

Application of Discontinuous Galerkin spectral method on hexahedral elements for aeroacoustic

Nicolas Castel, Gary Cohen, Marc Duruflé

▶ **To cite this version:**

Nicolas Castel, Gary Cohen, Marc Duruflé. Application of Discontinuous Galerkin spectral method on hexahedral elements for aeroacoustic. Journal of Computational Acoustics, 2009, 17 (2), pp.175-196. hal-00403787

HAL Id: hal-00403787
https://hal.science/hal-00403787

Submitted on 13 Jul 2009

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.

Application of Discontinuous Galerkin spectral method on hexahedral elements for aeroacoustic

N. CASTEL
INRIA Rocquencourt, projet POEMS
78 153 Le Chesnay, FRANCE
nicolas.castel@inria.fr

G. COHEN
INRIA Rocquencourt, projet POEMS
78 153 Le Chesnay, FRANCE
gary.cohen@inria.fr

M. DURUFLÉ
INRIA Rocquencourt, projet POEMS
78 153 Le Chesnay, FRANCE
marc.durufle@inria.fr

Received (Day Month Year)
Revised (Day Month Year)

A discontinuous Galerkin method is developed for linear hyperbolic systems on general hexahedral meshes. The use of hexahedral elements and tensorized quadrature formulas to evaluate the integrals leads to an efficient matrix-vector product. It is shown for high order approximations, the reduction in computational time can be very important, compared to tetrahedral elements. Two choices of quadrature points are considered, the Gauss points or Gauss-Lobatto points. The method is applied to the aeroacoustic system ("simplified" Linearized Euler Equations). Some 3-D numericals experiments show the importance of penalization, and the advantage of using high order.

Keywords: Aeroacoustic, Discontinuous Galerkin, Finite Element Method, Mass Lumping, Linearized Euler Equations

1. Discontinuous Galerkin method on hexahedral meshes

The discontinuous Galerkin method is widely used for many applications, like Maxwell equations [3], or Linearized Euler Equations (LEE) [16]. The discontinuous Galerkin method is well suited for equations, for which the functional spaces, where the solution lies, are difficult to discretize. The use of discontinuous Galerkin method has been widely detailed in the case of tetrahedral elements [3], but less so for hexahedral elements. Cohen's article [15] describes the use of hexahedral elements in the particular case of Maxwell equations. This paper describes the implementation of the discontinuous Galerkin method on general hyperbolic systems, for hexahedral elements with mass lumping techniques, and for application to Linearized Euler Equations (LEE). A comparison of the computational efficiency between tetrahedral and hexahedral elements is done in the particular case of aeroacoustic equations

(LEE). The advantage of using hexahedral elements is shown for high order approximations. Because of the discontinuity of the space, there are several possible quadrature formulas to perform mass lumping. Among these possiblities, two are relevant : Gauss Legendre points, and Gauss-Lobatto points. The Gauss Legendre formulas provide the best accuracy (exact integration of polynoms in Q_{2r+1}), while the Gauss-Lobatto formulas are a bit less accurate (exact integration of polynoms in Q_{2r-1}), but have the advantage to include the two extremities of the interval $[0, 1]$, therefore leading to faster computations. For the application of the method to LEE, we will show which choice is the more efficient and robust. If nodal finite element is chosen, some spurious waves appear, and can be removed with some specific methods, as detailed in this paper solving the Galbrun equation [1]. Such methods are a bit tedious to implement, particularly on corners (cf. [2]), the exploitation of discontinuous Galerkin would seem a better alternative. Similar to the treatment for Maxwell's equations [13], we show for LEE the importance of adding "penalty" terms to the original discontinuous formulation, in order to avoid spurious waves. These penalty terms can be viewed as considering upwind fluxes instead of centered fluxes.

Let $\mathbb{R}$ denote the set of all real numbers, then

$$\mathbb{R}^d = \{(x_1, \cdots, x_d) \text{ where } x_1, \cdots, x_d \in \mathbb{R}\}$$

Let us consider a linear hyperbolic system to solve in $(x, t) \in \mathbb{R}^d \times \mathbb{R}^+$:

$$\begin{cases} M(x)\dfrac{\partial U}{\partial t} + \displaystyle\sum_{i=1}^{d} \frac{\partial (A_i(x)\, U)}{\partial x_i} = F(x,t) \\ U(x,0) = U_0(x) \end{cases} \tag{1}$$

$$\begin{cases} A_i(x) : R^d \to M_m(R) \ , \forall i = 1..d \\ F : R^d \times R^+ \to R^m \quad , U_0 : R^d \to R^m \end{cases} \tag{2}$$

The system is hyperbolic in Friedrich's sense if

- $M(x)$ is symmetric and definite positive,
- $A_1, \cdots, A_d$ are symmetric matrices for all values of the space variable x.

The original Linearized Euler Equations (LEE) may exhibit instabilities for the continuous equations. These instabilities are physical (called Kelvin-Helmholtz instabilities) and are limited by non-linear terms for original Euler equations, whereas they lead to exponential growths for the LEE. In [16], a special treatment is proposed to remove these instabilities, and it is compared to the model of Bailly-Bogey-Juve [11]. That model has been numerically and theoretically proven accurate, is stable, and is particularly simple to use, which is why we have chosen the Bailly-Bogey-Juve model. In this model, the LEE are

written by introducing a "right-hand side" called H whose expression is

$$H = \begin{vmatrix} (\gamma-1)(p\nabla\cdot v_0 - \vec{v}\cdot\nabla p_0) \\ (\rho_0\vec{v}+\rho v_0)\cdot\nabla v_x \\ (\rho_0\vec{v}+\rho v_0)\cdot\nabla v_y \\ (\rho_0\vec{v}+\rho v_0)\cdot\nabla v_z \end{vmatrix}$$

The LEE are then simplified by neglecting this term, leading to "simplified" LEE. As a consequence, Kelvin-Helmholtz instabilities are removed and a decoupling of the mass density ρ and the pressure p is performed. As a consequence, we can discretize only four unknowns (in 3-D) : the pressure p and the vector $w = \rho_0 v$. The equations associated to this model, for a flow $\vec{v}_0$, can be written in 2-D as an hyperbolic system as follows :

$$\left\{ \begin{array}{l} \dfrac{\partial}{\partial t}p + \vec{v}_0\cdot\nabla p + c_0^2\,\nabla\cdot(\vec{w}) = f \\ \dfrac{\partial}{\partial t}\vec{w} + \begin{pmatrix} \vec{v}_0\cdot\nabla w_x \\ \vec{v}_0\cdot\nabla w_y \end{pmatrix} + \nabla p = \vec{0} \end{array} \right.$$

$$\Longleftrightarrow$$

$$\partial_t \begin{pmatrix} p \\ w_x \\ w_y \end{pmatrix} + \begin{pmatrix} v_{0x} & c_0^2 & 0 \\ 1 & v_{0x} & 0 \\ 0 & 0 & v_{0x} \end{pmatrix} \partial_x \begin{pmatrix} p \\ w_x \\ w_y \end{pmatrix} + \begin{pmatrix} v_{0y} & 0 & c_0^2 \\ 0 & v_{0y} & 0 \\ 1 & 0 & v_{0y} \end{pmatrix} \partial_y \begin{pmatrix} p \\ w_x \\ w_y \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ 0 \end{pmatrix}$$

$$\Longleftrightarrow$$

$$\partial_t U + A_1\,\partial_{x_1}U + A_2\,\partial_{x_2}U = F$$

where $c_0^2 = \frac{\gamma\rho_0}{p_0}$.

1.1. *Variational formulation*

The Discontinuous Galerkin formulation is written on each element K of the mesh as

$$\int_K \frac{\partial u}{\partial t}\varphi\,dx - \sum_{i=1}^{d}\int_K A_i(x)\,u\,\frac{\partial\varphi}{\partial x_i}\,dx + \int_{\partial K}\sum_{i=1}^{d} n_i\,A_i(x)\{u\}\varphi\;dx + \alpha\int_{\partial K} C(x)[u]\varphi\,dx = \int_K f\,\varphi\,dx$$

where n_i is the i-th component of the outward normal to the boundary, $\{u\}$ is the average value, $[u]$ the half the difference of u across the boundary, i.e.

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

$$[u] = \frac{1}{2}(u_2 - u_1)$$

where u_1 and u_2 are the two values of u across the boundary. u_1 is the "inside" value defined on the element considered K.

