Universitá degli studi di Bergamo
Dipartimento di Ingegneria Industriale

Dottorato in Tecnologie per l'Energia e l'Ambiente
XX Ciclo

An h-Multigrid Approach for High-Order Discontinuous Galerkin Methods

Author:
Pietro Tesini

Supervisor:
Prof. Francesco Bassi

January 2008

“γίγνομαι μη γιγμόςκηιν”

Σωκράτης

Abstract

The thesis presents the development of an h-multigrid solver for high-order accurate discontinuous Galerkin (DG) discretizations of non-linear systems of conservations laws on unstructured grids.

For this purpose, a high-order DG discretization on polyhedral grids is developed first and it is applied to the compressible Navier-Stokes equations. The proposed method employs shape functions, consisting of complete polynomials defined in the real space, which are hierarchic and orthonormal on arbitrarily shaped elements. As regards the discretization of the viscous terms of the Navier-Stokes equations, we use the well-known method introduced in [6] with suitably enlarged values of the stability parameter reported in [1]. The accuracy and the convergence properties of the method have been tested against classical inviscid problems such as the transonic Ringleb flow and the subsonic flow over a Gaussian bump. The Helmholtz problem and the solution of the Navier-Stokes equations around a NACA0012 airfoil have been used as viscous tests.

Then we present the development of an h-multigrid method where coarse grid levels are constructed by agglomerating neighbouring elements of the fine grid. First, an elegant yet practical set of transfer operators is derived for general space settings and for the current one. After that, a quasi-implicit multistage h-multigrid iteration strategy for the discontinuous Galerkin discretization of the steady Euler equations is developed and numerically investigated. Results are presented for a subsonic flow over a NACA0012 airfoil at 2^o of incidence. These results highlight the main

properties of the developed multigrid scheme and its different behaviour with respect to the classic p-multigrid scheme.

Acknowledgments

I would like to thank the people that made this work possible. First I would like to thank my supervisor, Francesco Bassi, who showed me the main concepts behind DG methods. His ideas drove me throughout this research, which is, in my opinion, new and promising. In addition, I'm thankful to Andrea Crivellini, not only for his contribute but also for his friendship, to Antonio Ghidoni, for all interesting information concerning p-multigrid methods, and to Daniele Di Pietro, for introducing me to C++ language.

Special thanks to all my friends, my brother, my sisters and my parents for supporting me during the last three years. Mind is not divisible indeed, thus non-academic life has a strong influence on academic results.

This work was done in the context of ADIGMA project, "*A European project on the development of adaptive high order variational methods for aerospace applications*"

Contents

Chapter 1

Introduction

1.1 Motivation

Modern Computational Fluid Dynamics (CFD) simulations require models that provide a faithful representation of flow physics and high-order of accurate algorithms. In this context, high-order spatial discretizations can improve the predictive capabilities of simulations in many applications. This is due to the fact that the higher is the discretization order the faster is the asymptotic convergence rate of the error. For example, fourth-order accurate spatial discretisations reduce the error by a factor of 2^4 each time the mesh resolution is doubled, while, using a second-order accurate spatial discretisation[1], the error is reduced only by a factor 2^2. Since a doubling of mesh resolution entails an increase of overall work by a factor of 4 in $2D$ and of 8 in $3D$, achieving an arbitrarily prescribed error tolerance with second-order accurate methods in $3D$ can quickly become unfeasible. Thus, for increasingly higher accuracy high-order methods ultimately become the

[1] Actually most currently employed CFD algorithms are asymptotically second order-accurate in space. These are usually second order finite volume methods, which are constructed by employing a second order reconstruction. Going further to a third order reconstruction on unstructured meshes is very cumbersome or even virtually impossible.

method of choice.
Among high-order methods, discontinuous Galerkin methods have become increasingly popular in several branches of numerical physics, owing to their sound mathematical foundation and to their easiness of implementation.
The order of DG methods, applied to problems with regular solutions, depends on the degree of the polynomial approximation which can be easily increased, dramatically simplifying the use of higher order methods on unstructured grids. Furthermore, the stencil of most DG schemes is minimal in the sense that each element communicates only with its direct neighbours. In contrast to the increasing number of elements or mesh points communicating for increasing accuracy of finite volume methods, the inter-element communication of DG methods is the same for any order. The compactness of the DG method has clear advantages in parallelization, which does not require additional element layers at partition boundaries. Also due to simple communication at element interfaces, elements with "hanging nodes" can be treated just as easily as elements without "hanging nodes", a fact that simplifies local mesh refinement[2] and the use of polyhedral elements. In addition to this, the communication at element interfaces is identical for any order of the method which simplifies the use of methods of differing orders in adjacent elements[3].
Unfortunately straightforward implementations of DG methods can readily become unacceptably costly.
Because of the discontinuities, the DG approximation is doubly valued at the interior cell boundaries, therefore, for a regular mesh and a given order of accuracy, we solve a larger system of discrete equations, compared with the classical finite element method. This leads to some inefficiency when simple smooth functions are approximated.
Moreover, due to the stringency of the CFL constraint, explicit Runge-Kutta schemes for time integration require too many time steps for convergence. Conversely, due to the large amount of degrees of freedom, fully implicit schemes for time integration require a very large amount of com-

