

Higher-Order Discontinuous Galerkin Method for Pyramidal Elements using Orthogonal Bases

Morgane Bergot, Marc Duruflé

▶ **To cite this version:**

Morgane Bergot, Marc Duruflé. Higher-Order Discontinuous Galerkin Method for Pyramidal Elements using Orthogonal Bases. Numerical Methods for Partial Differential Equations, 2013, 29 (1), pp.144-169. 10.1002/num.21703 . hal-00547319v2

HAL Id: hal-00547319
https://hal.science/hal-00547319v2

Submitted on 13 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HIGHER-ORDER DISCONTINUOUS GALERKIN METHOD FOR PYRAMIDAL ELEMENTS USING ORTHOGONAL BASES

Morgane Bergot
Projet POems, INRIA Rocquencourt, Le Chesnay, France
Email: morgane.bergot@inria.fr
Marc Duruflé
Institut Mathématique de Bordeaux, Université Bordeaux I, Bordeaux, France
Email: marc.durufle@math.u-bordeaux1.fr
Mathematics subject classification: 65F05, 65F50, 65M60, 65Y20

Abstract

We study finite elements of arbitrarily high-order defined on pyramids for discontinuous Galerkin methods. We propose a new family of high-order pyramidal finite elements using orthogonal basis functions which can be used in hybrid meshes including hexahedra, tetrahedra, wedges and pyramids. We perform a comparison between these orthogonal functions and nodal functions for affine and non-affine elements. Different strategies for the inversion of the mass matrix are also considered and discussed. Numerical experiments are conducted for the 3-D Maxwell's equations.

Key words: pyramidal element, higher-order finite element, hybrid mesh, conformal mesh, discontinuous Galerkin method, orthogonal basis functions.

Introduction

Discontinuous Galerkin methods have been extensively studied for tetrahedral meshes (e.g. Hesthaven and Warburton [1] for Maxwell's equations). The works of Cohen and collaborators (Pernet and Ferrières [2], Duruflé [3]) have shown the higher efficiency obtained by using hexahedral meshes with this method, thanks to the tensorization of the basis functions. However, generating high quality conformal or non-conformal hexahedral meshes is not an easy task.

A first solution is to consider a non-conformal mesh with only tetrahedra and hexahedra, some quadrilateral faces connecting an hexahedron with two tetrahedra. Even though attractive, this method requires to handle non-conformal meshes and may not be the most efficient. A second solution is to allow the insertion of other types of elements, pyramids and prisms, in order to make a conformal transition between triangular and quadrilateral faces. As pyramidal elements are not very common in the Discontinuous Galerkin literature, it seemed appealing to explore the second option. This second option is the only solution studied in this paper.

Pyramidal elements have been studied by several authors in the context of continuous finite elements (Zaglmayr cited by Demkowicz [4], Nigam and Phillips [5] [6], Bergot *et al.* [7], Sherwin *et al.* [8]) but their application to discontinuous Galerkin methods has received less attention. Obviously, basis functions developed for continuous finite elements can be used for discontinuous elements as done in [7]. However, since continuity is not required, other sets of basis functions are acceptable and may have better properties. An attractive choice consists of orthogonal bases, which has been proposed by Kirby *et al.* [9] for all the types of elements,

but the proposed basis functions for pyramids generate $\mathbb{P}_r$ instead of the optimal finite element space, and we illustrate here that the use of $\mathbb{P}_r$ for pyramids provides a poor convergence for non-affine pyramids. A recent work of Gassner *et al.* [10] proposes an original approach to construct nodal discontinuous Galerkin method for hybrid meshes, since it avoids the use of a reference element, by constructing directly nodal functions generating polynomials on the real element.

We consider linear hyperbolic problems like Maxwell's equations, with an explicit time scheme like Runge-Kutta. Continuous Galerkin finite element schemes are not attractive in this context, since the mass matrix is large and costly to invert. Mass-lumped elements are well-known for hexahedra (Cohen [11]) but less so for other elements (for tetrahedra see Mulder et al. [12]) that require a large number of additional degrees of freedom and lead to a more restrictive stability condition. Furthermore, the application of a continuous Galerkin formulation is tedious because of spurious modes: it often needs regularization, as described by Costabel in [13] for Maxwell's equations. When a discontinuous Galerkin method is considered, nonetheless we get a block-diagonal mass matrix, but orthogonal tensorized basis functions can be used to get an elementary sparse mass matrix, which induces a gain in computational time when the order of approximation is large enough.

In this paper, we propose finite elements that reduce both memory and computational time as much as possible. We propose two sets of basis functions based on orthogonal tensorized basis functions: a set generating optimal finite element spaces in the sense of the convergence defined in [7] used with non-affine hexahedra, prisms, pyramids, and a set generating polynomials $\mathbb{P}_r$ used with affine elements.

The outline of our paper is as follows:

- We present the representative problem we study and its discretization in space and in time in Section 1.
- We detail the construction of the mass matrix for pyramidal elements in Section 2, when using classical nodal basis functions in Section 2.1 and orthogonal basis functions in Section 2.2. The mass-matrix is constructed and factorised only once, but a sparse structure of the matrix would speed-up the resolution phase performed at each time step. A way to obtain a diagonal mass matrix proposed by Warburton [14] is considered in Section 2.3.
- Section 3 is devoted to the construction of a fast matrix-vector product using orthogonal basis functions for all the types of elements. For pyramidal elements, its complexity is in $O(r^4)$ where r is the order of approximation, instead of $O(r^6)$ when no tensorization is availabe (e.g. nodal functions of the pyramid). Orthogonal functions are asymptotically more efficient than nodal functions, but it has been numerically found that it was more efficient for pyramids from $r \geq 3$, for wedges from $r \geq 5$ and for tetrahedra from $r \geq 10$.
- We perform numerical comparisons of the finite elements constructed to several other elements in Section 4. For affine elements, the orthogonal basis functions generating optimal finite element spaces are more efficient. A comparison of computational times shows that affine pyramids are more efficient than affine tetrahedra from order 7 on, prisms are always more efficient than tetrahedra, and non-affine pyramids are more efficient than curved tetrahedra sticked to a quadrangular face from order 4 and above. That is why considering prisms and pyramids instead of non-conformal meshes with tetrahedra is relevant

- Numerical results are provided in Section 5 for the resolution of 3-D time-domain Maxwell's equations.

1. Definitions and Presentation of the Problem

1.1. Definitions and Variational Formulation

We consider the following representative linear hyperbolic problem (e.g. Godlewski and Raviart [15])

$$M\frac{\partial u}{\partial t}+\sum_{1\leq i\leq d}A_i\frac{\partial u}{\partial x_i}+\sum_{1\leq i\leq d}B_i\frac{\partial u}{\partial x_i}=0,\qquad (M,A_i,B_i)\in\left(\mathcal{M}_{n_s}(\mathbb{R})\right)^3,\ u\in\mathbb{R}^{n_s} \tag{1.1}$$

where n_s is the number of scalar unknowns of the equation, and d is the dimension. When the system is symmetric, we have

$$B_i=A_i^*.$$

Remark 1.1. *This formulation is useful to exhibit the antisymmetry of the stiffness matrix when using centered flux and without absorbing conditions, which is essential to get the stability.*

Let Ω be an open set of $\mathbb{R}^3$, composed of n_e elements K_i

$$\Omega=\bigcup_{1\leq i\leq n_e}K_i.$$

For any element K of boundary ∂K of outward normal n, we consider a discontinuous method. For example, the Local Discontinuous Galerkin (LDG) formulation (see Hesthaven and Warburton [16] for Maxwell's equations) writes as

$$\left\{\begin{array}{l}\text{Find } u\in V \text{ such that}\\ \forall v\in V,\quad \dfrac{\mathrm{d}}{\mathrm{d}t}\displaystyle\int_K M\,u\cdot v\,dx-\int_K\sum_{1\leq i\leq d}\left(A_i\,u\cdot\frac{\partial v}{\partial x_i}-B_i\,\frac{\partial u}{\partial x_i}\cdot v\right)dx\\ \qquad\qquad\qquad\qquad+\displaystyle\int_{\partial K}(N_1\,\{u\}+N_2\,[u])\cdot v\,ds=0,\end{array}\right. \tag{1.2}$$

where $V=\left(L^2(\Omega)\right)^{n_s}$, $N_1=\displaystyle\sum_{1\leq i\leq d}A_i\,n_i$, $N_2=\displaystyle\sum_{1\leq i\leq d}B_i\,n_i$. The average $\{u\}$ is

$$\{u\}=\frac{1}{2}(u_1+u_2)$$

and $[u]$ is

$$[u]=\frac{1}{2}(u_2-u_1)+\frac{1}{2}\alpha\int_{\partial K}C(u_2-u_1)\,ds,$$

where C is a symmetric positive matrix, u_1 value of u on the element K and u_2 value of u on a neighbour element of K, and $\alpha\leq 0$. In general, we take $\alpha=-0.5$ in our experiments.

1.2. Space Discretization

Given a finite-dimensioned subspace V_h of the space V, the discrete problem reads

$$\left\{\begin{array}{l} \text{Find } u_h \in V_h \text{ such that} \\ \forall v_h \in V_h, \quad \dfrac{\mathrm{d}}{\mathrm{d}t}\displaystyle\int_K M\, u_h \cdot v_h \, dx - \int_K \sum_{1\le i\le d} \left(A_i\, u_h \cdot \frac{\partial v_h}{\partial x_i} - B_i\, \frac{\partial u_h}{\partial x_i} \cdot v_h \right) dx \\ \qquad\qquad + \displaystyle\int_{\partial K} (N_1\,\{u_h\} + N_2\,[u_h]) \cdot v_h \, ds = 0. \end{array}\right. \tag{1.3}$$

We denote by $n = n_s\, n_r$ the dimension of V_h.

To write the integrals on the reference element, the standard transformation (Ciarlet [17]) F from a reference element $\hat{K}$ to the mesh element K is used for hexahedral, tetrahedral and wedge elements. For pyramidal elements, following Bedrosian [18], we define a transformation using rational fractions.

