio r is close to 275.



(a) Order distribution for the bowtie mesh



(b) Order selection in the vicinity of the antenna

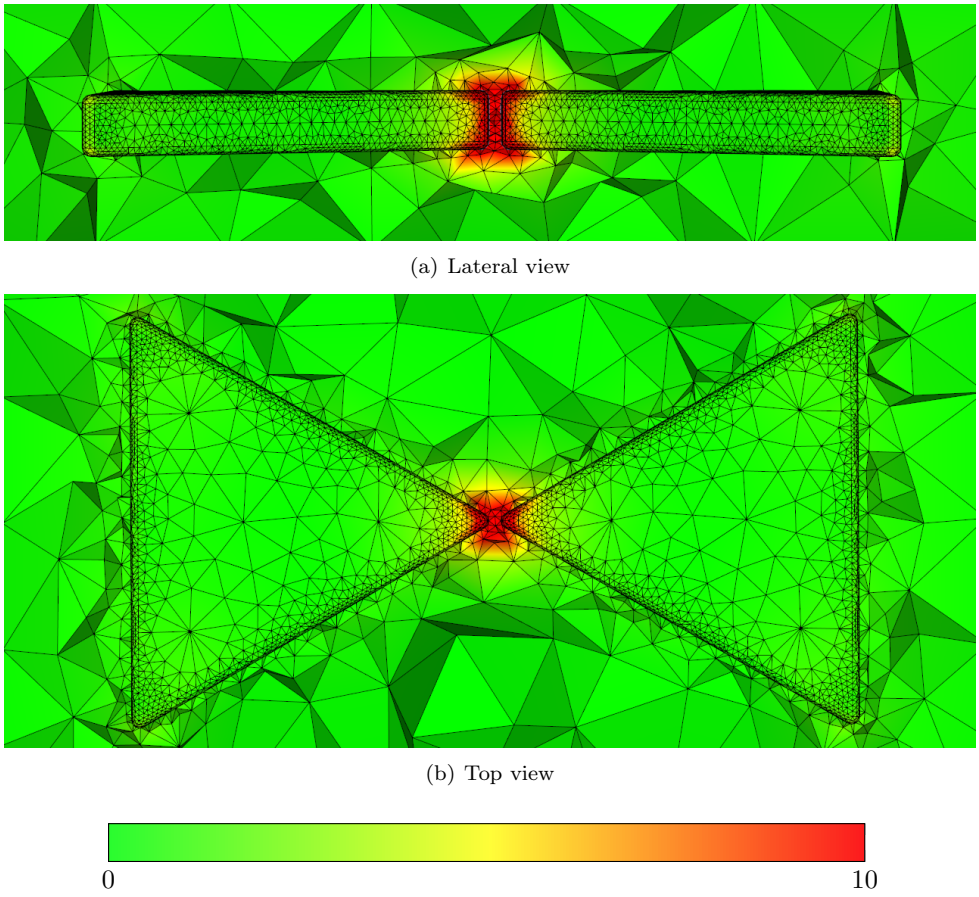
Figure 13: **Polynomial order repartition for the bowtie mesh** with respect to time-step (top), and geometrical repartition (bottom) for the $\mathbb{P}_1 - \mathbb{P}_3$ case. The white elements correspond to $\mathbb{P}_1$ approximation, while the purple cells are second-order. The remaining cells (not represented) receive third order approximation.

for all edges and tips. The considered material is gold, described by a Drude model of parameters $\varepsilon_\infty = 3.7362$, $\omega_d = 1.387 \times 10^7$ GHz and $\gamma_d = 4.515 \times 10^4$ GHz. The nanoantenna is enclosed in a TF/SF surface, and the domain is terminated by a layer of CFS-PML tetrahedra. As can be seen on figure 12, the typical setup for such computations requires very small elements (geometrical details of the nanoantenna) as well as very large ones (vacuum and PML cells), and could therefore make good use of the LA-DGTD formulation presented before. To compute the exctinction cross-section, the bowtie is illuminated from above with a wide-band plane wave of central frequency 500 THz, polarized along the major axis of the antenna. With an homogeneous polynomial order on the whole mesh, convergence is obtained for a $\mathbb{P}_3$ approximation, requiring 30 hours and 37 minutes on 16 cores. This is taken as a reference for the LA-DGTD strategy, for which $\mathbb{P}_1$ to $\mathbb{P}_3$ approximations are used. The repartition of orders with regards to the time-step is presented on figure 13, along with a visual representation of the order selection in the mesh. As expected, the first order is attributed to the small cells of the edges and tips, which are then enclosed into a second layer of $\mathbb{P}_2$ elements. All the remaining cells (not represented) receive a third order interpolation.