[2] A clear advantage in h-adaptivity
[3] A clear advantage in p-adaptivity (polynomial degree adaptation)

putational resources in order to achieve a prescribed level of accuracy. Therefore the development of optimal, or nearly optimal solution strategies for DG discretizations, including steady-state solution methodologies, remains one of the key determining factors in devising high-order methods which are not just competitive but superior to low-order methods in overall accuracy and efficiency.

1.2 Background

Recent works have examined the use of spectral multigrid methods, where convergence acceleration is achieved through the use of coarse levels constructed by reducing polynomial order inside the element (as opposed to coarsening the mesh) for discontinuous Galerkin discretizations [11],[5],[12]. Then the method was improved by implicit multilevel techniques for high-order discretizations [9] and combined to h-multigrid at $p = 0$ level [13], [14]. Furthermore, in [13] and [14] is developed a solution algorithm which delivers convergence rates which are independent of p (the polynomial degree inside each element) and independent of h (the degree of mesh resolution), making use of p-coarsened and h-coarsened multigrid levels.

In the following it is presented an alternative multigrid approach for DG methods. Indeed, making use of shape functions defined in the real space, it is possible to devise a multigrid method, where convergence acceleration is achieved through the use of coarse levels constructed by agglomerating neighbouring elements (coarsening the mesh) and keeping unchanged the polynomial degree.

Chapter 2

Discontinuous Galerkin discretization on polyhedral grids

Further it is presented a simple yet smart way to obtain high-order discontinuous Galerkin discretizations of hyperbolic equations on polygonal grids. As a matter of fact, once proper shape functions and integration rules are defined, the aimed discretization will be straightforward and it will not require any additional theoretical consideration, in respect of DG discretizations on "classic" grids.

2.1 Introduction

Let us consider the numerical solution of any hyperbolic PDE in the conservative form,

$$\partial_t \mathbf{u} + \boldsymbol{\nabla} \cdot \mathbf{f}(\mathbf{u}) = 0 \quad \text{in} \quad \Omega \times [0, t], \tag{2.1}$$

where $\Omega \subset \mathbb{R}^2$, $\mathbf{u}$ is the state vector and $\mathbf{f}(\mathbf{u})$ is the vector flux function. In order to discretize eq. (2.1) in space it is necessary to provide both a set of suitable shape functions on the domain Ω and a means of evaluating the

weak-form integrals on the domain.

Firstly Ω_h, approximation of Ω, is partitioned into an ensemble of non-overlapping elements K_{ie}, which is called $\mathcal{T}_h = \{K_{ie}\}_{ie=1}^{ne}$, triangulation of Ω_h. Then, the solution is approximated on $\mathcal{T}_h$ as piecewise polynomial function possibly discontinuous on element interfaces. As a matter of fact, in discontinuous Galerkin space discretization, the global discrete solution $\mathbf{u}_h(\mathbf{x}, t)$, $\mathbf{x} \in \Omega_h$, has no continuity requirements at element interfaces, *i.e.* we assume the following space settings :

$$\mathbf{u}_h \in \mathbf{V}_h \stackrel{\text{def}}{=} [\, V_h \,]^N \,, \quad N = \text{number of unknowns}, \tag{2.2}$$

where

$$V_h \stackrel{\text{def}}{=} \left\{ v_h \in L^2(\Omega_h) : \; v_h|_{K_{ie}} \in \mathbb{P}_k(K_{ie}) \; \forall K_{ie} \in \mathcal{T}_h \right\} \,,$$

being $\mathbb{P}_k(K_{ie})$ the space of polynomials of global degree at most k on the element K_{ie}. In other words, the DG approach results in solution approximations which are local, discontinuous, and doubled valued on each elemental interface. So that any discrete solution $\mathbf{u}_h$ can be decomposed on

$$\left\{ \{ b_{iv,\,ie} \}_{iv=1}^{nv} \right\}_{ie=1}^{ne} , \quad \text{where} : \quad \begin{array}{rcl} nv & = & (k+1)(k+2)/2 \\ ne & = & \text{number of elements} \end{array} \,, \tag{2.3}$$

basis of $\mathbf{V}_h$ space.
The discontinuous space $\mathbf{V}_h$ implies that $b_{iv,\,ie}(\mathbf{x}) = 0 \; \forall \mathbf{x} \notin K_{ie}$ and that $\mathbf{u}_h$, on any element K_{ie}, can be written as :

$$\mathbf{u}_h(\mathbf{x}, t)|_{K_{ie}} = \sum_{iv=1}^{nv} \mathbf{U}_{iv,\,ie}(t) \; b_{iv,\,ie}(\mathbf{x}) \qquad \forall \mathbf{x} \in K_{ie} \tag{2.4}$$

The potential of geometrical flexibility of DG methods, relying on the lack of basis continuity requirements on elemental interfaces, can be exploited in many ways.