Definition 1.1. *The transformation F from the reference pyramid $\hat{K}(\hat{x},\hat{y},\hat{z})$ taken as the unit symmetrical pyramid, centered at the origin (see Fig. 1.1) to a pyramid $K(x,y,z)$ of vertices $S_i = (x_i, y_i, z_i)$ is*

$$\begin{aligned} F = \ & \frac{1}{4}\Big[(S_1+S_2+S_3+S_4) \ + \ \hat{x}\,(-S_1+S_2+S_3-S_4) \ + \ \hat{y}\,(-S_1-S_2+S_3+S_4) \\ & + \hat{z}\,(4S_5-S_1-S_2-S_3-S_4) \ + \ \frac{\hat{x}\hat{y}}{1-\hat{z}}(S_1+S_3-S_2-S_4)\Big]. \end{aligned} \tag{1.4}$$

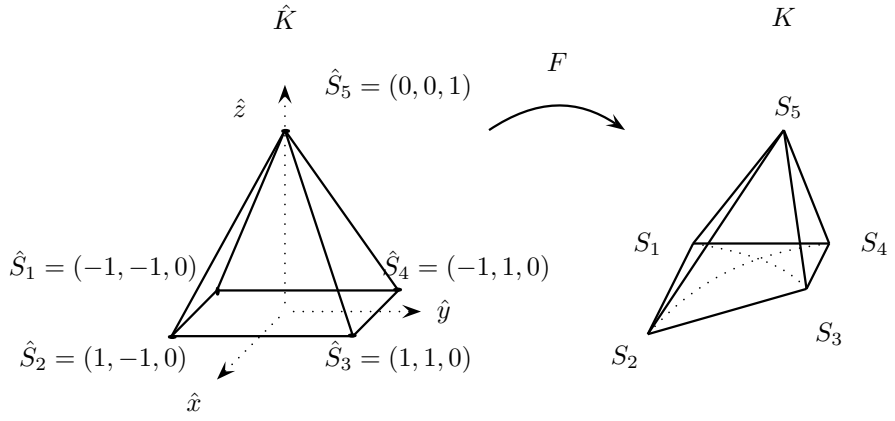


Fig. 1.1. Transformation of the reference pyramid $\hat{K}$ to the pyramid K via the transformation F

Remark 1.2. *The case of a non-invertible transformation may occur when considering a degenerated element, e.g. when the five vertices are co-planar. The characterisation of pyramids for which F is invertible remains an still open question, as for hexahedra (Duruflé* et al. *[19]). In what follows, we assume that F is always invertible.*

The finite element space V_h on Ω is given by

$$V_h = \left\{ u \in L^2(\Omega) \mid u_{|K} \in \left(\mathbb{P}_r^F(K)\right)^{n_s} \right\},$$

where $\mathbb{P}_r^F$ is the real space of order r for an element K of the mesh defined by

$$\mathbb{P}_r^F(K) = \left\{ u \mid u \circ F \in \left(\hat{P}_r(\hat{K}) \right)^{n_s} \right\}.$$

The finite element space $\hat{P}_r$ of order r on the reference element $\hat{K}$ is

- *Tetrahedron:* $\mathbb{P}_r(\hat{x}, \hat{y}, \hat{z}) = \left\{ \hat{x}^i \hat{y}^j \hat{z}^k,\ i+j+k \leq r \right\}$ of dimension $n_r = \dfrac{(r+1)(r+2)(r+3)}{6}$, and $\hat{K}$ is the unit tetrahedron $0 \leq \hat{x} + \hat{y} + \hat{z} \leq 1$;
- *Hexahedron:* $\mathbb{Q}_r(\hat{x}, \hat{y}, \hat{z}) = \left\{ \hat{x}^i \hat{y}^j \hat{z}^k,\ i, j, k \leq r \right\}$ of dimension $n_r = (r+1)^3$, and $\hat{K}$ is the unit cube $[0,1]^3$;
- *Wedge:* $\mathbb{P}_r(\hat{x}, \hat{y}) \otimes \mathbb{P}_r(\hat{z}) = \left\{ \hat{x}^i \hat{y}^j \hat{z}^k,\ i+j \leq r, k \leq r \right\}$ of dimension $n_r = \dfrac{(r+1)^2(r+2)}{2}$, and $\hat{K}$ is the unit wedge $0 \leq \hat{x} + \hat{y}, \hat{z} \leq 1$;
- *Pyramid:* $\mathbb{P}_r(\hat{x}, \hat{y}, \hat{z}) \oplus \displaystyle\sum_{0 \leq k \leq r-1} \left(\frac{\hat{x}\hat{y}}{1-\hat{z}}\right)^{r-k} \mathbb{P}_k(\hat{x}, \hat{y})$ of dimension $n_r = \dfrac{(r+1)(r+2)(2r+3)}{6}$, when $\hat{K}$ is the unit symmetrical pyramid, centered at the origin.

The finite element space is typical for tetrahedra, hexahedra and wedges (Ciarlet [17]) but is less standard in the case of a pyramidal element.

This choice of finite element space is optimal, that is by choosing this finite element space, the final error estimate is in $O(h^{r+1})$ in L^2-norm, whereas choosing any subspace included in this one leads to a convergence of at most $O(h^r)$ (see [7]). This can be seen by displaying the dispersion error (see Cohen [11] for the definition of the dispersion analysis, and how to perform it) on a periodic mesh of space step h containing non-affine pyramids, whose cell is presented on Fig. 1.2. As shows Fig. 1.3 displaying the dispersion error obtained for Maxwell's equations, using the optimal finite element space provides a dispersion error in $O(h^{2r+1})$ (see Pernet [20] for the factor $2r+1$), whereas the use of $\mathbb{P}_r$ as for tetrahedra leads to a low convergence rate.

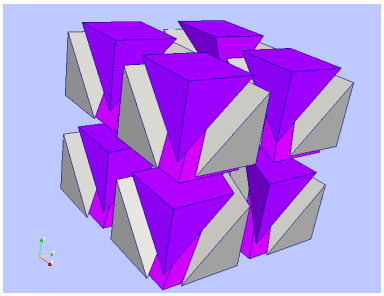


Fig. 1.2. Periodic pattern for the hybrid case, with distorted pyramids (dark) and tetrahedra (light)

Definition 1.2. *Let $(\varphi_i)_{\leq i \leq n_r}$ be a base of V_h, let M_h the mass matrix for the pyramid K, defined by*

$$(M_h)_{i,j} = \int_K M\, \varphi_i \cdot \varphi_j \, dx \tag{1.5}$$

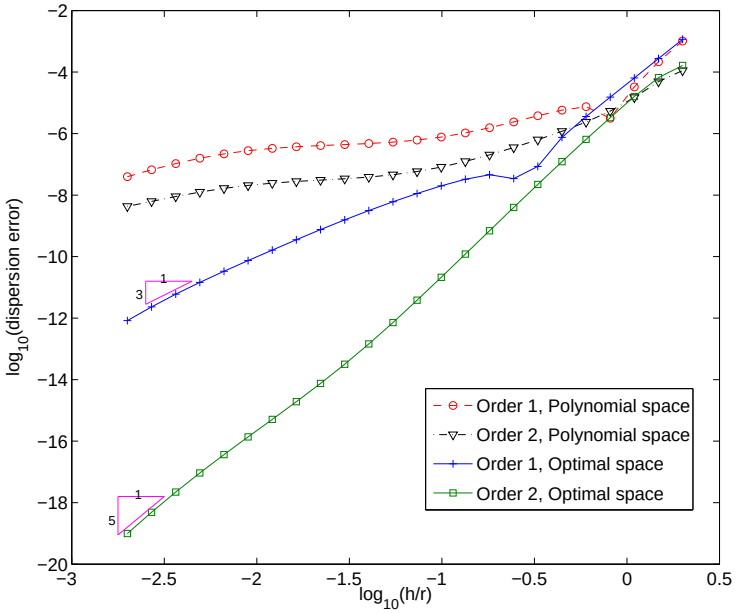


Fig. 1.3. Dispersion error for a periodic mesh with non-affine pyramids for Maxwell's equation

the stiffness matrix R_h *such that*

$$(R_h)_{i,j} = \int_K \sum_{1\leq k\leq d} \left(A_k \frac{\partial \varphi_i}{\partial x_k} \cdot \varphi_j - B_k \, \varphi_i \cdot \frac{\partial \varphi_j}{\partial x_k} \right) dx \tag{1.6}$$

and the flux matrix S_h *defined by*

$$(S_h)_{i,j} = \int_{\partial K} \sum_{1\leq k\leq d} \left(N_1 \left\{ \varphi_j \right\} + N_2 \left[\varphi_j \right] \right) \cdot \varphi_i \, ds. \tag{1.7}$$

The space discretization finally writes as

$$\frac{\mathrm{d}}{\mathrm{d}t} M_h U - R_h U + S_h U = 0. \tag{1.8}$$

1.3. Time Discretization

Using any explicit time scheme, for example the low-storage Runge-Kutta scheme of order 4 (Carpenter and Kennedy [21]), the time discretization writes as

$$\begin{aligned} &U^{n+1} = U^n \\ &\rho = U^n \\ &\text{for } i = 1 \text{ to } 5 \\ &\qquad \rho = \alpha_i \rho + \Delta t (M_h)^{-1} (R_h - S_h)(U^{n+1}) \\ &\qquad U^{n+1} = U^{n+1} + \beta_i \rho \\ &\text{end for} \end{aligned} \tag{1.9}$$

For each time-step, we then have to

- compute the matrix-vector products $R_h U^n$ and $S_h U^n$;
- solve a linear system $(M_h)\, X = Y$.

We first present an efficient technique to construct M_h and solve the linear system for pyramidal elements.

2. Mass Matrix

2.1. Dense Mass Matrix with Nodal Basis Functions

Using the H^1-conforming transformation (Monk [22]) of the basis functions from any element K of the mesh to the reference element $\hat{K}$

$$\hat{\varphi}_i = \varphi_i \circ F^{-1}, \tag{2.1}$$

the mass matrix writes as

$$(M_h)_{i,j} = \int_K M\, \varphi_i\, \varphi_j\; dx\, dy\, dz = \int_{\hat{K}} M\, |DF|\, \hat{\varphi}_i\, \hat{\varphi}_j\; d\hat{x}\, d\hat{y}\, d\hat{z}. \tag{2.2}$$

The nodal basis functions on $\hat{K}$ are obtained by inverting a Vandermonde system as explained in [7]. The resulting mass matrix is dense and has no particular property.