The computational domain Ω is discretized with a mesh of quadrilateral/hexahedral elements :

$$\Omega = \bigcup_e K_e$$

A quadrilateral/hexahedral element is defined as the image, by the classical bilinear/trilinear map F_e, of the unit square/cube $\hat{K} = [0,1]^d$. An illustration of the transformation F_e is shown in Fig 1. We denote DF_e as Jacobian matrix of this transformation, and J_e its deter-

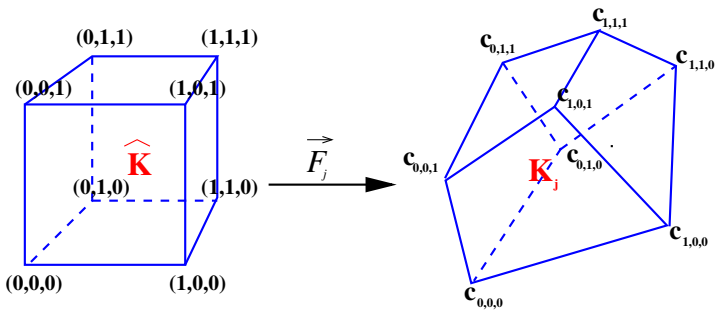


Fig. 1. F_e for the dimension 3.

minant. A more detailed definition of the properties required by this trilinear transformation in order to have a non-degenerated element, can be found in [7]. The finite element space is

$$U_h^r = \{u \in L^2(\Omega) \quad \forall e, \quad u|_{K_e} \in Q_r \}$$

where Q_r is the space of polynomials whose degree is equal or less than r in each variable.

For the discretization of Q_r, the use of tensorized basis functions will take advantage of the tensorized structure of the cube. Our choice has been to use a "mass-lumping" technique [14], in order to get a diagonal mass matrix, and to deal easily with the case where the coefficients A_i vary pointwise inside each element. The mass lumping technique selects the same points for both quadrature formulas and degrees of freedom (Lagrange interpolation). We shall consider the Gauss-Legendre or Gauss-Lobatto points and related quadrature formulas, as described in [8] or [9]. These are obtained by tensor products of 1-D points, as shown in Fig 2. Below is the fundamental accuracy property of the Gauss and Gauss-Lobatto quadrature rules

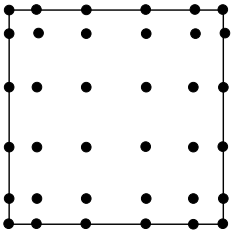


Fig. 2. 2-D Gauss-Lobatto points for $r = 5$

Proposition 1.1. *If we denote by* $(\widehat{\omega}_i^G, \widehat{\xi}_i^G)$ *and* $(\widehat{\omega}_i^{GL}, \widehat{\xi}_i^{GL})$, $1 \leq i \leq (r+1)^d$, *the Gauss and Gauss-Lobatto weights and points in* $\widehat{K}$ *[8], then we have:*

$$\int_{\widehat{K}} \widehat{f}(\widehat{x})\, d\widehat{x} \;=\; \sum_{k=1}^{(r+1)^d} \widehat{\omega}_k^G\, \widehat{f}(\widehat{\xi}_k^G)\,, \quad \forall\, \widehat{f} \in Q_{2r+1}(\widehat{K})\,, \tag{3}$$

$$\int_{\widehat{K}} \widehat{g}(\widehat{x})\, d\widehat{x} \;=\; \sum_{k=1}^{(r+1)^d} \widehat{\omega}_k^{GL}\, \widehat{g}(\widehat{\xi}_k^{GL})\,, \quad \forall\, \widehat{g} \in Q_{2r-1}(\widehat{K})\,. \tag{4}$$

The basis functions based on these degrees of freedom are tensorized as follows

$$\hat{\varphi}_i(\hat{x}_1, \hat{x}_2, \hat{x}_3) \;=\; \hat{\varphi}_{i_1}(\hat{x}_1)\, \hat{\varphi}_{i_2}(\hat{x}_2)\, \hat{\varphi}_{i_3}(\hat{z}_3)$$

For each "3-D" index i, there is a corresponding tuple denoted (i_1, i_2, i_3). The basis functions on the hexahedron K_e are defined via the transformation F_e :

$$\varphi_i \circ F_e \;=\; \hat{\varphi}_i$$

The problem is to find $u_h \in U_h^r$ so that

$$\begin{cases} \forall K \in \Omega_h, \quad \forall \varphi_h \in U_h^r \\ \displaystyle\int_K \frac{\partial u_h}{\partial t}\, \varphi_h\, dx \;-\; \sum_{i=1}^{d} \int_K A_i(x)\, u_h\, \frac{\partial \varphi_h}{\partial x_i}\, dx \;+\; \int_{\partial K} \sum_{i=1}^{d} n_i\, A_i(x)\, \{u_h\}\, \varphi_h\, dx \\ \displaystyle\;+\; \int_{\partial K} C(x)\, [u_h]\, \varphi_h\, dx \;=\; \int_K f\, \varphi_h\, dx \end{cases} \tag{5}$$

We define the mass matrix

$$(M_h)_{j,k} \;=\; \sum_e \int_{K_e} \varphi_j\, \varphi_k\, dx$$

the stiffness matrix

$$(K_h)_{j,k} \;=\; -\sum_e \Big(\sum_{i=1}^{d} \int_{K_e} A_i(x)\, \varphi_k \frac{\partial \varphi_j}{\partial x_i}\, dx \;+\; \int_{\partial K_e} \sum_{i=1}^{d} n_i\, A_i(x) \{\varphi_k\} \varphi_j\, dx \Big)$$

and the penalty matrix

$$(P_h)_{j,k} = \sum_e \int_{\partial K_e} C(x)[\varphi_k]\varphi_j \, dx$$

We then have the following differential system :

$$M_h \frac{\mathrm{d}U}{\mathrm{d}t} + K_h U + \alpha P_h U = F_h(t)$$

In the case of a symmetric system (ie the matrices A_i are symmetric), we can split the matrices A_i as follows

$$A_i = B_i + B_i^*$$

Several splittings are possible, we propose to use the following one :

$$B_i = \frac{1}{2} D_i + L_i$$

where D_i is the diagonal matrix of A_i and L_i the lower triangular part (not including diagonal). In the variational formulation, we do integration by parts on one term, so that we obtain the "split" formulation

$$\begin{aligned} \int_K \frac{\partial u}{\partial t} \varphi \, dx - \sum_{i=1}^d \int_K B_i(x) \, u \, \frac{\partial \varphi}{\partial x_i} \, dx + \int_{\delta K} \sum_{i=1}^d n_i \, B_i(x) \, \{u\} \, \varphi \, dx \\ + \sum_{i=1}^d \int_K B_i^*(x) \, \frac{\partial u}{\partial x_i} \, \varphi \, dx + \int_{\delta K} \sum_{i=1}^d n_i \, B_i^*(x) \, [u] \, \varphi \, dx \\ + \int_K \frac{\partial B_i}{\partial x_i} \, u \, \varphi \, dx + \int_{\delta K} \sum_{i=1}^d C_i(x) \, [u] \, \varphi \, dx = \int_K f \, \varphi \, dx \end{aligned} \tag{6}$$

The advantage of this decomposition is to explicitely show the skew-symmetry of the stiffness matrix K_h, in the special case where the matrices A_i are independent of the location (the term $\dfrac{\partial A_i}{\partial x_i}$ is equal to 0), and the penalty terms are set to zero.
The disadvantage of this split formulation is slower matrix-vector calculations. Hence, we have not used this formulation in practice and will provide more detail in the next subsection.

1.2. *Stability*

The skew-symmetry of the stiffness matrix induces a conservative scheme, if using a second-order leap-frog scheme [4]. In this thesis, the author shows the conservation of an energy and thus the stability of the scheme. This study is done on tetrahedral elements and can easily be extended to hexahedral elements since the only property needed is the skew-symmetry of the matrix.