A set of modal, hierarchical and orthonormal basis functions is useful to

construct very simple restriction and prolongation operators for p-multigrid methods, [13] [14].

Lagrangian basis functions, with the interpolation nodes located in the cubature nodes, can be employed to implement a collocation method, reducing computational time needed to evaluate integrals and simplifying the form of the jacobian matrix, [20] [21].

Hereinafter it will be presented a further extension of the potential geometrical flexibility of the discontinuous Galerkin space discretization. Indeed, it is possible to derive the discretization of eq. (2.1) over a triangulation of non-overlapping polyhedral elements. As already pointed out, the space discretization "ingredients" are a proper set of shape functions and a means of evaluating weak-form integrals on the domain.
In conventional finite element methods, polynomial shape functions are usually defined on quadrangular and triangular canonical elements, then the discretization of eq. (2.1) is derived in the reference space by means of the transformations from canonical elements to the real ones. Conversely, when dealing with polygonal elements, it is impossible to define a canonical element, hence polynomial shape functions should be defined directly in the real space.
Furthermore optimal cubature rules cannot be derived on generic polygons and polyhedra. Then each element is partitioned into sub-regions of conventional type and the integration node locations are taken to be the mapping in the real space of sub-region cubature nodes. Aiming to computational efficiency, the use of reduced integration rules in sub-elements was investigated.

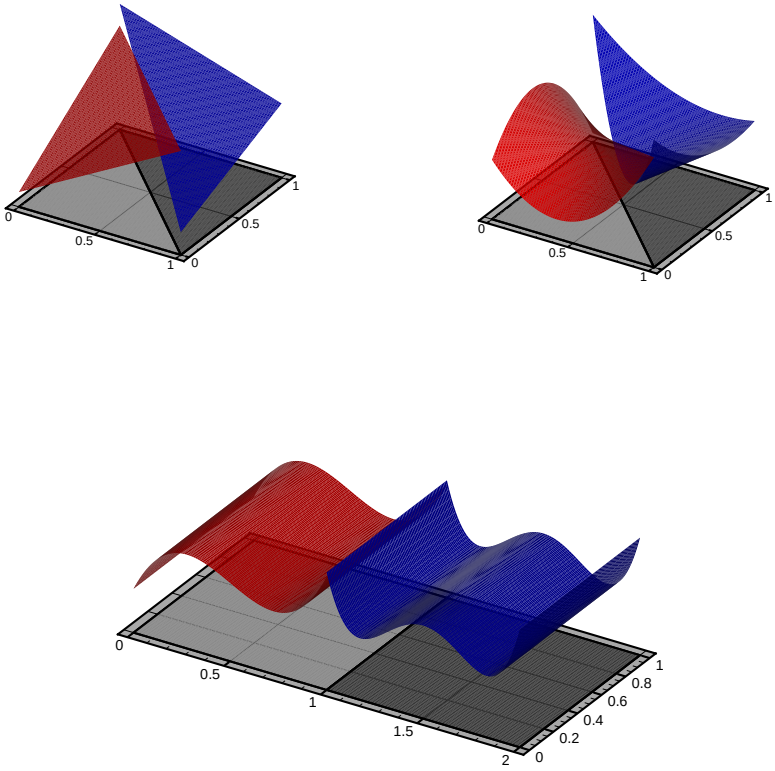


Figure 2.1: Discontinuous discretizations in two neighbouring elements

2.2 Basis functions

Basis functions are defined as complete polynomials in the real space. A modal hierarchic set of basis functions can be defined on any polyhedral element K_{ie} as :

$$b_{iv}(\mathbf{x}) = \frac{m_{iv}(\mathbf{x})}{s_{m_{iv}}}, \tag{2.5}$$

where :