2.2. Sparse Mass Matrix with Orthogonal Basis Functions

2.2.1. Orthogonal Basis Functions

To have a sparser mass matrix, we consider orthogonal basis functions. We denote by $P_m^{i,j}(x)$ the orthonomalized Jacobi polynomial of order m, orthogonal for the weight $(1-x)^i(1+x)^j$, and ξ_j^G the points of Gauss-Legendre on $[0,1]$ (cf Hammer, Marlowe and Stroud [23]).

Proposition 2.1. *The following set of basis functions is an orthogonal base of $\hat{P}_r$*

- ***Hexahedron***

$$\hat{\varphi}_{i_1}^G(\hat{x})\, \hat{\varphi}_{i_2}^G(\hat{y})\, \hat{\varphi}_{i_3}^G(\hat{z}), \quad 0 \le i_1, i_2, i_3 \le r,$$

where

$$\hat{\varphi}_i^G(\hat{x}) = \frac{\displaystyle\prod_{j \neq i} \hat{x} - \xi_j^G}{\displaystyle\prod_{j \neq i}, \xi_i^G - \xi_j^G},$$

- ***Wedge***

$$P_{i_1}^{0,0}\left(\frac{2\hat{x}}{1-\hat{y}} - 1\right) (1-\hat{y})^{i_1} P_{i_2}^{2i_1+1,0}\,(2\hat{y}-1)\, \varphi_{i_3}^G(\hat{z}), \quad 0 \le i_1 + i_2, i_3 \le r,$$

- ***Pyramid***

$$P_{i_1}^{0,0}\left(\frac{\hat{x}}{1-\hat{z}}\right) P_{i_2}^{0,0}\left(\frac{\hat{y}}{1-\hat{z}}\right) (1-\hat{z})^{max(i_1,i_2)} P_{i_3}^{2max(i_1,i_2)+2,0}(2\hat{z}-1),$$
$$0 \le i_1, i_2 \le r,\, 0 \le i_3 \le r - max(i_1, i_2),$$

- ***Tetrahedron***

$$P_{i_1}^{0,0}\left(\frac{2\hat{x}}{1-\hat{y}-\hat{z}}-1\right)P_{i_2}^{2i_1+1,0}\left(\frac{2\hat{y}}{1-\hat{z}}-1\right)(1-\hat{y}-\hat{z})^{i_1}P_{i_3}^{2(i_1+i_2)+2,0}(2\hat{z}-1)\,(1-\hat{z})^{i_2},$$
$$0\le i_1+i_2+i_3\le r.$$

Proof. See Proposition 1.10 in [7] for pyramidal elements. The proof is similar for the other types of elements.

When the element is affine, we can use $\hat{P}_r=\mathbb{P}_r$ (see Theorem 1.4 in [7]).

Proposition 2.2. *The following set of basis functions is an orthogonal base of* $\mathbb{P}_r$ *For* $0\le i_1+i_2+i_3\le r$,

- ***Hexahedron***

$$P_{i_1}^{0,0}(2\,\hat{x}-1)P_{i_2}^{0,0}(2\,\hat{y}-1)P_{i_3}^{0,0}(2\,\hat{z}-1),$$

- ***Wedge***

$$P_{i_1}^{0,0}\left(\frac{2\hat{x}}{1-\hat{y}}-1\right)(1-\hat{y})^{i_1}P_{i_2}^{2i_1+1,0}(2\hat{y}-1)\,\varphi_{i_3}^{G}(\hat{z}),$$

- ***Pyramid***

$$P_{i_1}^{0,0}\left(\frac{\hat{x}}{1-\hat{z}}\right)P_{i_2}^{0,0}\left(\frac{\hat{y}}{1-\hat{z}}\right)(1-\hat{z})^{i_1+i_2}P_{i_3}^{2(i_1+i_2)+2,0}(2\hat{z}-1),$$

- ***Tetrahedron***

$$P_{i_1}^{0,0}\left(\frac{2\hat{x}}{1-\hat{y}-\hat{z}}-1\right)P_{i_2}^{2i_1+1,0}\left(\frac{2\hat{y}}{1-\hat{z}}-1\right)(1-\hat{y}-\hat{z})^{i_1}P_{i_3}^{2(i_1+i_2)+2,0}(2\hat{z}-1)\,(1-\hat{z})^{i_2}.$$

Proof. The proof is similar to the proof of Proposition 2.1.

2.2.2. Fast Algorithm for Pyramidal Elements

We use the orthogonal base of $\hat{P}_r$ defined in Proposition 2.1 as set of basis functions. To make the computation easier, we write the integrals on the unit cube $\widetilde{Q}$ by using the following transformation T from the unit cube $\widetilde{Q}$ on the reference pyramid $\hat{K}$

$$T:\begin{cases}\hat{x}=(1-\widetilde{z})(2\widetilde{x}-1)\\ \hat{y}=(1-\widetilde{z})(2\widetilde{y}-1)\\ \hat{z}=\widetilde{z}.\end{cases}\tag{2.3}$$

For any function f, we denote

$$\widetilde{f}(\widetilde{x},\widetilde{y},\widetilde{z})=\hat{f}(\hat{x},\hat{y},\hat{z}),$$

and the change of variable provides

$$\int_{\hat{K}}\hat{f}(\hat{x},\hat{y},\hat{z})\,d\hat{x}\,d\hat{y}\,d\hat{z}=\int_{\widetilde{Q}}4\,\widetilde{f}(\widetilde{x},\widetilde{y},\widetilde{z})\,(1-\widetilde{z})^2\,d\widetilde{x}\,d\widetilde{y}\,d\widetilde{z}.\tag{2.4}$$

The mass matrix written on the unit cube $\widetilde{Q}$ is then

$$(M_h)_{i,j}=4\int_{\widetilde{Q}}M\,\widetilde{|DF|}\,\widetilde{\varphi}_i\,\widetilde{\varphi}_j\,(1-\widetilde{z})^2\,d\widetilde{x}\,d\widetilde{y}\,d\widetilde{z},$$

where the jacobian $|DF|$ of F on $\widetilde{Q}$ can be written as (see Lemma 3.5 in [7])

$$\widetilde{|DF|} = A + B_1(2\widetilde{x}-1) + B_2(2\widetilde{y}-1) + C(2\widetilde{x}-1)(2\widetilde{y}-1).$$

We recall the following property of Jacobi polynomial (Szegö [24])

Property 2.1.

$$tP_k^{i,j}(t) = \gamma_k^{i,j}P_{k+1}^{i,j}(t) + \alpha_k^{i,j}P_k^{i,j}(t) + \beta_k^{i,j}P_{k-1}^{i,j}(t), \tag{2.5}$$

where

$$\alpha_k^{i,j} = -a_{i,j,k}\frac{(2k+i+j)(i^2-j^2)}{2k+i+j} \qquad \beta_k^{i,j} = b_{i,j,k}\frac{2(k+i)(k+j)(2k+i+j+2)}{2k+i+j}$$
$$\gamma_k^{i,j} = c_{i,j,k}2(k+1)(k+i+j+1)$$

and $a_{i,j,k}$, $b_{i,j,k}$ and $c_{i,j,k}$ are coefficients due to orthonormalisation of Jacobi polynomial $P_k^{i,j}$.

Using Property 2.1, we decompose the mass matrix for all $i = (i_1, i_2, i_3)$, $1 \le i \le n_r$ and for all j_3, $0 \le j_3 \le r$

- Term in A:

$$\int_{\widetilde{Q}} \widetilde{\varphi}_i\widetilde{\varphi}_j\ (1-\widetilde{z})^2\ d\widetilde{x}\,d\widetilde{y}\,d\widetilde{z} = \underbrace{\int_0^1 P_{i_1}^{0,0}(2\widetilde{x}-1)P_{j_1}^{0,0}(2\widetilde{x}-1)\ d\widetilde{x}}_{\delta_{i_1j_1}} \underbrace{\int_0^1 P_{i_2}^{0,0}(2\widetilde{y}-1)P_{j_2}^{0,0}(2\widetilde{y}-1)\ d\widetilde{y}}_{\delta_{i_2j_2}} \underbrace{\int_0^1 (1-\hat{z})^{max(i_1,i_2)+max(j_1,j_2)+2}P_{i_3}^{2max(i_1,i_2)+2,0}(2\widetilde{z}-1)P_{j_3}^{2max(j_1,j_2)+2,0}(2\widetilde{z}-1)\ d\widetilde{z}}_{\delta_{i_3j_3}};$$

- Term in B_1:

$$\int_{\widetilde{Q}} \widetilde{\varphi}_i\widetilde{\varphi}_j\ (2\widetilde{x}-1)(1-\widetilde{z})^2\ d\widetilde{x}\,d\widetilde{y}\,d\widetilde{z} = \underbrace{\int_0^1 (2\widetilde{x}-1)P_{i_1}^{0,0}(2\widetilde{x}-1)P_{j_1}^{0,0}(2\widetilde{x}-1)\ d\widetilde{x}}_{\gamma_{i_1}^{0,0}\delta_{i_1+1j_1}+\beta_{i_1}^{0,0}\delta_{i_1-1j_1}} \underbrace{\int_0^1 P_{i_2}^{0,0}(2\widetilde{y}-1)P_{j_2}^{0,0}(2\widetilde{y}-1)\ d\widetilde{y}}_{\delta_{i_2j_2}} \underbrace{\int_0^1 (1-\hat{z})^{max(i_1,i_2)+max(j_1,j_2)+2}P_{i_3}^{2max(i_1,i_2)+2,0}(2\widetilde{z}-1)P_{j_3}^{2max(j_1,j_2)+2,0}(2\widetilde{z}-1)\ d\widetilde{z}}_{\mathfrak{C}_{i_1,j_1}^{i_2,j_2}(i_3,j_3)};$$