Theorem 1.1. *In the case where the matrices A_i are uniform per element, a quadrature formula exact for $Q_{2\,r+1}$ is exact for the evaluation of the stiffness matrix*

Proof. The demonstration is done in 3-D. Let us define :

$$Q_{r_1,r_2,r_3} \,=\, span[x_1^{\alpha_1} x_2^{\alpha_3} x_d^{\alpha_d}], \quad \alpha \in \mathcal{S}_{r_1,r_2,r_3} = \{\alpha \in \mathbb{N}^3, \quad \alpha_j \leq r_j,\ \forall j\}. \tag{7}$$

We have the following assessment

$$\text{if } u \in Q_{r_1,r_2,r_3}, v \in Q_{r'_1,r'_2,r'_3} \text{ then } uv \in Q_{r_1+r'_1,r_2+r'_2,r_3+r_3}$$

This property will be extensively used to estimate the polynomial degrees of the different terms. $\hat{\nabla}$ is the gradient operator for variable $\hat{x}$ relying on the unit cube $\hat{K}$ whereas ∇ is the gradient operator for variable $\hat{x}$ relying on the real hexahedron K. After a change of variables, the volumic integral in the stiffness matrix can be written as

$$\sum_i \int_{\hat{K}} A_i^{p,q}\, \hat{\varphi}_k\, (\, J_e\, DF_e^{*-1}\, \hat{\nabla}\hat{\varphi}_j) \cdot e_i\, dx$$

By definition, the transformation $F_e \,\in Q^3_{1,1,1}$. Differentiating with respect to each variable, we lose one order, thus we obtain the following properties for the Jacobian matrix :

$$(DF_e)_{i,1} \in Q_{0,1,1}\,,\ (DF_e)_{i,2} \in Q_{1,0,1}\,,\ (DF_e)_{i,3} \in Q_{1,1,0}$$

Matrix $J_e\, DF_e^{*-1}$ is the transpose of the cofactors of DF_e, and a straightforward computation provides

$$(J_e DF_e^{*-1})_{1,i} \in Q_{2,1,1}\,,\ (J_e DF_e^{*-1})_{2,i} \in Q_{1,2,1}\,,\ (J_e DF_e^{*-1})_{3,i} \in Q_{1,1,2}$$

For a basis function $\hat{\varphi}_i \,\in Q_{r,r,r}$, we have

$$\hat{\nabla}\hat{\varphi}_i \in Q_{r-1,r,r} \times Q_{r,r-1,r} \times Q_{r,r,r-1}$$

Finally, by summing the polynomial degrees, we get

$$J_e\, DF_e^{*-1} \hat{\nabla}\hat{\varphi}_i \in Q^3_{r+1}$$

Thus

$$\hat{\varphi}_k\, (J_e DF_e^{*-1} \hat{\nabla}\hat{\varphi}_j) \cdot e_i \;\in\; Q_{2\,r+1}$$

For the surface integral in the stiffness matrix, the change of variables provides

$$\sum_i \int_{\partial\hat{K}} ds_e\, n_e\, A_i^{p,q}\, \hat{\varphi}_j \hat{\varphi}_k\, dx$$

The weighed normal $ds_e\, n_e$ belongs to Q_1 because each face of the hexahedron is defined as image of a bilinear transformation F_N from the unit square. As a consequence $ds_e\, n_e\, \hat{\varphi}_j \hat{\varphi}_k \,\in Q_{2\,r+1}$. Therefore, We need a quadrature formula exact for $Q_{2\,r+1}$, to exactly compute the stiffness matrix. □

The use of Gauss points leads then to an exact evaluation of the stiffness matrix (for the mass matrix, the integration is not exact in 3-D, because $J_e \in Q_2$). That's why we have the same stability result as for tetrahedral elements. The use of Gauss-Lobatto points provide an approximate integration, and one can prove the stability only when using the split formulation (6). If we use the original formulation (5) to compute the stiffness matrix with Gauss-Lobatto points, the computed stiffness matrix is not skew-symmetric and can lead to instabilities. Nevertheless, these instabilities can be removed by setting a penalty term large enough in the original formulation. However, the stability in that case is an open question : an explanation can be that the instability is eliminated for a sufficiently long period of time so that the instability is not observed in our experiments.

1.2.1. *Effect of penalization*

We introduced a penalization term with a matrix C. If an upwind flux is chosen (Roe flux), the matrix C would read

$$C = |\sum A_i n_i|$$

In the case of a null flow, it is possible to prove that (if $\alpha = \delta = 1$)

$$C = \begin{pmatrix} \alpha & 0 \\ 0 & \delta n \; n^* \end{pmatrix}$$

with $\alpha < 0$ and $\delta < 0$.
We could also have taken $C = \alpha Id$ or C diagonal with negative coefficients, the effects are similar. We chose to consider the same matrix C even for a non-uniform flow, the main advantage is to have a matrix C independent of the flow. The case where $\alpha = delta = 0$ corresponds to centered fluxes, whereas the case $\alpha = \delta = 1$ correspond to classical Roe fluxes (upwind fluxes). For the acoustic equation (flow set to zero), the effect of penalization is to reduce the dispersion [5] and improve the order of convergence [3], but it adds some dissipation and degrades the CFL stability condition. To illustrate the benefit of the penalization in the solution, we display in Fig 3 the time-harmonic solution in an open domain (bounded with PML layers). The solution is slightly better, but the necessity of penalization is not obvious.

Let us try the same experiment with an uniform flow $M = (0.5; 0)$. As shown in Fig 4, the improvment is remarkable. We observe the same type of results, if absorbing, Neumann, or periodic boundary conditions are used. The penalization has the effect of removing spurious modes, as it has been observed for Maxwell equations [13]. Another beneficial effect of penalization is the possibility of using Gauss-Lobatto points without using the split formulation (6).

The need for penalization in the case of a non-null flow is obvious. So, we shall always use this penalization in the numerical experiments. For the choice of the coefficients α and δ,

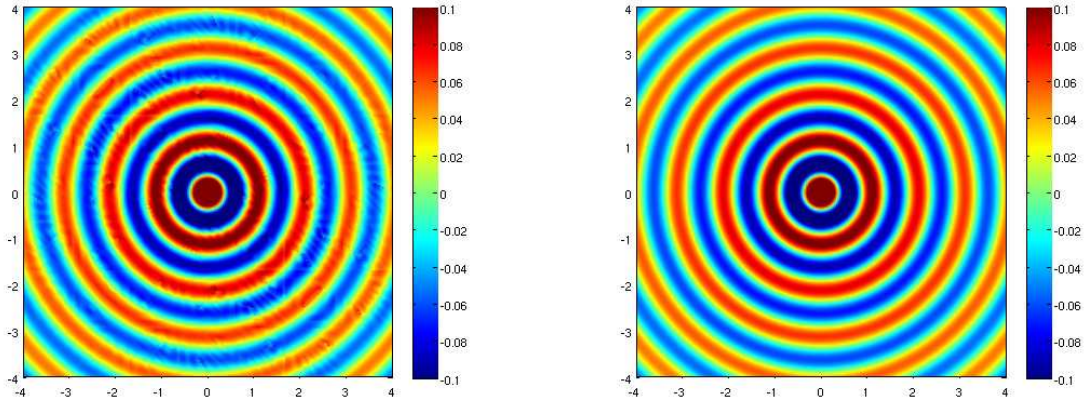


Fig. 3. Point source in an open domain with a null flow. At left, without penalization and at right with $\alpha = -0.5$. A triangular mesh split of quadrilaterals is used.

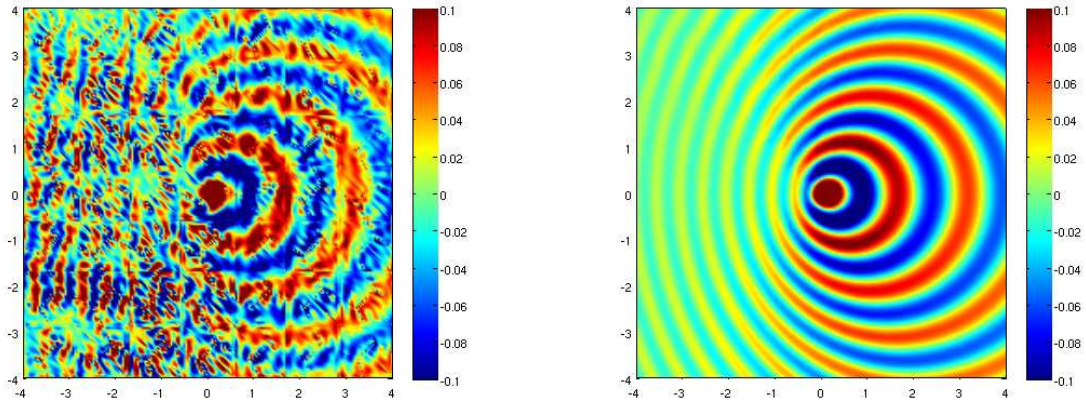


Fig. 4. Point source in an open domain with a uniform flow. At left, without penalization and at right with $\alpha = \delta = -0.5$. Triangular meshes split in quadrilaterals are used.

the error made on the solution displayed in Fig 4 is computed for different values of these parameters. In Fig 5, we can see that the solution is dramatically improved by adding these penalization terms. For a regular mesh, the error decreases from 5 % to 0.2 %, and for a distorted mesh, the error decreases from 35 % to 1 %. There is no "special" optimal value of these parameters, but a good choice can be

$$\alpha = \delta = -0.5$$

When the mesh is refined, or if the order of approximation is modified, similar error reductions are observed.