The computed exctinction cross sections are presented on figure 14. A single very large resonance is observed around 418 THz. As can be seen, the $\mathbb{P}_1 - \mathbb{P}_3$ solution properly fits the full $\mathbb{P}_3$ solution, with a deviation of less than 2 %. For further comparison, the full $\mathbb{P}_1$ solution is also plotted. In terms of performance, the $\mathbb{P}_1 - \mathbb{P}_3$ solution is obtained in 13 hours and 59 minutes on 16 cores, hence yielding a 2.2 speedup factor, which is lower than what was observed in section 6.1. The difference can be attributed to the lower proportion of high-order ($\mathbb{P}_3$) elements compared to low-order ones (less than 20 % of $\mathbb{P}_3$ elements here, against 40 % for the nanolens case). However, this remains an appreciable gain for a solution of similar accuracy. As an illustration, a field map of $|\mathbf{E}|$ is plotted on figure 15 for $\mathbb{P}_1 - \mathbb{P}_3$ approximation.

7. Conclusion

In this paper, the use of local polynomial approximation in the DGTD method is presented. The convergence of the algorithm is demonstrated on a standard PEC cavity case. Then, an order repartition algorithm is proposed that proved to be efficient, both for textbook and realistic nanophotonics-related cases. Although the obtained speedups are lower for realistic cases (between 2 and 2.6 for $\mathbb{P}_1 - \mathbb{P}_3$) than for academic cases (up to 4.5 for $\mathbb{P}_1 - \mathbb{P}_3$ and 6.15 for $\mathbb{P}_1 - \mathbb{P}_4$), the LA-DGTD strategy represents an interesting gain in speed for long time simulations. However, the repartition algorithm being based on the time step, it also implicitely relies on a basic knowledge of the physical behavior of the computed system (the preliminary grasp



(a) Lateral view

(b) Top view

Figure 15: **|E| field map in the nanolens device** at $t = 12.3$ fs, obtained with a $\mathbb{P}_1 - \mathbb{P}_3$ approximation. The field values are scaled to $[0, 10]$.

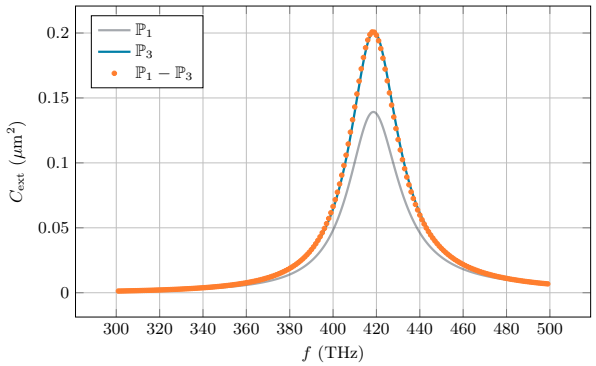


Figure 14: **Extinction cross-section of the bowtie nanoantenna** obtained with $\mathbb{P}_1$, $\mathbb{P}_3$ and $\mathbb{P}_1 - \mathbb{P}_3$ approximations. Less than 2 % of relative error is observed between full $\mathbb{P}_3$ and $\mathbb{P}_1 - \mathbb{P}_3$ computations, for a speedup factor superior to 2.

of the positions of intense fields, basically). A good remedy would consist in a coupling of the algorithm with an *a posteriori* error estimate, in order to dynamically adapt the polynomial order in the mesh.

Acknowledgements. The authors gratefully acknowledge support from the Direction Générale de l'Armement (DGA) which partially supports the doctoral thesis of Jonathan Viquerat.

References

[1] R. Ulbricht, E. Hendry, J. Shan, T. Heinz, M. Bonn, Carrier dynamics in semiconductors studied with time-resolved terahertz spectroscopy, Rev. Mod. Phys. 83 (2) (2011) 543–586.

[2] C. Wolff, R. Rodriguez-Oliveros, K. Busch, Simple magneto-optic transition metal models for timedomain simulations, Optics Express 21 (10) (2013) 12022–12037.

[3] F. J. G. de Abajo, Graphene nanophotonics, Science - Applied physics 339 (2013) 917–918.

[4] A. Moreau, C. Ciraci, D. Smith, Impact of nonlocal response on metallodielectric multilayers and optical patch antennas, Phys. Rev. B 87 (045401-1–045401-11) (2013) 6795–6820.

[5] A. Taflove, S. Hagness, Computational electrodynamics: the finite-difference time-domain method - 3rd ed., Artech House Publishers, 2005.

[6] A. Ditkowski, K. Dridi, J. Hesthaven, Convergent cartesian grid methods for Maxwell's equations in complex geometries, J. Comput. Phys. 170 (1) (2001) 39–80.