- Terms in B_2 and C: treated in a similar manner to B_1

The mass matrix is finally computed as follows

- $M_h[i,i] = A$
- $M_h[i,(i_1+1,i_2,j_3)] = B_1\,\gamma_{i_1}^{0,0}\,\mathfrak{C}_{i_1,i_1+1}^{i_2,i_2}(i_3,j_3)$
- $M_h[i,(i_1-1,i_2,j_3)] = B_1\,\beta_{i_1}^{0,0}\,\mathfrak{C}_{i_1,i_1-1}^{i_2,i_2}(i_3,j_3)$
- $M_h[i,(i_1,i_2+1,j_3)] = B_2\,\gamma_{i_2}^{0,0}\mathfrak{C}_{i_1,i_1}^{i_2,i_2+1}(i_3,j_3)$
- $M_h[i,(i_1,i_2-1,j_3)] = B_2\,\beta_{i_2}^{0,0}\mathfrak{C}_{i_1,i_1}^{i_2,i_2-1}(i_3,j_3)$
- $M_h[i,(i_1+1,i_2+1,j_3)] = C\,\gamma_{i_1}^{0,0}\,\mathfrak{C}_{i_1,i_1+1}^{i_2,i_2}(i_3,j_3)\,\gamma_{i_2}^{0,0}\,\mathfrak{C}_{i_1,i_1}^{i_2,i_2+1}(i_3,j_3)$
- $M_h[i,(i_1+1,i_2-1,j_3)] = C\,\gamma_{i_1}^{0,0}\,\mathfrak{C}_{i_1,i_1+1}^{i_2,i_2}(i_3,j_3)\,\beta_{i_2}^{0,0}\,\mathfrak{C}_{i_1,i_1}^{i_2,i_2-1}(i_3,j_3)$
- $M_h[i,(i_1-1,i_2+1,j_3)] = C\,\beta_{i_1}^{0,0}\,\mathfrak{C}_{i_1,i_1-1}^{i_2,i_2}(i_3,j_3)\,\gamma_{i_2}^{0,0}\,\mathfrak{C}_{i_1,i_1}^{i_2,i_2+1}(i_3,j_3)$
- $M_h[i,(i_1-1,i_2-1,j_3)] = C\,\beta_{i_1}^{0,0}\,\mathfrak{C}_{i_1,i_1-1}^{i_2,i_2}(i_3,j_3)\,\beta_{i_2}^{0,0}\,\mathfrak{C}_{i_1,i_1}^{i_2,i_2-1}(i_3,j_3)$

Using orthogonal basis functions, the mass matrix contains $O(r^4)$ non-zero entries instead of $O(r^6)$, which will be usefull for the matrix-vector product. Integrals $\mathfrak{C}_{i_1,j_1}^{i_2,j_2}(i_3,j_3)$ are precomputed, so that the cost of computation is finally in $O(r^4)$.

2.2.3. Mass Matrix for the Other Elements

For hexahedral elements, the mass matrix is diagonal thanks to the mass-lumping. For tetrahedral elements, as the jacobian is constant, the mass matrix is also diagonal when using orthogonal basis functions.

For wedge elements, as we have tensorisation and mass-lumping in z, the mass matrix is already block diagonal on each element, even for nodal elements for which the number of non-zero entries is in $O(r^5)$. Using orthogonal basis functions make each block sparser, but the gain is far less spectacular than for pyramidal elements. The same algorithm as for pyramidal elements could be used for wedge elements to speed-up the computation, but is not presented here.

2.3. Diagonal Mass Matrix with Warburton's Trick

In the discontinuous case, the resolution of the linear system can be avoided by using the following non-H^1-conforming transformation introduced by Warburton [14]

$$\hat{\varphi}_i = \frac{1}{\sqrt{|DF|}}\,\varphi_i \circ F^{-1}. \tag{2.6}$$

Using this transformation on the reference element, the mass matrix writes as

$$(M_h)_{i,j} = \int_K \varphi_i \cdot \varphi_j dx = \int_{\hat{K}} |DF| \frac{\hat{\varphi}_i}{\sqrt{|DF|}} \cdot \frac{\hat{\varphi}_j}{\sqrt{|DF|}} dx = \int_{\hat{K}} \hat{\varphi}_i \cdot \hat{\varphi}_j dx,$$

that is the mass matrix is independent of the geometry. If orthonormal basis functions are used, the matrix is equal to the identity.

This method was originally intended by Warburton to handle efficiently curvilinear tetrahedra, when the jacobian $|DF|$ tends to a constant. In that case, the method performs very well.Here we apply this technique beyond its area of application. As this transformation obviously does not preserve the constants, the method will probably not be consistent in this case.

3. Fast Matrix-Vector Product

3.1. Introduction

In the time discretization (see algorithm 1.9), a needed step consists of performing the following matrix-vector product

$$y^n = (R_h - S_h)\, U^n$$

Storing the matrix $R_h - S_h$ as a sparse matrix and performing a standard matrix-vector product would be an expensive solution as the memory required to store such a large matrix may become too important, especially when a high order is used.

Since inside each element (except hexahedron), all the degrees of freedom are interacting with each other, i.e.

$$\int_K \frac{\partial \varphi_k}{\partial x_i}\, \varphi_j \, dx \neq 0, \quad \forall j, k$$

the number of non-zero entries in matrix $R_h - S_h$ is equal to $O(n_e r^6)$ where n_e is the number of elements in the mesh and r the order of approximation. The computational time required for the matrix-vector product would therefore be in $O(n_e r^6)$ too if the matrix $R_h - S_h$ is stored.

Another solution, well-known for tetrahedra (Hesthaven [1]) consists in performing the matrix-vector product without storing the matrix. For nodal basis functions, the computational time will remain in $O(n_e r^6)$, whereas for orthogonal functions, the tensorization of basis functions induce fast algorithms in $O(n_e r^4)$ as pointed out in Warburton's thesis [25].

In the following section, we detail an efficient algorithm for the discontinuous Galerkin formulation used in this paper.

3.2. H^1-conforming Transformation

3.2.1. General Method

We are considering the following matrix vector product

$$y_j = \int_K \sum_{1 \leq i \leq d} \left(A_i\, U \cdot \frac{\partial \varphi_j}{\partial x_i} - B_i \frac{\partial U}{\partial x_i} \cdot \varphi_j \right) dx - \int_{\partial K} \left(N_1 \{U\} + N_2 [U] \right) \cdot \varphi_j \, ds$$

We use F^{-1} to transform any element K of the mesh into the reference element $\hat{K}$, and T defined by equation 2.3 to transform the reference pyramid $\hat{K}$ into the unit cube $\widetilde{Q}$. Let us denote by $G = F \circ T$, the transformation from the pyramid K into the unit cube $\widetilde{Q}$. We also use a transformation g^{-1} from a face ∂K to a reference face $\partial \widetilde{Q}$. As

$$\widetilde{u} = \sum_k \widetilde{u}_k \, \widetilde{\varphi}_k,$$

y_j finally writes as

$$\begin{aligned} y_j = & \int_{\widetilde{Q}} \widetilde{|DG|} \sum_{1\le i\le d} \sum_{1\le l\le d} \left(A_i \widetilde{u} \cdot (\widetilde{DG^{-1}})_{l,i} \frac{\partial \widetilde{\varphi}_j}{\partial \widetilde{x}_l} - B_i \, (\widetilde{DG}^{-1})_{l,i} \frac{\partial \widetilde{u}}{\partial \widetilde{x}_l} \cdot \widetilde{\varphi}_j \right) d\widetilde{x}\, d\widetilde{y}\, d\widetilde{z} \\ & - \int_{\partial \widetilde{Q}} \widetilde{|Dg|} \, (N_1 \{\widetilde{u}\} + N_2 [\widetilde{u}]) \cdot \widetilde{\varphi}_j \, d\widetilde{s}. \end{aligned}$$

Volume integrals are evaluated with quadrature formulas (ω_m, ξ_m), suited for the cube $\widetilde{Q}$, whereas surface integrals are evaluated with quadrature formulas (ω'_n, ξ'_n) suited for the faces of cube $\partial\widetilde{Q}$. Let us define

$$\bar{U} = -\left(N_1 \{\widetilde{u}(\xi'_n)\} + N_2 [\widetilde{u}(\xi'_n)]\right).$$

We finally write

$$\begin{aligned} y_j = & \sum_m \omega_m \, \widetilde{|DG|}(\xi_m) \sum_{1\le i\le d} \sum_{1\le l\le d} \left((\widetilde{DG}^{-1})_{l,i} \, A_i \widetilde{u} \cdot \frac{\partial \widetilde{\varphi}_j}{\partial \widetilde{x}_l} - B_i \, (\widetilde{DG}^{-1})_{l,i} \frac{\partial \widetilde{u}}{\partial \widetilde{x}_l} \cdot \widetilde{\varphi}_j \right)(\xi_m) \\ & + \sum_n \omega'_n \, \widetilde{|Dg|}(\xi'_n) \, \bar{U} \cdot \widetilde{\varphi}_j(\xi'_n). \end{aligned}$$

We decompose this matrix-vector into several stages.