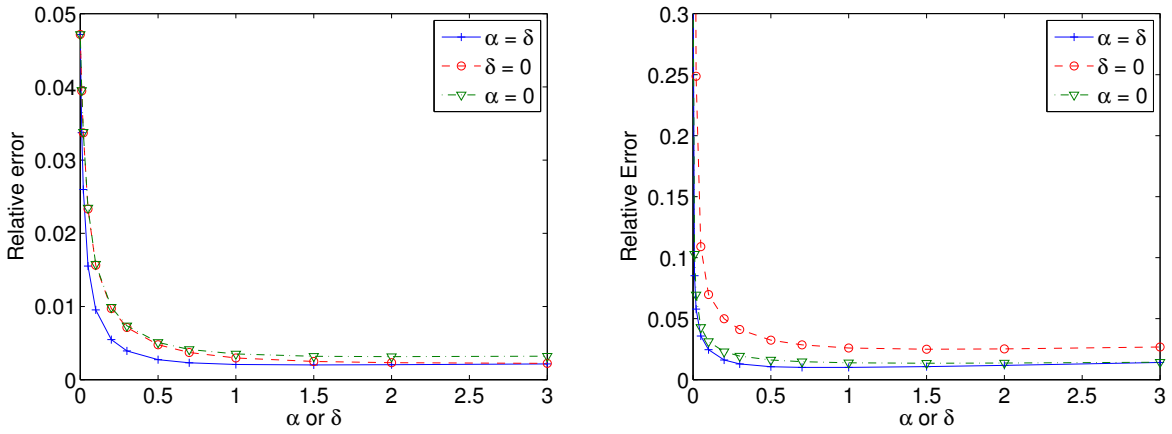


Fig. 5. Relative error according to the absolute value of α or δ, if we choose $\alpha = \delta$, $\alpha = 0$ or $\delta = 0$. At left, the mesh is regular, at right the mesh is composed of triangles split in quads. Gauss-Lobatto points are used with a Q_5 approximation.

These penalization terms decrease the CFL number, which slows the computation. To illustrate this effect, the maximal time step for which stability is observed is displayed in Fig 6 when $\alpha = \delta$ for a Runge-Kutta scheme. In these results, the CFL doesn't decrease until the

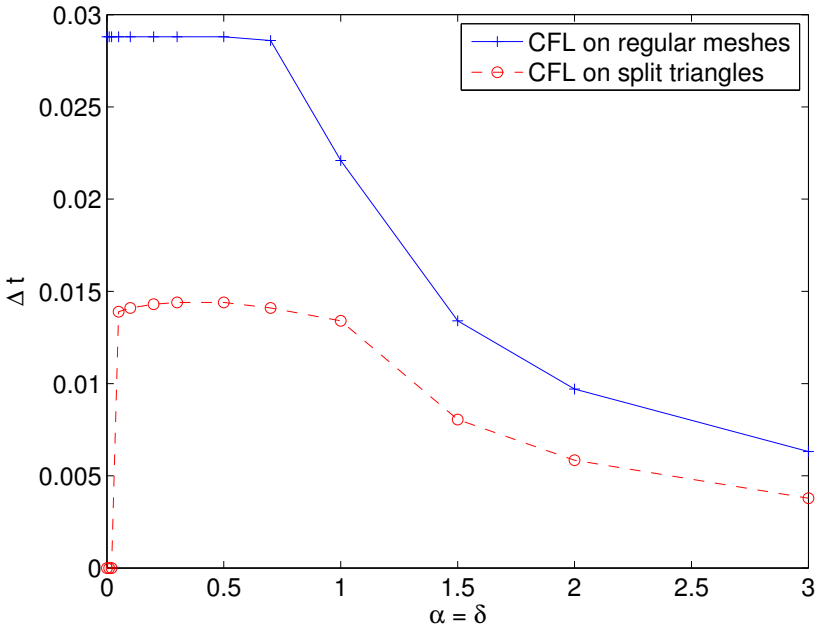


Fig. 6. Maximal Δt for which the fourth order Runge-Kutta scheme is stable, for a regular or distorted mesh.

value approaches

$$\alpha = \delta = -0.5$$

Then, the reduction scales as $1/\alpha$. In the case of distorted mesh, the CFL observed for very small values of α is equal to 0, because Gauss-Lobatto points are used with the original formulation (5). We have already mentioned that in this case, instabilities can be observed. It is important to note that these instabilities are "spatial" in the sense that they always occur at the same time T, independently of the time scheme and time step used. The effect of the penalization is to reduce these instabilities, i.e. the value T increases with the value of $|\alpha|$, so that for $|\alpha|$ large enough, we do not observe the instability during the simulation time period. An illustration of this phenomenon is provided in Fig 7 For $\alpha = -0.5$, instability

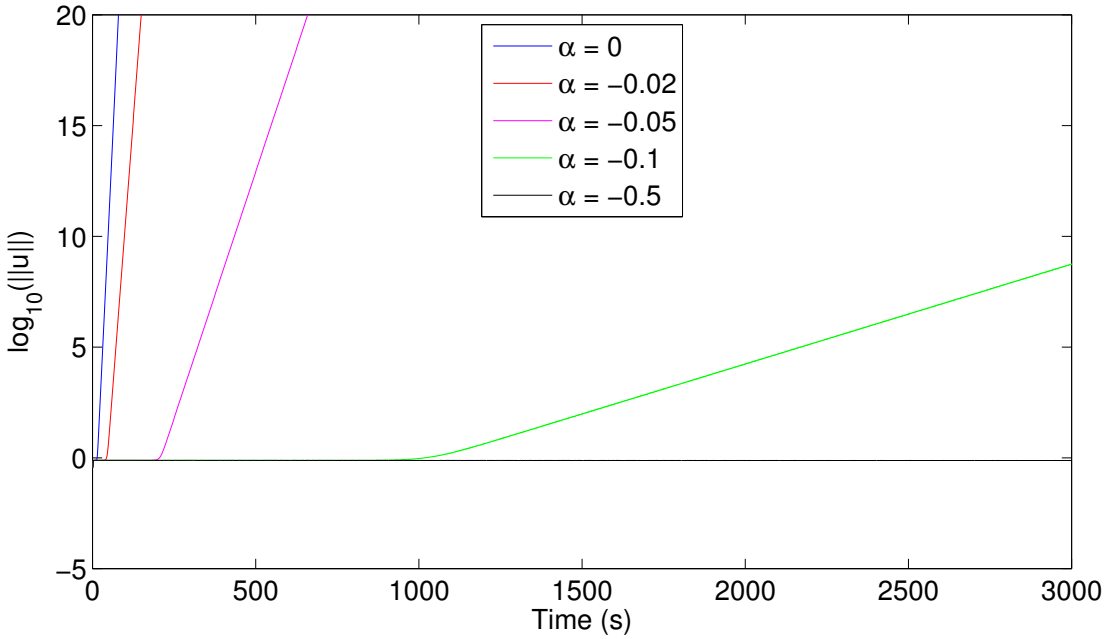


Fig. 7. Time evolution of the logarithm of the L^2 norm of the solution. Gauss-Lobatto points are used on a distorted mesh and different values of the penalization parameter are selected

was not observed until $T = 100\,000s$.

The behaviour of the deterioration of the CFL appears to depend on the time scheme, the mesh and the order of approximation, Let us denote α^* as the value at which the CFL begins to decrease. In Fig 6, α^* is approximately -0.5. When the mesh is refined or the order increased , α^* increases.

The effect of penalization is similar when using Gauss points instead of Gauss-Lobatto. The only difference is that Gauss points provide always stable solutions (e.g. the figure 7 is not observed for Gauss points) as proven theoretically in a previous theorem.

1.3. *Fast matrix-vector product*

Let us denote n the number of unknowns (4 for the 3-D aeroacoustic equations), N_{dof} the number of degrees of freedom on an hexahedron ($N_{\text{dof}} = (r+1)^3$) The row index is denoted (p, e, j) where e is the number of element and j the local number of degree of freedom inside the element, p the number of the unknown. The column index is denoted (q, e', k).