- $m_{iv}(\mathbf{x})$ refers to the iv^{th} component of an ordered list of monomials $m_{\mathbb{P}_k}$ which is defined by choosing the space of polynomials $\mathbb{P}_k$. For example, if $\mathbb{P}_2$ is the space of polynomials of degree at most 2 in $2D$, $m_{\mathbb{P}_2}(\mathbf{x}) = \{1,\, x_1,\, x_2,\, x_1^2,\, x_1x_2,\, x_2^2\}$
- $\mathbf{x} = (x_1,\, x_2)$ are coordinates in a local reference frame defined by the element principal axes. The origin of the reference frame is located in the element center of mass,

  $$\int_{K_{ie}} x_1 \, \mathrm{d}\mathbf{x} = 0, \qquad \int_{K_{ie}} x_2 \, \mathrm{d}\mathbf{x} = 0, \tag{2.6}$$

  and frame axes are rotated in order to diagonalize the element inertia tensor,

  $$\int_{K_{ie}} x_1 x_2 \, \mathrm{d}\mathbf{x} = 0 \tag{2.7}$$

- $s_{m_{iv}}$ is a scaling factor defined as

  $$s_{m_{iv}} = \sqrt{\int_{K_{ie}} m_{iv}^2(\mathbf{x}) \, \mathrm{d}\mathbf{x}} \tag{2.8}$$

The choice of element principal axes to define the local reference frame produces a diagonal mass matrix for $\mathbb{P}_1$ space discretizations, for (2.6) and (2.7). Moreover, the scaling $s_{m_{iv}}$ insures that all diagonal terms of the mass matrix equal one,

$$\mathbf{M}_{i,\,i} = 1 \quad \forall i \in \mathbb{N}_{(1,\,nv)}{}^{1}\,, \tag{2.9}$$

[1] Notation $\mathbb{N}_{(1,\,nv)}$ indicates the set of natural numbers from 1 to nv.

and that all the off-diagonal terms of the mass matrix satisfy the relation

$$|\mathbf{M}_{i,j}| \leq \sqrt{\mathbf{M}_{i,i}\mathbf{M}_{j,j}} = 1 \quad \forall i,j \in \mathbb{N}_{(1,nv)} \,. \tag{2.10}$$

The previous relation is derived by means of the Cauchy-Schwartz inequality,

$$|\mathbf{M}_{i,j}| = \left| \frac{\int_{K_{ie}} m_i m_j \, \mathrm{d}\mathbf{x}}{s_{m_i} s_{m_j}} \right| \leq \sqrt{\frac{\int_{K_{ie}} m_i^2 \, \mathrm{d}\mathbf{x}}{s_{m_i}^2} \frac{\int_{K_{ie}} m_j^2 \, \mathrm{d}\mathbf{x}}{s_{m_j}^2}} = \sqrt{\mathbf{M}_{i,i}\mathbf{M}_{j,j}} \,.$$

In practice, using grids of polyhedral elements with low aspect ratios[2], it can be observed that many off-diagonal terms of $\mathbf{M}$ respect the relation $|\mathbf{M}_{i,j}| \ll \mathbf{M}_{j,j}$, effecting $\mathbf{M}$ to be strictly diagonally dominant[3].
Then the set of basis functions described so far produce a well-conditioned mass matrix for approximations on elements with low aspect ratios. In facts the condition number for a real, symmetric and positive definite matrix $\mathbf{A}$ reads

$$K_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 = \frac{\lambda_{\max}}{\lambda_{\min}} \,. \tag{2.11}$$

Furthermore the Gershgorin theorem states that the eigenvalues of $\mathbf{A} \in \mathbb{R}^{n \times n}$ belong to S_R, which reads

$$S_R \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} R_i, \qquad R_i = \{z \in \mathbb{R} : |z - \mathbf{A}_{i,i}| \leq \sum_{j \neq i}^{n} |\mathbf{A}_{i,j}|\}, \tag{2.12}$$

[2]The elemental aspect ratio, χ, is defined as

$$\chi_{2D} = \frac{l^2}{s} \qquad \text{or} \qquad \chi_{3D} = \frac{S^{\frac{2}{3}}}{V},$$

where l, s, S and V are respectively the perimeter, the surface, the boundary surface and the volume of an element. Then, minimizing aspect ratios is equivalent to constructing elements which are circular symmetric (or spherical symmetric in $3D$) about their barycenters.

[3]A square matrix $\mathbf{A}$ is called diagonally dominant if $|\mathbf{A}_{i,i}| \geq \sum_{j \neq i} |\mathbf{A}_{i,j}|$ for all i. $\mathbf{A}$ is called strictly diagonally dominant if $|\mathbf{A}_{i,i}| > \sum_{j \neq i} |\mathbf{A}_{i,j}|$ for all i.

where R_i are called Gershgorin circles. Then, an upper bound for mass matrix condition number reads

$$K_2(\mathbf{M}) = \frac{\lambda_{\max}}{\lambda_{\min}} \leq \frac{1+\varepsilon}{1-\varepsilon}, \tag{2.13}$$

where

$$\varepsilon = \max_{1 \leq i \leq nv} \left\{ \sum_{j \neq i}^{nv} |\mathbf{M}_{i,j}| \right\} . \tag{2.14}$$

Conversely, these basis functions yield an ill-conditioned mass matrix for high polynomial degree approximation on elements with curved sides and high aspect ratio.