For volume integrals :

1. Evaluate

$$v_m = \widetilde{u}(\xi_m) = \sum_k \widetilde{u}_k \, \widetilde{\varphi}_k(\xi_m), \qquad dv^l_m = \frac{\partial \widetilde{u}}{\partial \widetilde{x}_l}(\xi_m) = \sum_k \widetilde{u}_k \, \frac{\partial \widetilde{\varphi}_k}{\partial \widetilde{x}_l}(\xi_m)$$

2. Apply geometry and physics coefficients

$$v^{1,l}_m = \sum_{1\le i\le d} \omega_m \, \widetilde{|DG|}(\xi_m) \, \widetilde{DG}^{-1}_{l,i} \, A_i v_m, \qquad v^2_m = -\sum_{1\le i,l\le d} \omega_m \, \widetilde{|DG|}(\xi_m) \, B_i \, \widetilde{DG}^{-1}_{l,i} \, dv^l_m$$

3. Evaluate

$$w^1_j = \sum_m \sum_{1\le l\le d} v^{1,l}_m \cdot \frac{\partial \widetilde{\varphi}_j}{\partial \widetilde{x}_l}(\xi_m), \qquad w^2_j = \sum_m v^2_m \cdot \widetilde{\varphi}_j(\xi_m)$$

For surface integral :

1. Evaluate

$$s_n = \sum_k u_k \, \widetilde{\varphi}_k(\xi'_n)$$

2. Apply geometry and physics coefficients to get

$$\begin{aligned} \bar{s}_n &= -\left(N_1 \{s_n\} + N_2 [s_n]\right) \\ s^1_n &= \omega'_n \, \widetilde{|Dg|}(\xi'_n) \, \bar{s}_n \end{aligned}$$

3. Evaluate

$$w_j^3 = \sum_n s_n^1 \cdot \widetilde{\varphi}_j(\xi'_n)$$

Get the result vector :

$$y_j = w_j^1 + w_j^2 + w_j^3$$

We consider tensorized quadrature points

$$\xi_m = (\xi_{m_1}, \xi_{m_2}, \xi_{m_3})$$

and tensorized basis functions

$$\varphi_j(x, y, z) = \widetilde{\varphi}_{j_1}(\widetilde{x})\, \widetilde{\varphi}_{j_2}^{j_1}(\widetilde{y})\, \widetilde{\varphi}_{j_3}^{j_1,j_2}(\widetilde{z})$$

We now detail how the different stages containing sums can be decomposed because of tensorization. By performing computations along each coordinate $\widetilde{x}$, $\widetilde{y}$, $\widetilde{z}$, we are able to have sums with only $r+1$ terms, instead of $(r+1)^3$ terms if basis functions and quadrature points do not exhibit any tensorization.

3.2.2. Computation of Volume Integrals

1. For $m = (m_1, m_2, m_3)$, we want to evaluate

$$v_m = \sum_{k_1,k_2,k_3} \widetilde{\varphi}_{k_1}(\xi_{m_1})\, \widetilde{\varphi}_{k_2}^{k_1}(\xi_{m_2})\, \widetilde{\varphi}_{k_3}^{k_1,k_2}(\xi_{m_3})\, u_{k_1,k_2,k_3}.$$

We notice $\widetilde{\varphi}_{j_3}^{j_1,j_2}$ when the basis function depends on j_1 and j_2. This triple sum is split into three single sums

$$\begin{aligned} u^1_{k_1,k_2,m_3} &= \sum_{k_3} \widetilde{\varphi}_{k_3}^{k_1,k_2}(\xi_{m_3})\, u_{k_1,k_2,k_3} \\ u^2_{k_1,m_2,m_3} &= \sum_{k_2} \widetilde{\varphi}_{k_2}^{k_1}(\xi_{m_2})\, u^1_{k_1,k_2,m_3} \\ v_{m_1,m_2,m_3} &= \sum_{k_1} \widetilde{\varphi}_{k_1}(\xi_{m_1})\, u^2_{k_1,m_2,m_3} \end{aligned}$$

Remark 3.1. *We observe that the dependency has to be "opposite" between φ_j and ξ_m, otherwise the tensorized structure of quadrature points and the basis functions can not be exploited to get fast algorithms. Semi-tensorized quadrature point could also be considered*

$$\xi_m = (\xi_{m_1}^{m_2,m_3}, \xi_{m_2}^{m_3}, \xi_{m_3}).$$

The three sums involve only $O(r)$ terms and are computed for $O(r^3)$ values, leading to a cost in $O(r^4)$. Each of these sums can be interpreted as a matrix vector product, i.e.

$$U^1 = C_1\, U, \qquad U^2 = C_2\, U^1, \qquad V = C_3\, U^2$$

We have found a factorization of matrix C

$$C_{m,k} = \widetilde{\varphi}_k(\xi_m)$$

which is

$$C = C_3\, C_2\, C_1$$

While matrix C is dense, the matrices C_3, C_2 and C_1 are sparse. Matrices C_3, C_2, C_1 are independent from the geometry, and are precomputed once for each reference element. Hence, we have

$$V = C\, U$$

For $n = n_1, n_2, n_3$, we now want to evaluate

$$dv_n^l(\xi_n) = \sum_k \widetilde{u}_k\, \frac{\partial \widetilde{\varphi}_k}{\partial \widetilde{x}_l}(\xi_n)$$

Since for all elements, we have the inclusion $\widetilde{V}_r \subset \mathbb{Q}_r$, we can write

$$\widetilde{u}(x,y,z) = \sum_k \widetilde{u}_k\, \widetilde{\varphi}_k(\widetilde{x},\widetilde{y},\widetilde{z}) = \sum_{m_1,m_2,m_3} v_{m_1,m_2,m_3}\, \psi_{m_1}(\widetilde{x})\, \psi_{m_2}(\widetilde{y})\, \psi_{m_3}(\widetilde{z})$$

where ψ_{m_1}, ψ_{m_2} , ψ_{m_3} are Lagrange interpolation basis function associated respectively with points ξ_{m_1}, ξ_{m_2} and ξ_{m_3}

$$\psi_{m_1}(\widetilde{x}) = \frac{\displaystyle\prod_{n_1 \neq m_1} \widetilde{x} - \xi_{n_1}}{\displaystyle\prod_{n_1 \neq m_1} \xi_{m_1} - \xi_{n_1}} \qquad \psi_{m_2}(\widetilde{y}) = \frac{\displaystyle\prod_{n_2 \neq m_2} \widetilde{y} - \xi_{n_2}}{\displaystyle\prod_{n_2 \neq m_2} \xi_{m_2} - \xi_{n_2}} \qquad \psi_{m_3}(\widetilde{z}) = \frac{\displaystyle\prod_{n_3 \neq m_3} \widetilde{z} - \xi_{n_3}}{\displaystyle\prod_{n_3 \neq m_3} \xi_{m_3} - \xi_{n_3}}$$

We have

$$\widetilde{\nabla} \psi_{m_1}(\widetilde{x})\, \psi_{m_2}(\widetilde{y})\, \psi_{m_3}(\widetilde{z}) \;=\; \left| \begin{array}{l} \displaystyle\sum_{m_1,m_2,m_3} v_{m_1,m_2,m_3}\, \frac{d\psi_{m_1}}{d\widetilde{x}}(\widetilde{x})\; \psi_{m_2}(\widetilde{y})\; \psi_{m_3}(\widetilde{z}) \\ \displaystyle\sum_{m_1,m_2,m_3} v_{m_1,m_2,m_3}\, \psi_{m_1}(\widetilde{x})\; \frac{d\psi_{m_2}}{d\widetilde{y}}(\widetilde{y})\; \psi_{m_3}(\widetilde{z}) \\ \displaystyle\sum_{m_1,m_2,m_3} v_{m_1,m_2,m_3}\, \psi_{m_1}(\widetilde{x})\; \psi_{m_2}(\widetilde{y})\; \frac{d\psi_{m_3}}{d\widetilde{z}}(\widetilde{z}) \end{array} \right.$$

Since we have

$$\psi_{m_1}(\xi^{n_1}) = \delta_{m_1,n_1}, \quad \psi_{m_2}(\xi^{n_2}) = \delta_{m_2,n_2}, \quad \psi_{m_3}(\xi^{n_3}) = \delta_{m_3,n_3}$$

Triple sums over m_1, m_2, m_3 are reduced to single sums

$$\begin{aligned} \left(\frac{\partial v}{\partial \widetilde{x}}\right)_{n_1,n_2,n_3} &= \sum_{m_1} \frac{d\psi^{m_1}}{d\widetilde{x}}(\xi_{n_1})\, v_{m_1,n_2,n_3} \\ \left(\frac{\partial v}{\partial \widetilde{y}}\right)_{n_1,n_2,n_3} &= \sum_{m_2} \frac{d\psi^{m_2}}{d\widetilde{y}}(\xi_{n_2})\, v_{n_1,m_2,n_3} \\ \left(\frac{\partial v}{\partial \widetilde{z}}\right)_{n_1,n_2,n_3} &= \sum_{m_3} \frac{d\psi^{m_3}}{d\widetilde{z}}(\xi_{n_3})\, v_{n_1,n_2,m_3} \end{aligned}$$

These operations can be viewed as matrix vector product

$$dV = RV$$

where the matrix R is sparse and independent from the geometry, and can be precomputed once for each reference element.

Eventually, we have exhibited the following factorization

$$dV = R\,C\,U.$$

2. We compute

$$v_m^{1,l} = \sum_{1\le i\le d} \omega_m\, \widetilde{|DG|}(\xi_m)\, \widetilde{DG}_{l,i}^{-1}\, A_i v_m, \qquad v_m^2 = \sum_{1\le i\le d} \omega_m\, \widetilde{|DG|}(\xi_m)\, B_i \,(\sum_{1\le l\le d} \widetilde{DG}_{l,i}^{-1}\, dv_m^l)$$

The complexity of this operation is in $O(r^3)$, and can be viewed as a matrix-vector product

$$V^1 = A\,V, \qquad V^2 = B\,dV$$

where matrices A and B are block-diagonal, each block being related to a quadrature point, and depend on the geometry.