1.3.1. *Mass matrix*

Because of mass-lumping, the mass matrix is diagonal and we have

$$(M_h)_{(p,e,j),(q,e',k)} = \delta_{p,q}\,\delta_{e,e'}\,\delta_{j,k}\,J_e(\xi_k)\,\omega_k$$

1.3.2. *Stiffness matrix*

We can split the stiffness matrix in two parts: $K_h = R_h + S_h$
The volumic integral component in the stiffness matrix is:

$$\begin{aligned}(R_h)_{(p,e,j),(q,e',k)} &= \delta_{e,e'} \int_{K_e} \sum_i A_i^{p,q}\,\varphi_k\,\frac{\partial \varphi_j}{\partial x_i}\,dx \\ &\approx \delta_{e,e'} \sum_i \omega_k\,(J_e\,A_i^{p,q})(\hat{\xi}_k)\,\frac{\partial \varphi_j}{\partial x_i}(\xi_k) \\ &= \delta_{e,e'} \sum_{i,m} \omega_k[(\,J_e\,DF_e^{-1})_{m,i}\,A_i^{p,q}](\hat{\xi}_k)\,\frac{\partial \hat{\varphi}_j}{\partial \hat{x}_m}(\hat{\xi}_k)\end{aligned}$$

So, we choose to store only the $d \times d$ matrices $J_e\,DF_e^{-1}$, i.e. the values of $J_e\,DF_e^{-1}$ at each degree of freedom. The matrix-vector product $Y = R_h U$ is calculated "on the fly", for each element, in three steps :

$$(1)\ v_{i,p,k} = \sum_{q=1}^{n} A_i^{p,q}\,u_k^q, \quad i = 1..d,\ p = 1..n,\ k = 1..N_{\text{dof}}$$

$$(2)\ w_{m,p,k} = \omega_k \sum_{i=1}^{d} (J_e\,DF_e^{-1})_{m,i}(\hat{\xi}_k)\,v_{i,p,k}, \quad m = 1..d,\ p = 1..n,\ k = 1..N_{\text{dof}}$$

$$(3)\ y_{p,j} = \sum_{m=1}^{d} \sum_{k=1}^{Ndof} \frac{\partial \hat{\varphi}_j}{\partial \hat{x}_m}(\hat{\xi}_k)\,w_{m,p,k}, \quad p = 1..n,\ j = 1..N_{\text{dof}}$$

The first step applies the matrices A_i, the second step applies geometrical transformations and the final step performs the integration against derivatives of basis functions.
These steps can be seen as a factorization of matrix R_h as follows

$$R_h = \widehat{S}\,DF_h\,A_h$$

The multiplication by the matrices A_h, DF_h and $\widehat{S}$ corresponds to the first, second and third step of the previous algorithm. Matrix DF_h is a $d \times d$ block-diagonal matrix.
The calculational scaling of the two first steps is obviously in $O(r^3)$ whereas the last step

seems to scale as $O(r^6)$ because of the loop in N_{dof}. But that order is too high because matrix $\widehat{S}$ is very sparse on hexahedral elements due to the tensorization of the degrees of freedom. To compute each derivative, only $r+1$ values are needed, and not $(r+1)^3$ values. To illustrate this property, we decompose each index into its three coordinates :

$$j = (j_1, j_2, j_3), \quad k = (k_1, k_2, k_3)$$

The basis functions are tensorized as follows

$$\hat{\varphi}_j(\hat{x}_1, \hat{x}_2, \hat{x}_3) = \hat{\varphi}_{j_1}(\hat{x}_1)\, \hat{\varphi}_{j_2}(\hat{x}_2)\, \hat{\varphi}_{j_3}(\hat{x}_3)$$

The first term of the last step can be then written as

$$y_{p,j} \;=\; \sum_{k_1,k_2,k_3=1}^{r+1} \frac{\partial \hat{\varphi}_{j_1}}{\partial \hat{x}_1}(\hat{\xi}_{k_1})\, \hat{\varphi}_{j_2}(\hat{\xi}_{k_2})\, \hat{\varphi}_{j_3}(\hat{\xi}_{k_3})\, w_{1,p,k}$$

The basis functions are Lagrange functions with the fundamental property :

$$\hat{\varphi}_i(\hat{\xi}_j) = \delta_{i,j}$$

Thus, the triple sum is reduced to a single sum :

$$y_{p,j} \;=\; \sum_{k_1=1}^{r+1} \frac{\partial \hat{\varphi}_{j_1}}{\partial \hat{x}_1}(\hat{\xi}_{k_1})\, w_{1,p,(k_1,j_2,j_3)}$$

We clearly see that we need only $r+1$ terms to compute the derivative along x_1. Similar computations can be done for the derivatives along x_2 and x_3.

1.3.3. *"Jump" matrix*

We now show the method to calculate the "jump" matrix, i.e. matrix coming from boundary integrals of the variational formulation. For the sake of simplicity, we write only the "internal" part of the jump matrix (and not interactions between the elements).

$$S_{h(p,e,j),(q,e',k)} \;=\; \delta_{e,e'} \int_{\partial \hat{K}} ds_e \sum_i n_i\, A_i^{p,q}\, \hat{\varphi}_k\, \hat{\varphi}_j\, dx$$

where ds_e is the length of an edge in $2D$ or the surface of a face in $3D$.
After integration over quadrature points $\hat{\zeta}_m$ on the boundary, the matrix-vector product can be written as

$$(S_h\, U)_{p,j} \;=\; \sum_{m,q,k,i} \omega_m\, (n_i\, ds_e\, A_i^{p,q})(\hat{\zeta}_m)\, \hat{\varphi}_k(\hat{\zeta}_m)\, \hat{\varphi}_j(\hat{\zeta}_m)\, u_k^q$$

This can be split into three steps

(1) $v_{q,m} \;=\; \sum_k \hat{\varphi}_k(\hat{\zeta}_m)\, u_k^q$

$$(2)\ \ w_{p,m} \;=\; \omega_m \sum_{q,i} (n_i\, ds_e A_i^{p,q})(\hat{\zeta}_m)\, v_{q,m}$$

$$(3)\ \ y_{p,j} \;=\; \sum_{m} \hat{\varphi}_j(\hat{\zeta}_m)\, w_{p,m}$$

This also can be seen as the following factorization

$$S_h \;=\; \widehat{E}\, N_h\, \widehat{E}^*$$

Matrix $\widehat{E}$ computes the extrapolation needed to determine the value of U on quadrature points on edges.
When we use Gauss-Lobatto points, matrix $\widehat{E}$ is the identity matrix, because the Gauss-Lobatto points contain the two ends of the interval $[0, 1]$. Thus, we have the values on edges. In the case of Gauss points, we need to know the values on Gauss points on edges/faces. By using hexahedral mesh, matrix $\widehat{E}$ is very sparse because of the tensorization of the degrees of freedom.
In Fig 8, an example is shown for Q_2.

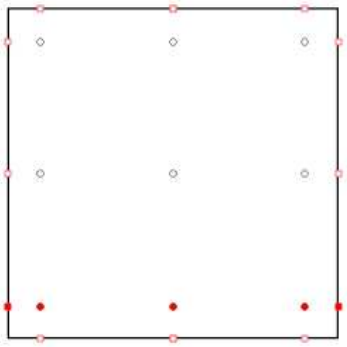


Fig. 8. Degrees of freedom necessary to compute extrapolation on the quadrature points of the edge.

1.4. *Computational complexity*

In order to compare hexahedral elements to tetrahedral elements, we show computational scaling. In the case of tetrahedral elements, one can write similar algorithms by using any type of interpolation points (it works only for straight elements, i.e. without curvature), without the mass lumping technique [3]. Again the notations are d the dimension, r the order of approximation, n the number of unknowns and N_{dof} the number of degrees of freedom inside an element.
For hexahedra, $N_{\text{dof}} = (r+1)^3$ and for tetrahedra $N_{\text{dof}} = \frac{(r+1)(r+2)(r+3)}{6}$
Matrices A_i are considered full, whereas in most applications, these matrices are sparse and require less operations.
All the costs detailed next, are costs for ONE element of the mesh.