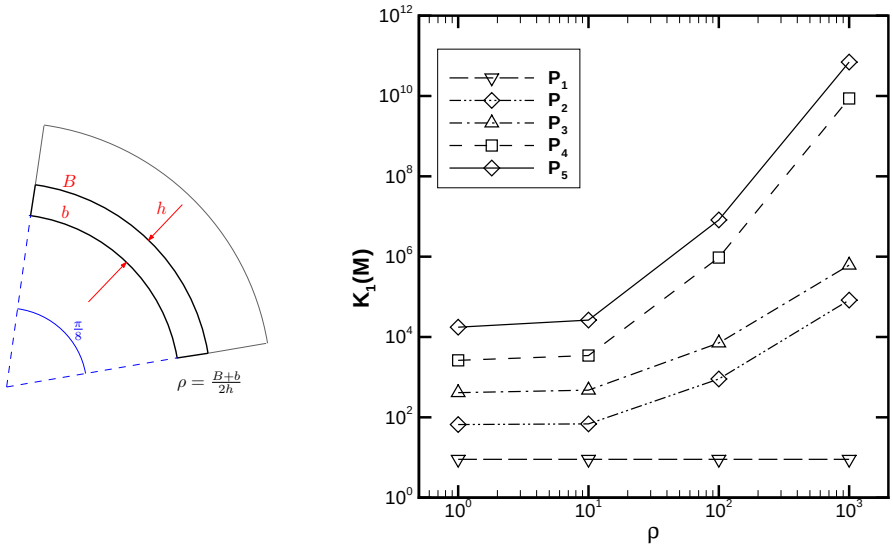


Figure 2.2: Mass matrix condition number for different ratios and polynomial degrees

In facts, considering a quadrilateral whose sides approximate a circumference arc of $\pi/8$, the mass matrix condition number increases when rising

either the elemental aspect ratio or the polynomial degree of the approximation (figure 2.2). Unfortunately in many aerodynamic applications the use of this kind of elements is required. For example, in viscous cases at high Reynolds number, in order to approximate correctly the boundary layer over a NACA airfoil, boundary elements are constructed curved and stretched.

The set of basis functions, described so far, can be further improved by means of a numerical procedure, see section 2.5. However, the higher is the condition number of elemental mass matrices, associated to starting sets of shape functions, the more detrimental is the effect of round-off errors in that numerical process. Hence, previous considerations play a leading role in further sections too.

2.3 Integration

In one dimension, the notion of optimality of an integration rule can be clearly defined. Cubature rules on quadrilaterals and hexahedra are generally obtained by tensor product of $1D$ quadratures, instead cubature rules on triangles, tetrahedrons, prisms and pyramids can be obtained from the previous cubatures by using a proper mapping (see appendix A). On generic polygons or polyhedra, however, no parallel notion of optimality can be formulated, then a different approach has to be considered.
Integration on the element boundary is performed computing separately the contribute of all faces, which are of conventional type (edges in $2D$), and then summing them together. The same idea is exploited to integrate on generic polygons. Indeed, the more natural solution lies in partitioning the element into n_{se} sub-regions of conventional type. Then, integration node locations are taken to be the mapping in the real space of sub-region cubature nodes and integration node weights equal the corresponding sub-region cubature weights (figure 2.3). Thus, integration rules on generic

polyhedral elements are derived as follows :

$$\begin{aligned}\int_K p\,(\mathbf{x})\ \mathrm{d}\mathbf{x} &= \\ &= \sum_{K_{se}\in K}\int_{K_{se}} p\,(\mathbf{x})\ \mathrm{d}\mathbf{x} \\ &= \sum_{K_{se}\in K}\int_{\widehat{K}_{se}} p\,(\widehat{T}_{se}(\widehat{\mathbf{x}}))\ |J_{\widehat{T}_{se}}|\ \mathrm{d}\widehat{\mathbf{x}} \qquad (2.15)\\ &= \sum_{K_{se}\in K}\sum_{i=1}^{g_{se}} p\,(\widehat{T}_{se}(\widehat{\mathbf{x}}_{i,\,se}))\ |J_{\widehat{T}_{se}}|\ w_{i,\,se}\end{aligned}$$

where :

- $\mathbf{x}$ and $\widehat{\mathbf{x}}$ refer to coordinates in real and reference spaces, respectively
- K_{se} is a sub-element belonging to K
- $\widehat{T}_{se}$ is the mapping of the canonical element $\widehat{K}_{se}$ onto the physical sub-element K_{se} and $|J_{\widehat{T}_{se}}|$ is the related jacobian
- $\widehat{T}_{se}(\widehat{\mathbf{x}}_{i,\,se})$ is the location of the i^{th} cubature node of K_{se} in the real space and $w_{i,\,se}$ is the corresponding weight