3. We are interested in the stage

$$w_j^1 = \sum_{m,l} v_m^{1,l} \cdot \frac{\partial \widetilde{\varphi}_j}{\partial \widetilde{x}_l}(\xi_m)$$

which is the transpose operation to the computation of derivative of basis functions on quadrature points. Therefore, we have

$$W^1 = C^*\,R^*\,V^1$$

We are now interested in the step

$$w_j^2 = \sum_m v_m^2 \cdot \widetilde{\varphi}_j(\xi_m)$$

which can be interpreted as

$$W^2 = C^*\,V^2$$

that is

$$W^2 = C_1^*\,C_2^*\,C_3^*\,V^2$$

Remark 3.2. *Using the basis functions of Equation 2.1, we have $C = Id$, which makes computations faster for hexahedra. That is why we want to have high percentage of hexahedra in a mesh.*

3.2.3. Computation of Surface Integrals

1. We are interested in the stage

$$s_p = \widetilde{u}(\xi'_p) = \sum_k \widetilde{u}_k\, \widetilde{\varphi}_k(\xi'_p)$$

Since we are considering faces of the cube, we have three families of quadrature points

$$(\delta, \xi'_{p_2}, \xi'_{p_3}), \qquad (\xi'_{p_1}, \delta, \xi'_{p_3}), \qquad (\xi'_{p_1}, \xi'_{p_2}, \delta)$$

where δ is equal to 0 or 1. The starting point is to consider the expansion with basis functions of the cube, ie

$$\widetilde{u}(\widetilde{x},\widetilde{y},\widetilde{z}) = \sum_{m_1,m_2,m_3} v_{m_1,m_2,m_3}\,\psi_{m_1}(\widetilde{x})\,\psi_{m_2}(\widetilde{y})\,\psi_{m_3}(\widetilde{z})$$

Then we evaluate u on the families of points

$$\begin{aligned} u^1_{m_2,m_3} &= \widetilde{u}(\delta,\xi_{m_2},\xi_{m_3}) \\ u^2_{m_1,m_3} &= \widetilde{u}(\xi_{m_1},\delta,\xi_{m_3}) \\ u^3_{m_1,m_2} &= \widetilde{u}(\xi_{m_1},\xi_{m_2},\delta) \end{aligned}$$

These operations consist of single sums

$$\begin{aligned} u^1_{m_2,m_3} &= \sum_{m_1} \psi_{m_1}(\delta)\; v_{m_1,m_2,m_3} \\ u^2_{m_1,m_3} &= \sum_{m_2} \psi_{m_2}(\delta)\; v_{m_1,m_2,m_3} \\ u^3_{m_1,m_2} &= \sum_{m_3} \psi_{m_3}(\delta)\; v_{m_1,m_2,m_3} \end{aligned}$$

which can be interpreted as matrix vector products with sparse matrices P_1, P_2, P_3

$$U^1 = P_1 V, \quad U^2 = P_2 V, \quad U^3 = P_3 V$$

Then, if the first face is a quadrangular face, we can split the computation

$$s^1_{p_2,p_3} = \sum_{m_2,m_3} \psi_{m_2}(\xi'_{p_2})\,\psi_{m_3}(\xi'_{p_3})\,u^1_{m_2,m_3}$$

into two stages

$$\begin{aligned} z_{m_2,p_3} &= \sum_{m_3} \psi_{m_3}(\xi'_{p_3})\,u^1_{m_2,m_3} \\ s^1_{p_2,p_3} &= \sum_{m_2} \psi_{m_2}(\xi'_{p_2})\,z_{m_2,p_3} \end{aligned}$$

which can be interpreted as matrix vector products with sparse matrices

$$S^1 = T_2\,T_1\,U^1$$

If the first face is a triangle, since we are using symmetric quadrature points for triangles [26] which are not tensorized, we only have

$$S^1 = T\,U^1$$

where matrix T is dense, but only restricted to the face. Therefore the complexity of computation of s_p is in $O(r^3)$ if the element is an hexahedron (only quadrangular faces) and in $O(r^4)$ for other elements, because of the presence of triangular faces. Of course, for other elements, some faces of the cube are not treated since they are reduced to a single point in the real element K. For pyramids, the face $z = 1$ is not treated.

2. We compute

$$\bar{s}_n = -\left(N_1\left\{s_n\right\} + N_2\left[s_n\right]\right)$$

$$s^1_n = \omega'_n\,\widetilde{|Dg|}(\xi'_n)\,\bar{s}_n$$

3. We are interested in the stage:

$$w_j^3 = \sum_n s_n^1 \cdot \widetilde{\varphi}_j(\xi'_n)$$

This stage is exactly the transpose operation to the computation of s_n, therefore the computation of w^3 is done by using transpose of matrices defined in the previous subsubsection (matrices P_1, P_2, P_3, T_1, T_2).

3.3. With Warburton's Trick

Using the transfomation 2.6, the stiffness matrix is transformed into

$$\begin{aligned}(R_h)_{j,k} &= \int_{\hat{K}} |DF| \sum_{1\le i\le d} A_i \frac{\partial(\frac{\hat{\varphi}_k}{\sqrt{|DF|}})}{\partial x_i} \cdot \frac{\hat{\varphi}_j}{\sqrt{|DF|}} d\hat{x} - \int_{\hat{K}} |DF| \sum_{1\le i\le d} B_i \frac{\hat{\varphi}_k}{\sqrt{|DF|}} \cdot \frac{\partial(\frac{\hat{\varphi}_j}{\sqrt{|DF|}})}{\partial x_i} d\hat{x} \\ &= \int_{\hat{K}} \sum_{1\le i\le d} A_i \frac{\partial \hat{\varphi}_k}{\partial x_i} \cdot \hat{\varphi}_j \, d\hat{x} - \int_{\hat{K}} \sum_{1\le i\le d} B_i \varphi_k \cdot \frac{\partial \hat{\varphi}_j}{\partial x_i} \, d\hat{x} \\ &\quad + \frac{1}{2} \int_{\hat{K}} \sum_{1\le i\le d} (B_i - A_i) \frac{1}{|DF|} \frac{\partial |DF|}{\partial x_i} \hat{\varphi}_k \cdot \hat{\varphi}_j \, d\hat{x}\end{aligned}$$

We can use the fast matrix vector product detailed above with this modified stiffness matrix without important increase in cost. Moreover, since the mass matrix is equal to identity, this leads to faster computations as shows Table 4.8. However, the stiffness matrix involves the derivative of the jacobian and this has to be treated carefully with pyramids. Indeed, for non-affine pyramids, the derivative of jacobian is singular :

$$\frac{\partial |DF|}{\partial \hat{x}} = B_1 \frac{1}{1-\hat{z}} + C \frac{\hat{y}}{(1-\hat{z})^2} = B_1 \frac{1}{1-\widetilde{z}} + C \frac{2\widetilde{y}-1}{1-\widetilde{z}}$$

It is preferable, for the volume integrals, to use a Gauss-Jacobi quadrature formula exact for polynomials in $(1-\widetilde{z})\mathbb{Q}_r$ instead of $(1-\widetilde{z})^2\mathbb{Q}_r$ as chosen classically for pyramids.

4. Comparison of Several Types of Elements

Experiments to compare the elements previously constructed are conducted on Maxwell's equations for which $d = 3$ and $n_s = 6$, and where u^m is E_x, E_y, E_z, H_x, H_y or H_z. The system is the following for E and H in $\mathbb{R}^3$

$$\begin{cases} \dfrac{\mathrm{d}E}{\mathrm{d}t} - \mathrm{rot}H = 0 \\ \dfrac{\mathrm{d}H}{\mathrm{d}t} + \mathrm{rot}E = 0 \end{cases} \tag{4.1}$$

where the matrices M and A_i are easily deduced from Equation 4.1, and the matrices B_i are obtained by taking the transpose of A_i.

We consider a cubic cavity $[-5,5]^3$ meshed with different types of elements, depending on the studied case, with a gaussian source

$$f = x\, e^{-\alpha r^2}\, e_x$$

4.1. Hexahedron

The orthogonal base of Proposition 2.1 provides a diagonal mass matrix for non-affine hexahedra, while the tensor product of Legendre polynomials proposed by Kirby *et al.* [9] (with $P_1 = P_2 = P_3 = r$) and Warburton [25] only provides mass-lumping for affine elements.

When the hexahedron is affine, the use of Legendre polynomials combined with a quadrature-free implementation of the matrix-vector product conduces to very efficient computations. In the Table 4.1, we have compared the computation time for one million degrees of freedom for $\mathbb{Q}_r$ and $\mathbb{P}_r$. We observe that for $\mathbb{P}_r$, the cost remains constant for higher orders than for $\mathbb{Q}_r$.

Table 4.1: Computational time spent for 100 iterations of leap-frog scheme for meshes of hexahedra containing 1 million DOFs for unknown E_x.

Order	2	3	4	5	6	7	8	10	12	15
$\mathbb{Q}_r$	136s	111s	107s	100s	101s	99s	109s	105s	113s	140s
$\mathbb{P}_r$	214s	157s	135s	117s	111s	105s	106s	101s	97s	99s

For a sharper comparison, we study the L^2-error obtained for a cubic cavity meshed with small cubes. Because of symmetry, only the cube $[0, 5\lambda]^2$ is meshed. In Table 4.2, we measure the minimal computational time and minimal number of degrees of freedom necessary to obtain an error less than 1%, for $\mathbb{P}_r$ and $\mathbb{Q}_r$.

Table 4.2: Time step, number of DOFs and computational time necessary to reach an error level below one percent for a cubic cavity (affine mesh) with hexahedra. Use of $\mathbb{P}_r$ with Legendre orthogonal expansion (top) - Use of $\mathbb{Q}_r$ with Lagrange interpolation functions base on Gauss-Legendre points (bottom).

Order	3	4	5	6	8	10	14	18	26
Δt	0.035	0.0405	0.0394	0.0395	0.0348	0.0325	0.0272	0.0235	0.0179
DOFs	540 000	240 065	153 664	111 804	84 480	61 776	43 520	35 910	29 232
CPU Time	11 818s	2865s	1387s	856s	695s	500s	444s	376s	620s

Order	3	4	5	6	7	9	11	16	30
Δt	0.032	0.036	0.0353	0.0344	0.0381	0.0315	0.0301	0.023	0.0156
DOFs	512 000	216 000	157 464	117 649	64 000	64 000	46 656	39 304	29 791
CPU Time	5 575s	1800s	1180s	1013s	417s	580s	426s	1134s	3520s

It seems better to use $\mathbb{P}_r$ than $\mathbb{Q}_r$ for higher orders, but all in all, the two methods give similar computational times. We think that using $\mathbb{Q}_r$ is a better choice than using $\mathbb{P}_r$ since the method also works for non-affine hexahedron, which means implementing a single method for all the cases.

4.2. Wedges

The orthogonal basis functions of Proposition 2.1 are almost the same as Kirby *et al.* [9] and Warburton [25], replacing Legendre polynomials in z by Lagrange interpolation functions

with Gauss points. For nodal functions, the tensorization in z leads to a matrix-vector product in $O(r^5)$ instead of $O(r^4)$ for orthogonal basis. In Table 4.3, the computational times given by both basis are very close, orthogonal functions provide faster computations from order 5.

Table 4.3: Time spent for 100 iterations of leap-frog scheme for meshes of non-affine wedges containing one million DOFs for E_x.