Cost of the product by mass matrix : $N_{\text{dof}}\, n$ operations
Cost of the product by R_h for hexahedral elements : $2\,d\,n^2\,N_{\text{dof}} + 2\,d^2\,n\,N_{\text{dof}} + 2\,d\,n\,N_{\text{dof}}\,(r+1)$
Cost of the product by R_h for tetrahedral elements : $2\,d\,n^2\,N_{\text{dof}} + 2\,d^2\,n\,N_{\text{dof}} + 2\,d\,n\,N_{\text{dof}}^2$
Cost of the product by S_h for hexahedral elements (Gauss points) : $24\,n\,N_{\text{dof}} + 18\,d\,n^2\,(r+1)^2$
Cost of the product by S_h for hexahedral elements (GL points) : $18\,d\,n^2\,(r+1)^2$
Cost of the product by S_h for tetrahedral elements : $6\,d\,n^2\,r\,(r+1)$

The main economy realized by using hexahedral elements is for the evaluation of derivatives. For each point of quadrature, only $(r+1)$ values are needed to evaluate derivative along x. This results in a scaling of $O(r^4)$ (the greatest term is $2\,d\,n\,N_{\text{dof}}\,(r+1)$, the scaling N_{dof} is of $O(r^3)$). For tetrahedral elements all the N_{dof} values are needed to evaluate derivatives, resulting in scaling of $O(r^6)$. In Fig 9, we have displayed the number of operations needed

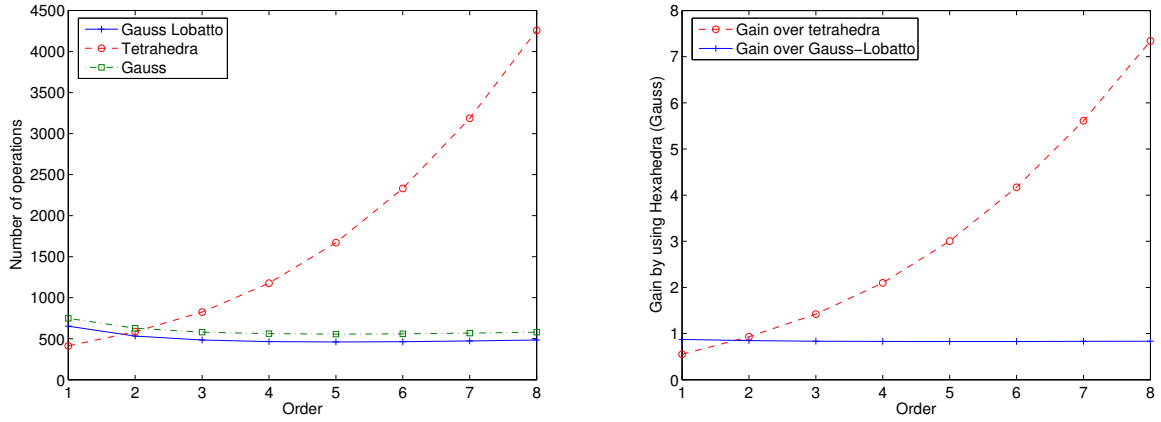


Fig. 9. Number of operations per dof for tetrahedral or hexahedral elements according the order of approximation. We selected $d = 3$ $n = 4$ (four unknowns for the aeroacoustic equations). At right, gain obtained by using Gauss points for hexahedral elements over tetrahedral elements or Gauss-Lobatto points.

by degree of freedom. The cost for hexahedral elements is almost constant versus the order of approximation, whereas it can become very large for tetrahedral elements. In particular example, for a Q_5 approximation, hexahedra are three times faster than tetrahedra.
For tetrahedral elements, the storage required is quite neglegible because the matrices DF_e are constant inside each element, whereas for hexahedral elements we need to store the values of DF_e on each degree of freedom, and also the normal vector on each face. It induces a storage of $9+\frac{9}{r+1}$ coefficients per degree of freedom. for the aeroacoustic system, for example $(n = 4)$, this storage is equivalent between two and three vectors. The additional storage induced by the use of hexahedral elements is minimal.

1.4.1. *Gauss or Gauss-Lobatto ?*

As described in the previous section, the Gauss-Lobatto points enable a faster matrix-vector product than Gauss points.

Order	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
Gauss points	0.50000	0.24748	0.15063	0.10164	0.07319	0.05518	0.04308
GL points	1.00000	0.39493	0.21966	0.14019	0.09686	0.07071	0.05377

Table 1. 1-D CFL numbers for Gauss or GL points, with no penalization ($\alpha = \delta = 0$), no flow and for a leap frog scheme

Another advantage of Gauss-Lobatto points is that the CFL number is higher than that of Gauss points. The CFL number in dimension d on a regular mesh is related to 1-D CFL number

$$CFL_{d\,D} = \frac{1}{\sqrt{d}} CFL_{1D}$$

We see from table 1, (extracted from [5]). that the CFL number for GL points of order $r+1$ is merely the same as the CFL for Gauss points of order r. Moreover, if a leap frog scheme is chosen, the penalty terms can be centered whereas it is not possible for Gauss points in an explicit scheme.

The main advantage of Gauss points is to produce a more accurate integration (integration of the stiffness matrix is exact for straight elements) and consequently more accurate solutions. This increase of accuracy has been shown by a dispersion analysis [5].

Finally, both Gauss points and Gauss-Lobatto points have advantages, but the choice of which to use is difficult. Only experiments can provide guidance as to the best choice and then only if there is the possiblity of clear benefit. We show a comparison in the next section.

2. Numerical results

2.1. *Non-uniform flows*

Let us recall that Vector U and matrices A_i are then equal to :

$$U = \begin{pmatrix} p \\ \rho_0\, v_1 \\ \rho_0\, v_2 \\ \rho_0\, v_3 \end{pmatrix} \qquad A_i = \begin{pmatrix} v_i & c_0^2 e_i \\ e_i & v_i \mathbb{I} \end{pmatrix}$$

For the expression of boundary conditions in a discontinuous Galerkin framework, please refer to [4], where Neumann and first-order absorbing boundary conditions are developed. The computation of boundary integrals is performed with the same method as for the "jump"

matrix.

The type of the time integration scheme is important, but not essential in this paper. So we have chosen to use an explicit, low-storage (2N) Runge-Kutta scheme for all the experiments [12]. For all the experiments, the meshes are produced by CUBIT, and isoparametric curved elements are used to correctly approximate the geometry.

2.2. *2-D computations*

The computational domain is a square $[0, 12]^2$ with an interior circular hole. A Neumann condition is set on the circular hole, and boundaries $y = y_{min}$ and $y = y_{max}$. On the other boundaries, an absorbing boundary condition is set. In order to obtain a stationary flow compatible with the Neumann condition, one can solve the following Poisson problem

$$\begin{cases} \Delta u &= 0 \\ \dfrac{\partial u}{\partial n} &= M \cdot n \text{ on } \Sigma \\ \dfrac{\partial u}{\partial n} &= 0 \text{ on } \Gamma \end{cases}$$

where Σ represents the external boundary to the computational domain, and Γ the interior boundaries. M is the velocity vector of the flow at infinity and taken equal to :

$$M = (0.5, 0)$$

The flow v_0 is then deduced from u by the relation

$$v_0 = \nabla u$$

Density ρ_0 and pressure p_0 are uniform so that the velocity c_0 is uniformly equal to 1. The obtained flow around the disk is displayed in Fig 10. The source function f is sinusoidal and is modulated by a gaussian function in time and in space (centered at the origin) as follows

$$\begin{cases} f(t;x,y) = & r(t)\, g(x,y) \\ r(t) & = \sin(2\pi t)\, exp(-\alpha(t-t_0)^2) \\ g(x,y) & = e^{\frac{-7(x^2+y^2)}{r_0^2}} \end{cases}$$

r_0 is the influence radius of the gaussian, and $\alpha = 0.25$. The experiment is simulated until $t = 15s$. In table 2, we compute the number of degrees of freedom and the number of operations needed to reach an error less than 2% for the solution at $t = 14s$. When referring to the number of degrees of freedom, multiply by the number of unknowns (three unknowns in 2-D) to get the size of a vector. Except for Q_2 and Q_3, the efficiency of the

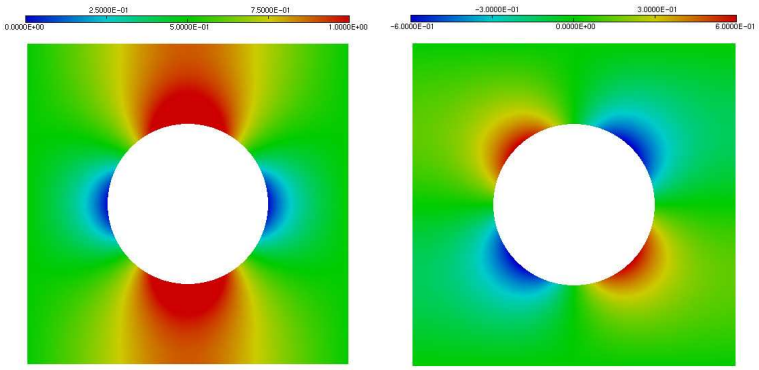


Fig. 10. Calculated flow around a disk. At left, component of the stationary flow along x, at right component along y.