2.4 Reduced integration rules

Integration on polyhedral elements can be very expensive. Indeed, eq. (2.15) shows that integrating $p\,(\mathbf{x})$ in the real space is equivalent to integrating $p\,(\widehat{T}_{se}(\widehat{\mathbf{x}}))\ |J_{\widehat{T}_{se}}|$ in the reference space. Then, computing terms of elemental mass matrices requires cubature rules which integrate polynomials of degree

$$2\,k\,k_{\widehat{T}_{se}} + k_{|J_{\widehat{T}_{se}}|}\ ^{4} \qquad (2.16)$$

[4]Where $k\,k_{\widehat{T}_{se}}$ is the degree of $p\,(\widehat{T}_{se}(\widehat{\mathbf{x}}))$ in the reference space.
In facts, $p\,(\mathbf{x}) \in \mathbb{P}_k(\mathbf{x})$ and $\widehat{T}_{se}(\widehat{\mathbf{x}})$ is a polynomial which belongs either to $\mathbb{P}_{k_{\widehat{T}_{se}}}(\widehat{\mathbf{x}})$,

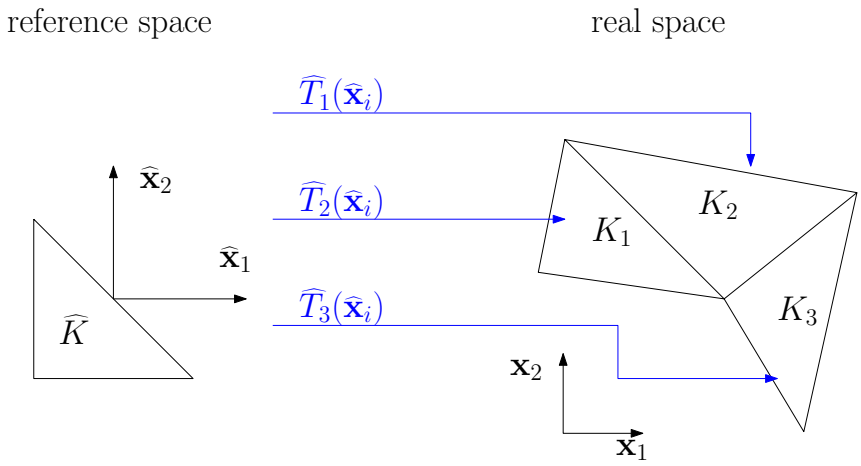


Figure 2.3: Any reference space cubature node $\widehat{x}_i$ is mapped in real space sub-regions by means of the proper transformation $\widehat{T}_j$

on the canonical element, for each sub-element K_{se} belonging to the polygonal element K.

The computational cost of integration on polygonal elements becomes unacceptably large when increasing either the order of the geometrical approximation $k_{\widehat{T}_{se}}$ or the number of sub-elements $K_{se} \in K$. Therefore, in order to improve computational efficiency, it was derived an adaptive criterion to reduce integration rules for the fine elements composing a polyhedral element.

---

for triangles, or to $\mathbb{Q}_{k_{\widehat{T}_{se}}}(\widehat{\mathbf{x}})$, for quadrangles, where $k_{\widehat{T}_{se}}$ is the order of geometrical approximation Ω_h. Thus,

$$p\left(\widehat{T}_{se}(\widehat{\mathbf{x}})\right) \in \begin{cases} \mathbb{P}_k\left(\mathbb{P}_{k_{\widehat{T}_{se}}}(\widehat{\mathbf{x}})\right) = \mathbb{P}_{k\,k_{\widehat{T}_{se}}}(\widehat{\mathbf{x}}) & \text{triangles} \\ \mathbb{P}_k\left(\mathbb{Q}_{k_{\widehat{T}_{se}}}(\widehat{\mathbf{x}})\right) = \mathbb{Q}_{k\,k_{\widehat{T}_{se}}}(\widehat{\mathbf{x}}) & \text{quadrangles} \end{cases} .$$

(refer to appendix B)

Aiming to integrate accurately the terms of elemental mass matrices, we define, for any product of shape functions[5] , the integration error

$$E_{iv,\,jv}(K) = \frac{|I_{K,\,approx}(\varphi_{iv}\varphi_{jv}) - I_{K,\,exact}(\varphi_{iv}\varphi_{jv})|}{\|\varphi_{iv}\|_{L_2(K)}\ \|\varphi_{jv}\|_{L_2(K)}}\,, \tag{2.17}$$

which, for a set of shape functions scaled by their L_2 norms, as in (2.5), reads :

$$E_{iv,\,jv}(K) = |I_{K,\,approx}(\varphi_{iv}\varphi_{jv}) - I_{K,\,exact}(\varphi_{iv}\varphi_{jv})|\ , \tag{2.18}$$