Order	2	3	4	5	6	7	8
Nodal	296	312	239	342	343	353	400
Orthogonal	338	326	280	315	292	317	361

Similar experiments than for affine-hexahedron have been conducted to compare the use of $\mathbb{P}_r$ or the optimal finite element space for wedges. For $\mathbb{P}_r$, we take the orthogonal basis functions of Proposition 2.2. The required number of degrees of freedom and computational time for the experiments can be read in Table 4.4.

Table 4.4: Time step, number of DOFs and computational time necessary to reach an error level below one percent for a cubic cavity meshed with affine prisms. Use of $\mathbb{P}_r$ (top) and optimal finite element space (bottom) with orthogonal expansion.

Order	3	4	5	6	7	8	9
Δt	0.0268	0.0296	0.0309	0.029	0.0285	0.0268	0.0258
DOFs	351 520	172 955	96 768	84 000	61 440	56 595	47 520
CPU Time	30 330s	12 920s	7 145s	7028s	5 560s	6 306s	6 602s

Order	3	4	5	6	7	8	9
Δt	0.0277	0.0306	0.031	0.0298	0.0268	0.025	0.025
DOFs	425 920	205 800	126 000	100 352	98 784	87 480	68 750
CPU Time	22 020s	6 834s	3 904s	3 340s	3 930s	4 397s	3 727s

In this case, we do not have the same advantage than for hexahedra where quadrature-free algorithms lead to faster computations. The use of optimal finite element space is then much better, even though the number of degrees of freedom is lower for $\mathbb{P}_r$.

4.3. Pyramids

Computations are done for a mesh containing only non-affine pyramids and one million degrees of freedom for unknown E_x. In Table 4.5, we compare the CPU time spent for the computation and Cholesky factorization of mass matrix for nodal and orthogonal basis functions, using the algorithm previously described. The computational time needed for nodal functions grows quickly, making the use of orthogonal functions very attractive when r is high. However, the impressive computional time (only spent during the computation of mass matrix) observed for nodal basis is not an important burden, since this operation is performed only once.

In Table 4.6, we display the ratio between the amount of memory used for a dense and a sparse mass matrix with orthogonal basis. The induced reduction of memory requirements is quite interesting for both the matrix and its Cholesky factorisation.

Table 4.5: Time to compute and invert mass matrix with the fast algorithm for pyramid elements

Order	2	3	4	5	6	7	8
Dense Matrix	1.91s	7.98s	26.2s	73.6s	177s	391s	786s
Sparse Matrix	0.577s	0.98s	1.32s	2.27s	4.08s	5.17s	7.62s

Table 4.6: Gain in storage for the matrix and its Cholesky factorization by using orthogonal functions instead of nodal functions for pyramidal elements.

Order	2	3	4	5	6	7	8
Gain for matrix	1.81	2.94	4.46	6.38	8.7	11.4	14.6
Gain for factorization	1.52	1.91	2.39	2.83	3.38	3.85	4.8

We finally perform a comparison between different solvers to invert the mass matrix: to be exhaustive, we consider a dense, a sparse and an iterative solver using fast matrix vector product deduced from the expression for the mass matrix given in Section 2.1. Table 4.7 shows that the sparse solver outperforms the other solvers. The main drawback of iterative solver is the relatively high number of iterations, since 10 iterations are needed to get a residual below 10^{-12}.

We compare the computation time spent for 100 leap-frog iterations with a mesh made of only non-affine pyramids only with one million DOFs in Table 4.8. The efficiency of the matrix-vector product seems quite interesting, since with Warburton's trick, the cost remains almost constant (between 400 and 500s here) with respect to the order of approximation. When a sparse solver is used, the additional increase in cost is quite acceptable if a high accuracy is sought. The cost of the matrix vector product using nodal functions becomes quickly prohibitive when r is large enough. We also observe a gain in time by using orthogonal functions from order 3.

Experiments are done to compare $\mathbb{P}_r$ and the optimal finite element space for pyramids, with the orthogonal base of Proposition 2.2 for $\mathbb{P}_r$, which is also the base proposed by Kirby *et*

Table 4.7: Comparison of iterative, sparse and dense solver used to solve $Mx = y$ for pyramidal elements.

Order	2	3	4	5	6	7	8
Dense Solver	91s	144s	183s	295s	340s	503s	652s
Iterative Solver	150s	150s	149s	331s	366s	403s	439s
Sparse Solver	74s	101s	121s	148s	178s	227s	235s

Table 4.8: Time spent for 100 iterations of leap-frog scheme for meshes of non-affine pyramids containing one million DOFs for E_x.

Order	2	3	4	5	6	7	8
Sparse solver	523s	505s	508s	569s	619s	692s	766s
Warburton trick	460s	432s	411s	451s	471s	494s	548s
Nodal functions	378s	532s	702s	1135s	2425s	7618s	15350s

al. [9]. The number of degrees of freedom and computational time can be read in Table 4.9.

Table 4.9: Time step, number of DOFs and computational time necessary to reach an error level below one percent for a cubic cavity meshed with affine pyramids. Use of $\mathbb{P}_r$ (top) and optimal finite element space (bottom) with orthogonal expansion.

Order	3	4	5	6	7	8	9
Δt	0.0238	0.0258	0.0248	0.024	0.0216	0.0205	0.0203
DOFs	960 000	461 370	336 000	258 048	246 960	213 840	165 000
CPU Time	60 449s	27 564s	22 422s	19 111s	21 433s	22 074s	19 056s

Order	3	4	5	6	7	8
Δt	0.0276	0.0307	0.028	0.0285	0.0268	0.0268
DOFs	737 280	330 000	279 552	181 440	153 000	109 440
CPU Time	28 267s	10 898s	10 800s	7 246s	6 933s	5 204s

The optimal finite element space is more efficient, even on affine pyramids, since the computational time required for eigth-order is 5 204s for optimal space, and 22 074s for polynomials.

4.4. Tetrahedra

For tetrahedra, the use of orthogonal functions is known to be not very attractive for low orders (Warburton [25]). In Table 4.10, we report computational times obtained for different orders of approximation with both nodal and orthogonal bases. Nodal basis functions provide faster algorithms when $r < 10$. We use quadrature-free algorithms as described in Hesthaven and Warburton [1] to evaluate the integrals.

Table 4.10: Time spent for 100 iterations of leap frog-scheme for meshes containing one million DOFs for E_x with affine tetrahedra

Order	2	3	4	5	6	7	8	9	10
Nodal	379s	358s	327s	343s	428s	541s	680s	1023s	1751s
Orthogonal	527s	601s	559s	595s	679s	722s	798s	992s	1074s

5. Applications to Maxwell's Equations

5.1. Convergence for a Cubic Cavity

To get the convergence order of the different methods tested, we consider the same test-case as in Section 4 with a mesh containing non-affine pyramids (see Fig 1.2). Here, the non-affine pyramids are not tending to affine elements as the mesh size h tends to 0. For example, you could think of trapezoids, whose size is tending to 0, but whose shape is always the same.

Fig. 5.1 displays the solutions for $T = 5$ and $T = 50$. An error analysis is performed for the solution obtained at $T = 50s$ for orders 3 and 5 by using the sparse solver to compute the mass matrix or Warburton's trick. The reference solution is taken as the numerical solution obtained with hexahedra of order 7. As shows Fig. 5.2, this trick does not provide a h-convergent scheme as expected, whereas our method does. However, we may conjecture that Warburton's

trick does provide a p-convergent scheme, since the error of consistency decreases when order 5 is used instead of order 3. Morerover, the level of consistency error (less than 2%) is quite acceptable in most simulations.

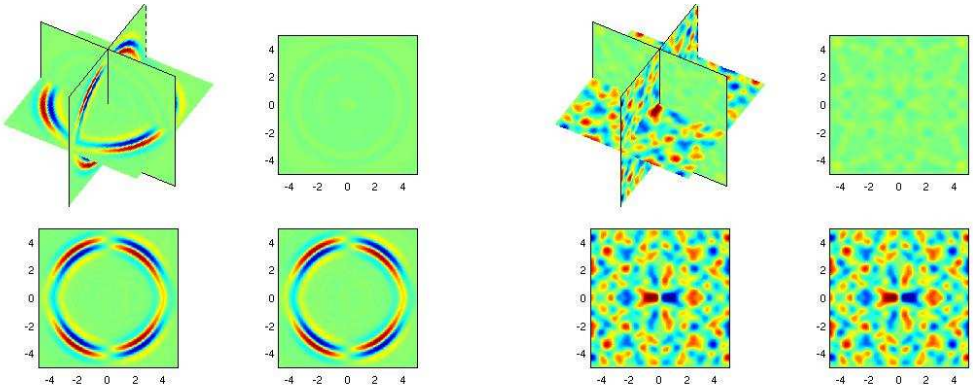


Fig. 5.1. Cube of size 10λ - Solution for $T = 5$ (left) and $T = 50$ (right)

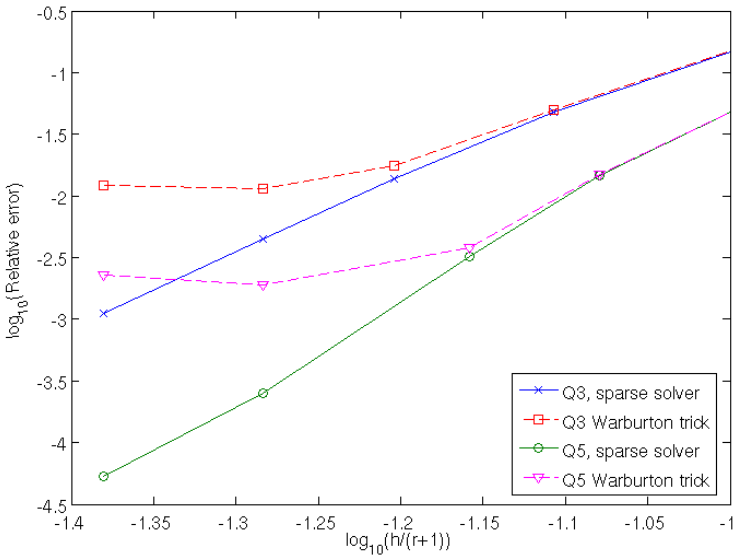


Fig. 5.2. Relative L^2 error obtained on deformed hybrid mesh (tetras+pyramids) - Cube of size 10λ - Fourth-order low-storage Runge-Kutta scheme

5.2. Air Balloon

We consider the scattering of an air balloon in a parallelepiped box $[-250, 50] \times [-130, 180] \times [90, 490]$, with the following source

$$f(x,t) = e^{-13.8(\frac{r}{r_0})^2}\, e^{-0.001(t-t_0)^2}\, sin(2\pi f_0 t), \qquad r_0 = 15,\ f_0 = 0.08$$

Table 5.1: Number of degrees of freedom and computational time needed to reach an L^2-error of 1% with fifth order approximation, for $T = 432s$ - Leap-frog scheme

Mesh used	L^2-error	Nb of DOFs	Δt	CPU Time	
				Nodal	Ortho
Hybrid	0.0093	22.4 million	0.032	4d 2h 36min	3d 17h 2min
Tetrahedral	0.011	37.9 million	0.046	11d 10h 19min	-

The hybrid mesh used for the experiment, chosen to provide an L^2-error below one percent with $\mathbb{Q}_5$ approximation, is presented on Fig. 5.3. We compare the results obtained on this hybrid mesh made of 122620 elements (92790 hexahedra (88966 affine), 12223 tetrahedra, 15063 pyramids (4363 affine) and 2544 wedges (1 affine), with the results obtained on a pure tetrahedral mesh made of 677069 affine tetrahedra.