Element	Q_2	Q_3	Q_4	Q_5	Q_6
Gauss	57 987 $\Delta t = 0.0138$ 69 GFlops	29 264 $\Delta t = 0.0173$ 27 GFlops	22 275 $\Delta t = 0.0168$ 21 GFlops	18 540 $\Delta t = 0.0167$ 18.2 GFlops	19 649 $\Delta t = 0.0145$ 23 GFlops
Element	Q_3	Q_4	Q_5	Q_6	Q_7
Gauss-Lobatto	52 016 Δt ,$= 0.0164$ 39.2 Gflops	31 525 $\Delta t = 0.0177$ 22.2 GFlops	25 416 $\Delta t = 0.0172$ 19.0 GFlops	23 128 $\Delta t = 0.0150$ 20.6 GFlops	20 608 $\Delta t = 0.0133$ 21.6 GFlops

Table 2. Number of dofs, time stepping and operations needed to reach an error less than 2% for the disk.

other orders shows little difference, thus there is no advantage between Gauss points and Gauss-Lobatto points.

Let us consider the flow around a NACA airfoil (see Fig 11). We see that near the two extremities of the airfoil, the flow is varying rapidly because of the high curvature. The airfoil geometry has been defined by a closed spline (the right end is curved, and is not a corner). Because the Neumann condition is imposed in a weak sense, this condition is not numerically satisfied, especially on the two ends of the airfoil. Hence, we need to modify the flow in order to satisfy this condition :

$$v_{0x}n_x + v_{0y}n_y + v_{0z}n_z = 0$$

The component modified is related to the maximum normal component. If this modification is not done, instabilities can appear. The computed solution is displayed in Fig 12. In this "difficult case", because of the sharp right extremity, Gauss-Lobatto points provide unstable algorithms, while Gauss points provide acceptable results. In table 3, we computed the

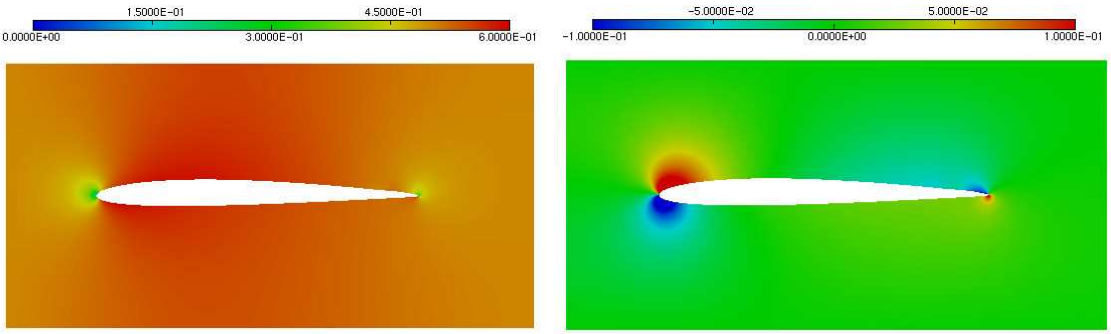


Fig. 11. At left, component of the stationary flow along x, at right component along y.

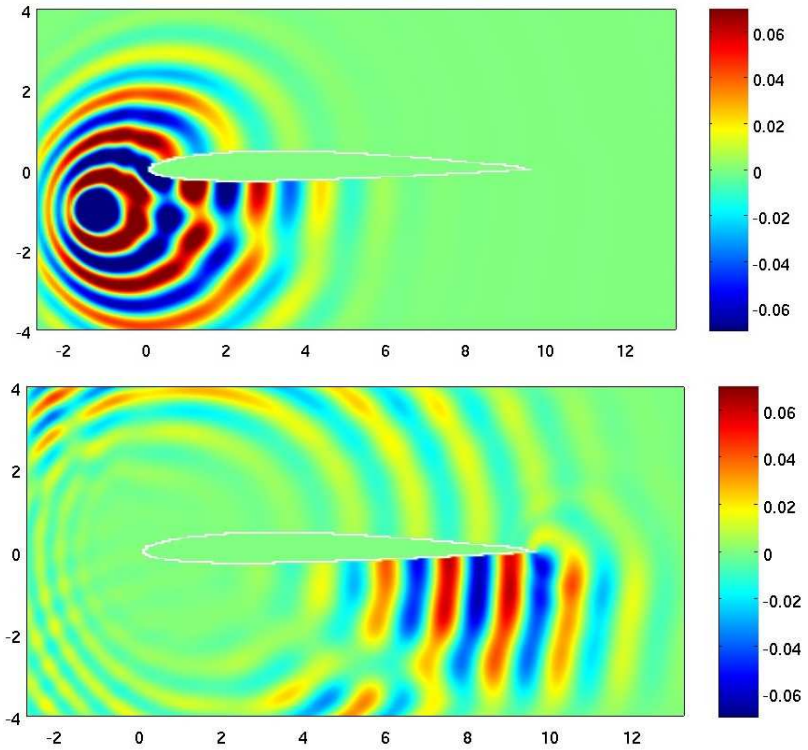


Fig. 12. Pressure at $t = 9s$ and $t = 14s$ for the NACA airfoil.

number of degrees of freedom and the number of operations needed to reach an error less than 2%, for the solution at $t = 14s$. In that case, Q_3 is the optimal order, with 14.7 GFlops. Because of the geometry, local refinement near the extremities is needed for approximations of an order greater than three, reducing computational speeed.. The mesh used for Q_6 is displayed in the Fig 13.

Order	Q_2	Q_3	Q_4	Q_5	Q_6
Dofs	27 792	18 000	15 500	13 788	14 602
Time Step	$\Delta t = 0.025$	$\Delta t = 0.0195$	$\Delta t = 0.0139$	$\Delta t = 0.0134$	0.0085
Operations	18.3 GFlops	14.7 GFlops	17.9 GFlops	16.9 GFlops	29.1 GFlops

Table 3. Number of dofs, time stepping and operations needed to reach an error less than 2% for the NACA airfoil.

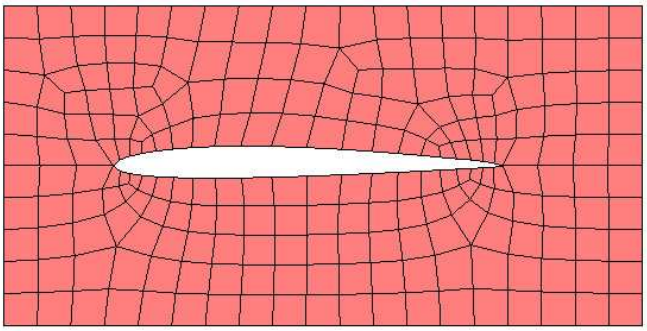


Fig. 13. Mesh used for Q_6 and NACA airfoil.

2.3. *3-D computations*

Another method to obtain a stationary flow, satisfying the condition $v \cdot n = 0$ on the surface of the objects, is to solve a Poisson problem for v_{0x}

$$\begin{cases} \Delta v_{0x} = 0 \\ v_{0x} = 0.5 \text{ on } \Sigma \\ v_{0x} = 0 \text{ on } \Gamma \end{cases}$$

where Σ represents the external boundary to the computational domain, and Γ the internal boundaries. The flow obtained for a spherical obstacle is displayed in Fig 14. On the sphere and the lateral sides of the external parallelipedic boundary, we impose a Neumann boundary condition. On the left and right side, a first-order absorbing boundary condition is set. Fig 15 shows the temporal evolution of the pressure. The temporal source is a sinusoid modulated by a gaussian function, like for the 2-D cases.