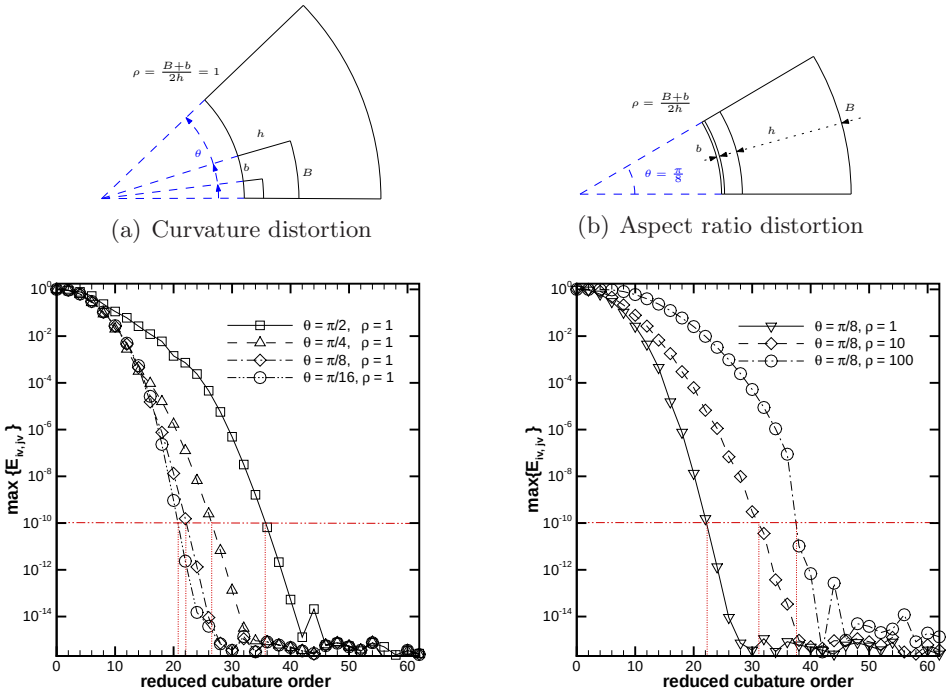
where $I_{K,\,exact}(\cdot)$ and $I_{K,\,approx}(\cdot)$ refer to integrals computed by means of cubatures which integrate, on the reference element $\widehat{K}$, polynomials of degree equal to (2.16) and less than (2.16), respectively. Then we reduce the integration rule until the condition

$$\max_{\substack{1\le iv\le nv\\ 1\le jv\le nv}} \{E_{iv,\,jv}(K)\} \le tol \tag{2.19}$$

is satisfied.
The inequality (2.19) and the definition (2.17) set up a criterion to reduce integration rules in conventional elements. For example, we consider a set of polynomial functions, as defined in (2.5), which are a basis for $\mathbb{P}_{10}(K)$ space, where K is a quadrangle whose sides are approximated by cubic polynomials. According to (2.16) it is necessary to use a cubature which integrates a polynomial of order 65 $(= 2 \cdot 10 \cdot 3 + 5)$, but, even admitting a little error ($tol = 10^{-10}$), the integration rule can be significantly reduced (figure 2.4). Moreover, as it was expected, figure 2.4 shows that the more "distorted" is the geometry the higher is the requested cubature order.

The inequality (2.19) controls the integration error on the whole element K. Conversely, in order to deal with polygons, integration errors and their

[5]The notation $\{\varphi_{iv}\}_{iv=1}^{nv}$ refers to any set of polynomials which are a basis for $\mathbb{P}_k(K_{ie})$ space.



(a) Curvature distortion (b) Aspect ratio distortion

(c) θ effect (d) ρ effect

Figure 2.4: Integration error versus cubature order in curved quadrangles

upper bounds should be defined locally to each sub-region $K_{se} \in K$. Thus, "exploding" twice the left term of (2.19) by means of the triangle inequality

$$\max_{\substack{1\le iv\le nv\\ 1\le jv\le nv}} \{E_{iv,\,jv}(K)\} \;\le\; \max_{\substack{1\le iv\le nv\\ 1\le jv\le nv}} \left\{ \sum_{K_{se}\in K} E_{iv,\,jv}(K_{se}) \right\} \;\le$$

$$\le \sum_{K_{se}\in K} \left(\max_{\substack{1\le iv\le nv\\ 1\le jv\le nv}} \{E_{iv,\,jv}(K_{se})\} \right), \qquad (2.20)$$

we extract from the global integration error in K the local contributes related to any sub-element $K_{se} \in K$. Then any local contribute to the

global integration error is bounded by a suitable portion of the admitted error

$$\max_{\substack{1\le iv\le nv\\ 1\le jv\le nv}} \{E_{iv,\,jv}(K_{se})\} \le \frac{\|\varphi_{iv}\|_{L_2(K_{se})}\|\varphi_{jv}\|_{L_2(K_{se})}}{\|\varphi_{iv}\|_{L_2(K)}\|\varphi_{jv}\|_{L_2(K)}}\; tol \quad \forall K_{se} \in K. \quad (2.21)$$

Finally, we note that (2.21) implies

$$\max_{\substack{1\le iv\le nv\\ 1\le jv\le nv}} \{E_{iv,\,jv}(K)\} \le \sqrt{n_{K_{se}}}\; tol\;, \quad (2.22)$$

where $n_{K_{se}}$ in the number of $K_{se} \in K$, which equals (2.19) if $n_{K_{se}} = 1$.