[7] J. Niegemann, M. König, K. Stannigel, K. Busch, Higher-order time-domain methods for the analysis of nano-photonic systems, Photonics and Nanostructures - Fundamentals and Applications 7 (1) (2009) 2–11. doi:http://dx.doi.org/10.1016/j.photonics.2008.08.006.

[8] K. Stannigel, M. Koenig, J. Niegemann, K. Busch, Discontinuous Galerkin time-domain computations of metallic nanostructures, Optics Express 17 (2009) 14934–14947.

[9] K. Busch, M. König, J. Niegemann, Discontinuous Galerkin methods in nanophotonics, Laser and Photonics Reviews 5 (2011) 1–37.

[10] M. König, K. Busch, J. Niegemann, The Discontinuous Galerkin Time-Domain method for Maxwells equations with anisotropic materials, Photonics and Nanostructures - Fundamentals and Applications 8 (4) (2010) 303–309. doi:http://dx.doi.org/10.1016/j.photonics.2010.04.001.

[11] C. Matyssek, J. Niegemann, W. Hergert, K. Busch, Computing electron energy loss spectra with the Discontinuous Galerkin Time-Domain method, Photonics and Nanostructures - Fundamentals and Applications 9 (4) (2011) 367–373. doi:http://dx.doi.org/10.1016/j.photonics.2011.04.003.

[12] J. Niegemann, M. König, C. Prohm, R. Diehl, K. Busch, Using curved elements in the discontinuous Galerkin time-domain approach, in: D. Chigrin (Ed.), 3rd International Workshop on Theoretical and Computational Nano-Photonics (TaCoNa-Photonics 2010), Vol. 1291 of AIP Conf. Proc., AIP, Bad Honnef, Germany, 2010, pp. 76–78.

[13] R. Diehl, K. Busch, J. Niegemann, Comparison of low-storage Runge-Kutta schemes for discontinuous Galerkin time-domain simulations of Maxwell's equations, J. Comp. Theor. Nanosc. 7 (2010) 1572.

[14] J. Niegemann, R. Diehl, K. Busch, Efficient low-storage Runge-Kutta schemes with optimized stability regions, J. Comput. Phys. 231 (2) (2012) 364–372. doi:http://dx.doi.org/10.1016/j.jcp.2011.09.003.

[15] A. Demirel, J. Niegemann, K. Busch, M. Hochbruck, Efficient multiple time-stepping algorithms of higher order, J. Comput. Phys. 285 (2015) 133–148. doi:http://dx.doi.org/10.1016/j.jcp.2015.01.018.

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

[17] J. Viquerat, S. Lanteri, C. Scheid, Theoretical and numerical analysis of local dispersion models coupled to a discontinuous galerkin time-domain method for Maxwell's equations, Tech. Rep. 8298, INRIA (2013).
URL `https://hal.inria.fr/hal-00819758`

[18] R. Léger, J. Viquerat, C. Durochat, C. Scheid, S. Lanteri, A parallel non-conforming multi-element DGTD method for the simulation of electromagnetic wave interaction with metallic nanoparticles, J. Comp. Appl. Math. 270 (2014) 330–342. doi:http://dx.doi.org/10.1016/j.cam.2013.12.042.

[19] J. Viquerat, C. Scheid, A 3D curvilinear discontinuous Galerkin time-domain solver for nanoscale light-matter interactions, J. Comp. Appl. Math. 289 (2015) 37–50. doi:http://dx.doi.org/10.1016/j.cam.2015.03.028.

[20] J. Verwer, Component splitting for semi-discrete Maxwell equations, BIT Numer. Math. 51 (2) (2011) 427–445.

[21] L. Moya, S. Descombes, S. Lanteri, Locally implicit time integration strategies in a discontinuous Galerkin method for Maxwell's equations, J. Sci. Comp. 56 (1) (2013) 190–218.
URL `http://dx.doi.org/10.1007/s10915-012-9669-5`

[22] S. Piperno, Symplectic local time stepping in non-dissipative DGTD methods applied to wave propagation problem, ESAIM: Math. Model. Num. Anal. 40 (5) (2006) 815–841.

[23] J. Diaz, M. Grote, Energy conserving explicit local time stepping for second-order wave equations, SIAM J. Sci. Comput. 31 (1985–2014) (2009) 3. doi:http://dx.doi.org/10.1137/070709414.

[24] M. Grote, T. Mitkova, Explicit local time-stepping methods for Maxwells equations, J. Comp. Appl. Math. 234 (3283–3302) (2010) 12. doi:http://dx.doi.org/10.1016/j.cam.2010.04.028.