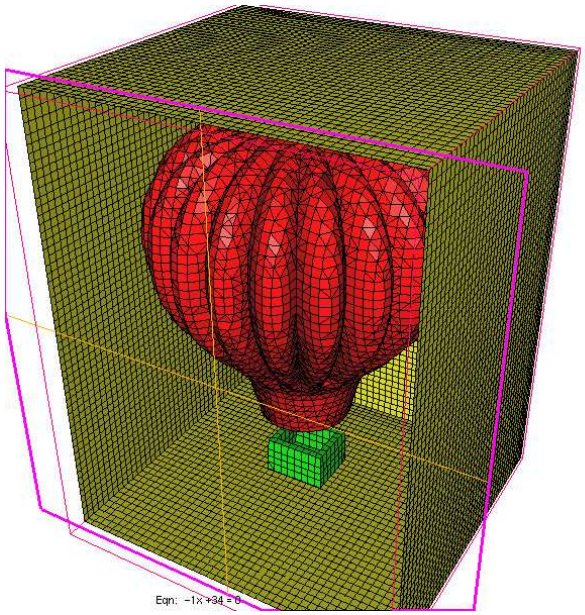


Fig. 5.3. Hybrid mesh used for the computations of the scattering of an air balloon.

For this experiment, we use the classical leap-frog scheme for the time discretization. Because of the presence of small elements, the stability condition (CFL) is quite restrictive, so that we have chosen the time step as the maximal allowable time step. Due to the restrictive stability condition, a second-order leap-frog scheme has been preferred. The error observed here is mainly due to the space approximation, the time error is below 0.1%.

Fig. 5.4 displays the E_x component of the numerical solution obtained for $T = 288$ and $T = 432$. Since the mesh is not curved, we compare each solution to a numerical solution computed on the same mesh but with an order of approximation equal to $r + 1$ instead of r. In Table 5.1, we detail the computational time needed to reach an L^2 error lower than one percent.

Numerical experiments have been completed on 256 processors. Computational times are obtained by summing the CPU time spent on each processor and subtracting the cost of communications so that the global time is not influenced by problems of load-balancing. The obtained

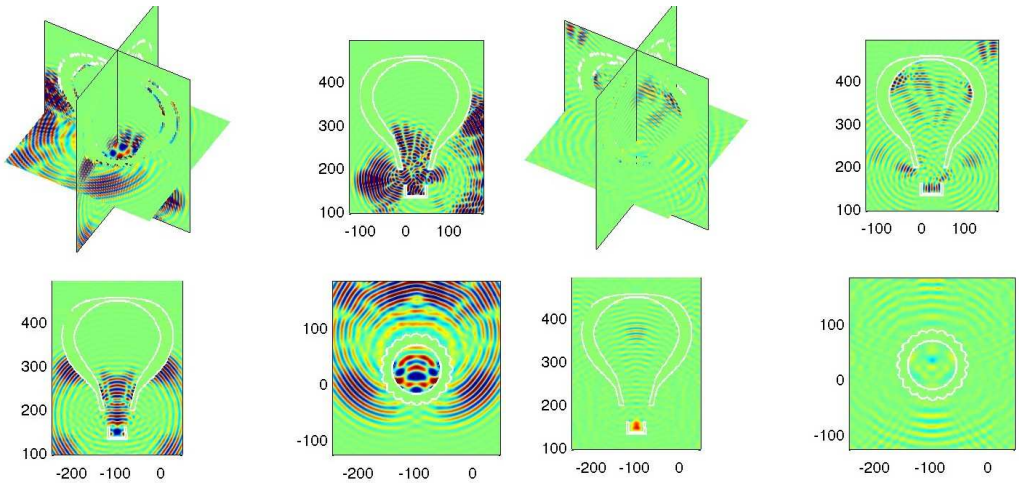


Fig. 5.4. Solution obtained for $T = 288$ (left) and $T = 432$ (right).

ideal time would be the same time obtained if the simulation had been launched on a single processor. Besides, the observed parallel efficiency was greater than 80 %.

Conclusion

Highly efficient pyramidal elements of any order have been constructed for discontinuous Galerkin methods. Numerical experiments conducted with these elements on Maxwell's equations showed a very good behaviour with hybrid meshes.

References

[1] J. Hesthaven and T. Warburton, High order discontinuous Galerkin methods for the Maxwell eigenvalue problem, *Phil. Trans. R. Soc. Lond. Ser. Math. Phys. Eng. Sci.*, **1816** (2004), 493–524.

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

[3] M. Duruflé, Intégration numérique et éléments finis d'ordre élevé appliqués aux équations de Maxwell en régime harmonique., PhD thesis, Université Paris IX-Dauphine, 2006.

[4] L. Demkowicz, J. Kurtz, D. Pardo, M. Paszynski, W. Rachowicz and A. Zdunek, Computing With hp-Adaptive Finite Element, Volume II, Chapman and Hal, 2007.

[5] N. Nigam and J. Phillips, Higher-order finite elements on pyramids, *accepted IMA J. Numer.Anal*, (2011).

[6] N. Nigam and J. Phillips, Numerical integration for high order pyramidal finite elements, *submitted to Comp. Meth. in App. Mech. and Eng.*, (2010).

[7] M. Bergot, G. Cohen and M. Duruflé, Higher-order finite elements for hybrid meshes using new nodal pyramidal elements, *J. Sci. Comput*, **42**:3 (2010), 345–381.

[8] S. Sherwin, T. Warburton and G. Karniadakis, Spectral/hp methods for elliptic problems on hybrid grids, *Contemp. Math.*, **218** (1998), 191–216.

[9] R. Kirby, T. Warburton, I. Lomtev and G. Karniadakis, A discontinuous Galerkin spectral/hp method on hybrid grids, *App. Num. Math.*, **33** (2000), 393–405.

[10] G. Gassner, F. Lörcher, C. Munz and J. Hesthaven, Polymorphic nodal elements and their application in discontinuous Galerkin methods, *J. Comp. Phys.*, **228**:5 (2009), 1573–1590.

[11] G. Cohen, Higher-Order Numerical Methods for Transient Wave Equations, Springer Verlag, 2002.

[12] M.J.S.Chin-Joe-Kong, W. Mulder and M.V. Veldhuizen, Higher-order triangular and tetrahedral finite element with mass lumping for solving the wave equation, *J. Eng. Math*, **35** (1999), 405–426.

[13] M. Costabel and M. Dauge, Weighted regularization of Maxwell equations in polyhedral domains a rehabilitation of nodal finite element, *Numer. Math.*, **93** (2002), 239–277.

[14] T. Warburton, A low storage curvilinear discontinuous Galerkin time-domain method for electromagnetics, Electromagnetic Theory (EMTS), URSI International Symposium, pages 996–999, IEEE, 2010.

[15] P.R. E. Godlewski, Hyperbolic Systems of Conservation Laws, Ellipses, 1991.

[16] J. Hesthaven and T. Warburton, High-order nodal methods on unstructured grids. i. time-domain solution of Maxwell's equations, *J. Comp. Phys.*, **181**:1 (2002), 186–221.

[17] P. Ciarlet, The Finite Element Method for Elliptic Problems, North-Holland, 1978.

[18] G. Bedrosian, Shape functions and integration formulas for three-dimensional finite element analysis, *Int. J. Numer. Meth. Eng.*, **35** (1992), 95–108.

[19] M. Duruflé, P. Grob and P. Joly, Influence of the gauss and gauss-lobatto quadrature rules on the accuracy of a quadrilateral finite element method in the time domain, *Num. Meth. Part. Diff. Eq.*, **25** (2009), 526–551.

[20] S. Pernet, Étude de méthodes d'ordre élevé pour résoudre les équations de Maxwell dans le domaine temporel. Application à la détection et à la compatibilité électromagnétique, PhD thesis, Université de Paris IX Dauphine, 2004.

[21] H. Carpenter and C.A. Kennedy, Fourth-order 2n-storage Runge-Kutta schemes, Technical report, NASA Langley Research Center, 1994.

[22] P. Monk, Finite elements methods for Maxwell's equations, Oxford Science Publication, 2002.

[23] P. Hammer, O. Marlowe and A. Stroud, Numerical integration over simplexes and cones, *Mathematical Tables and Other Aids to Computation*, **10**:55 (1956), 130–137.

[24] G. Szegö, Orthogonal Polynomials - Ch. 4 Jacobi Polynomials, 4th ed Amer. Math. Soc., Provi-

dence, RI, 1975.

[25] T. Warburton, Spectral/hp Methods on Polymorphic Multi-Domains: Algorithms and Applications, PhD thesis, Brown University, 1999.

[26] D. Dunavant, High degree efficient symmetrical gaussian quadrature rules for the triangle, *Int. J. Num. Meth. Eng.*, **21** (1985), 1129–1148.