We will determine the minimal number of dofs needed to reach an error less than 5%. The time step, dofs, and number of operations are displayed in table 4. We observe a reduction by ten of the computational time, by using Q_7 instead of Q_3. We note that the complexity of Gauss points of order r is almost equivalent to the complexity of Gauss-Lobatto points of order $r + 1$. For example, Q_3 Gauss needs 31 TeraFlops while Q_4 Lobatto needs 26 Tflops and Q_4 Gauss needs 13.6 Tflops while Q_5 Lobatto needs 14.4 Tflops ... That is why the choice between the Gauss and Gauss-Lobatto points is not crucial. Nevertheless, we were not able to prove stability for Gauss-Lobatto points, and in practice such instabilities

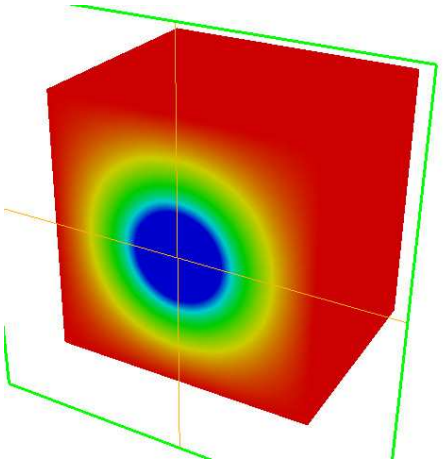


Fig. 14. Flow v_{0x} around the sphere.

Element	Q_3	Q_4	Q_5	Q_6	Q_7
Gauss	2.412 Mdof $\Delta t = 0.0112$ 31.23 TFlops	1.189 Mdof $\Delta t = 0.0123$ 13.55 TFlops	0.849 Mdof $\Delta t = 0.0128$ 9.22 TFlops	0.677 Mdof $\Delta t = 0.011$ 8.61 TFlops	0.623 Mdof $\Delta t = 0.0111$ 7.98 TFlops
Gauss-Lobatto	7.06 Mdof Δt ,$= 0.0105$ 81.3 Tflops	2.68 Mdof $\Delta t = 0.0119$ 26.16 TFlops	1.513 Mdof $\Delta t = 0.0121$ 14.39 TFlops	0.851 Mdof $\Delta t = 0.0129$ 7.64 TFlops	0.734 Mdof $\Delta t = 0.0124$ 6.99 TFlops

Table 4. Efficiency of the method for the sphere.

can be observed if we use this points for the unsplit formulation. A solution is to use the split formulation, which ensures the stability, whatever the quadrature formula. However, the split formulation induces additional computational time, that's why it seems better to use Gauss points, for which the stability (in the case of an uniform flow) is certified for the unsplit formulation. For non-uniform flows, we didn't observe any instability by using Gauss points with the unsplit formulation.

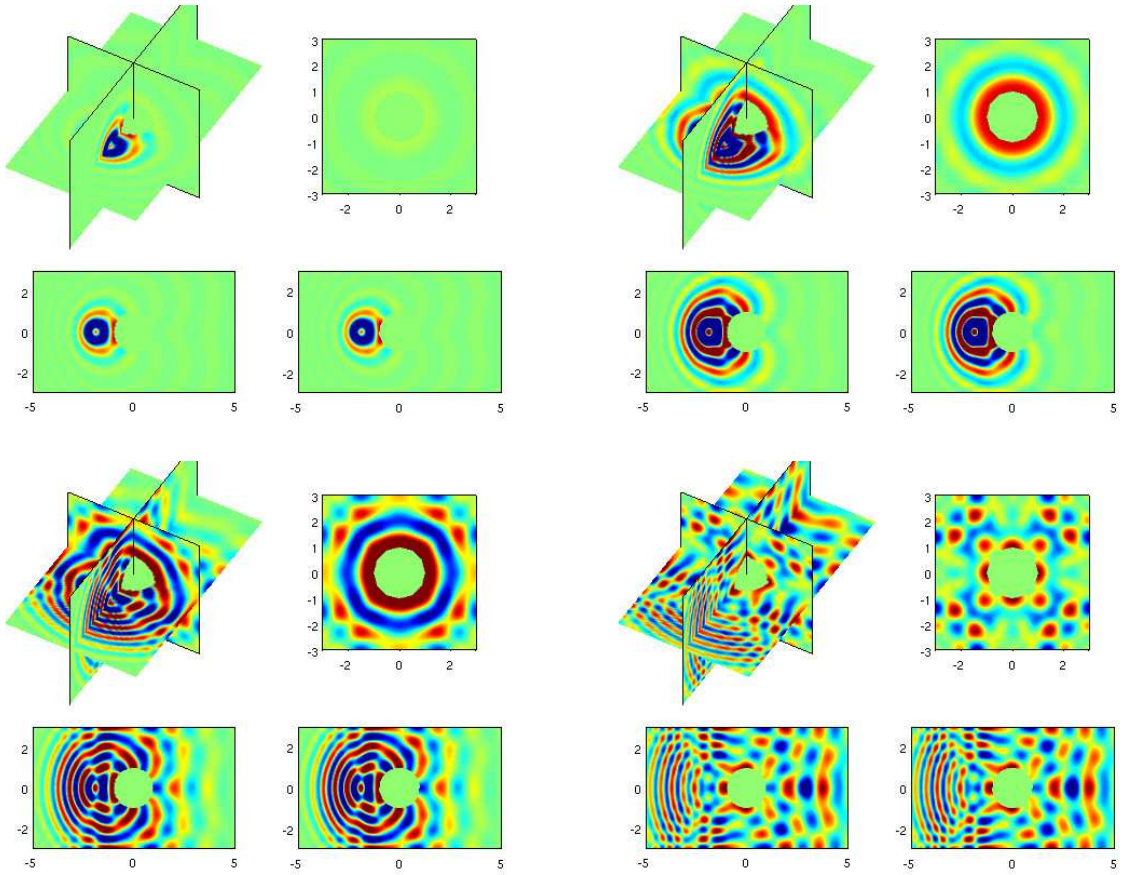


Fig. 15. Evolution of a gaussian source in space with a spherical obstacle. Snapshots at $t = 6s,\ 8s,\ 10s$ and $12s$.

[1] A-S Bonnet-Ben Dhia, E-M Duclairoir, G. Legendre and J-F Mercier, *Time-harmonic acoustic propagation in the presence of a shear flow* Journal of Computational and Applied Mathematics, 204(2), pp 428-439 (2007)

[2] M. Costabel and M. Dauge, *Weighted regularization of Maxwell equations in polyhedral domains*, Numer. Math. 93, pp 239-277 (2002)

[3] J.S. Hesthaven and T. Warburton, *High-Order Nodal Methods on Unstructured Grids. I. Time-Domain Solution of Maxwell's Equations*, J. Comput. Phys. 181(1), pp. 186-221 (2002)

[4] M. Bernacki, *Méthodes de type Galerkin discontinu pour la propagation des ondes en aéroacoustique*, PhD Thesis, Ecole Nationale des Ponts et Chaussées (2005)

[5] S. Pernet *Etude 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)

[6] P. Castillo, B. Cockburn, I. Perugia and D. Schoetzau, *An a priori error analysis of the local discontinuous Galerkin method for elliptic problems*, SIAM J. Numer. Anal. 38(5), pp. 1676-1706 (2000)

[7] P. Grob, M. Duruflé 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*, submitted to Numerical Methods for Partial Differential Equations

[8] A. H. Stroud, *Approximate calculation of multiple integrals*, Prentice-Hall Inc., Englewood Cliffs, N.J. (1971)

[9] O. C. Zienkiewicz, *The finite element method in engineering science*, McGraw-Hill, London (1971)

[10] M. Duruflé, *Intégration numérique et éléments finis d'ordre élevé appliqués aux équations de Maxwell en régime harmonique*, PhD Thesis, Université de Paris IX Dauphine (2006)

[11] C. Bogey, C. Bailly and D. Juvé *Computation of flow noise using source terms in linearized Euler's equations*, AIAA Journal 40(2), pp. 225-243 (2002)

[12] M. H. Carpenter and C.A. Kennedy, *A fourth-order 2N-storage Runge-Kutta scheme*, NASA TM 109112, June (1994)

[13] G. Cohen, M. Duruflé *Non Spurious Spectral-Like Element Methods for Maxwell's Equations*, Journal of Computational Mathematics, vol 25, pp 282-304 (2007)

[14] G. Cohen, *Higher-order Numerical Methods for Transient Wave Equations*, Springer Verlag (2002)

[15] G. Cohen, X. Ferrieres and S. Pernet, *A spatial high-order hexahedral discontinuous Galerkin method to solve Maxwell equations in time domain*, Journal of Computational Physics, vol 217(2), pp 340-363 (2006)

[16] M. Bernacki and S. Piperno, *A dissipation-free time-domain Discontinuous Galerkin method applied to three dimensional linearized Euler equations around a steady-state non-uniform inviscid flow*, J. Computational Acoustics, vol. 14, No. 4, pp. 445-467 (2006)