[25] M. Grote, M. Mehlin, T. Mitkova, Runge-Kutta-based explicit local time-stepping methods for wave propagation, SIAM J. Sci. Comput. 37 (A747-A775) (2015) 2. doi:http://dx.doi.org/10.1137/140958293.

[26] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comp. 20 (1) (1998) 359–392.

[27] J. Dai, F. Cajko, I. Tsukerman, M. I. Stockman, Electrodynamics effects in plasmonic nanolenses, Phys. Rev. B 77 (2008) 115419.

[28] K. Li, M. I. Stockman, D. Bergman, Self-similar chains of metal nanospheres as an efficient nanolens, Phys. Rev. Lett. 91 (2003) 227402.

[29] P. Drude, Zur elektronentheorie der metalle, Annalen der Physik 306 (1900) 566–613.

[30] J.-P. Bérenger, Perfectly matched layer for computational electromagnetics, 1st Edition, Morgan & Claypool, 2007.
[31] N. A. Hatab, C.-H. Hsueh, A. L. Gaddis, S. Retterer, J.-H. Li, G. Eres, Z. Zhang, B. Gu, Free-standing optical gold bowtie nanoantenna with variable gap size for enhanced Raman spectroscopy, Nano Lett. 10 (2010) 4952–4955.
[32] J. Zhang, M. Irannejad, B. Cui, Bowtie nanoantenna with single-digit nanometer gap for surface-enhanced Raman scattering (SERS), Plasmonics 10 (4) (2014) 831–837.
[33] B. Grzeskiewicz, K. Ptaszynski, M. Kotkowiak, Near and far-field properties of nanoprisms with rounded edges, Plasmonics 9 (3) (2014) 607–614.

Table 3: **CPU times, memory consumption and error levels for mixed orders of approximation on locally refined meshes.** The order repartition was obtained *via* the procedure described in section 4. All simulations were run with upwind fluxes and LSRK4 time integration.

		M1	**M2**	**M3**
$\mathbb{P}_1$	CPU (s)	1.77	48.7	6030
	Mem. (MB)	14.1	22.1	107.3
	$\|\mathbf{E}-\mathbf{E}_h\|$	3.38×10^{-2}	3.59×10^{-2}	3.87×10^{-2}
$\mathbb{P}_2$	–	6.50	180	22460
		17.8	34.4	211
		3.68×10^{-3}	3.71×10^{-3}	3.82×10^{-3}
$\mathbb{P}_3$	–	21.8	611	90020
		23.9	54.6	382
		2.14×10^{-4}	2.39×10^{-4}	2.53×10^{-4}
$\mathbb{P}_4$	–	73.8	2106	228220
		32.9	84.7	635
		1.42×10^{-5}	1.76×10^{-5}	1.97×10^{-5}
$\mathbb{P}_1-\mathbb{P}_2$	CPU (s)	3.84	101	15000
	Speedup	1.69	1.78	1.50
	Mem. (MB)	17.8	34.4	211
	Tet. ratios	$0.18, 0.82$	$0.29, 0.71$	$0.33, 0.67$
	$\|\mathbf{E}-\mathbf{E}_h\|$	3.84×10^{-3}	3.71×10^{-3}	3.82×10^{-3}
$\mathbb{P}_2-\mathbb{P}_3$	–	14.3	372	40470
		1.52	1.64	2.22
		23.9	54.6	382
		$0.25, 0.75$	$0.38, 0.62$	$0.43, 0.57$
		2.38×10^{-4}	2.39×10^{-4}	2.53×10^{-4}
$\mathbb{P}_3-\mathbb{P}_4$	–	49.2	1390	130730
		1.5	1.51	1.74
		33.0	84.8	635
		$0.17, 0.83$	$0.28, 0.72$	$0.31, 0.69$
		1.42×10^{-5}	1.76×10^{-5}	1.97×10^{-5}
$\mathbb{P}_1-\mathbb{P}_2-\mathbb{P}_3$	–	–	180	19890
			3.39	4.53
			54.7	392
			$0.29, 0.31, 0.40$	$0.33, 0.38, 0.29$
			2.39×10^{-4}	2.53×10^{-4}
$\mathbb{P}_2-\mathbb{P}_3-\mathbb{P}_4$	–	–	695	64820
			3.03	3.52
			84.8	635
			$0.38, 0.22, 0.40$	$0.43, 0.27, 0.30$
			1.76×10^{-5}	1.97×10^{-5}
$\mathbb{P}_1-\mathbb{P}_2-\mathbb{P}_3-\mathbb{P}_4$	–	–	347	37130
			6.07	6.15
			84.9	636
			$0.29, 0.32, 0.13, 0.26$	$0.33, 0.38, 0.15, 0.14$
			1.80×10^{-5}	1.99×10^{-5}