Some results concerning the use of reduced integration rules on polyhedral elements are presented in section 3.6. The effect of reduced integration rules is tested in terms of accuracy and CPU time.

2.5 Hierarchic orthonormal shape functions

In section 2.2 we described a modal hierarchic set of basis functions which could be defined on any polyhedral element. That set is well-suited for many applications, nevertheless the use of hierarchic orthonormal shape functions is preferable. In facts, in that case, mass matrices are identity-matrices, which are associated to optimal condition numbers, and h-multigrid transfer operators get two interesting properties, as discussed in section 4.1.4. Hereinafter, effects of numerical procedures, deriving an orthonormal set of shape functions from any starting set, are investigated. While the implemented numerical procedure will be presented later on in detail.

A set of shape functions $\{\varphi_{iv}\}_{iv=1}^{nv}$ is orthonormal if it verifies :

$$\int_K \varphi_{iv}\,\varphi_{jv}\;\mathrm{d}\mathbf{x} = \delta_{iv,\,jv} \qquad \text{for} \quad \forall iv, jv \in \mathbb{N}_{(1,\,nv)}\,, \quad (2.23)$$

where $\delta_{iv,\,jv}$ is the Kronecker delta. Property (2.23) involves integrals of shape functions products, then, if we associate to each shape function a vector whose components are its evaluations in integration nodes

$$\boldsymbol{\phi}_{iv} = [\ \ldots\ \varphi_{iv}(\mathbf{y}_j)\ \ldots\ ]^{\mathrm{T}} \quad \text{with} \quad \{\mathbf{y}_j\}_{j=1}^{g_K} \stackrel{\text{def}}{=} \left\{ \{\widehat{T}_{se}(\widehat{\mathbf{x}}_{i,\,se})\}_{i=1}^{g_{se}} \right\}_{se=1}^{n_{Kse}} ,$$

the integral of any product of shape functions can be written as :

$$\int_K \varphi_{iv}\,\varphi_{jv}\ \mathrm{d}\mathbf{x} = \boldsymbol{\phi}_{iv}^{\mathrm{T}}\ \mathbf{W}\ \boldsymbol{\phi}_{jv}\ , \tag{2.24}$$

where $\mathbf{W}$ is a diagonal $g_K \times g_K$ square matrix

$$\mathbf{W} = \mathrm{diag}\,(\ \ldots\ ,\ |J_{\widehat{T}_{se}}|\ w_{i,\,se}\ ,\ \ldots\ )\ ,$$

whose diagonal terms are the integration weights.
The complete set of shape functions can be represented as a $g_K \times nv$ rectangular matrix

$$\boldsymbol{\Phi} = [\ \ldots\ \boldsymbol{\phi}_{iv}\ \ldots\ ]\ ,$$

so that the mass matrix reads :

$$\mathbf{M} = \boldsymbol{\Phi}^{\mathrm{T}}\ \mathbf{W}\ \boldsymbol{\Phi}\ . \tag{2.25}$$

The mass matrix is always symmetric and positive-definite[6] thus it can be diagonalized,

$$\mathbf{M} = \mathbf{Q}\ \mathbf{D}\ \mathbf{Q}^{\mathrm{T}}\ , \tag{2.26}$$

where $\mathbf{Q}$ is the set of the eigenvectors and $\mathbf{D}$ is the diagonal matrix of the positive eigenvalues. In (2.26) $\mathbf{Q}^{\mathrm{T}}$ replaces $\mathbf{Q}^{-1}$ because the diagonalization of a symmetric matrix implies an orthonormal set of eigenvectors. Then, substituting expression (2.25) for $\mathbf{M}$ in (2.26), we write

$$(\mathbf{Q}\ \boldsymbol{\Lambda})^{\mathrm{T}}\ \boldsymbol{\Phi}^{\mathrm{T}}\ \mathbf{W}\ \boldsymbol{\Phi}\ (\mathbf{Q}\ \boldsymbol{\Lambda})\ =\ \mathbf{I}\,, \tag{2.27}$$

[6]The commutativity of the dot product insures the symmetry of $\mathbf{M}$, whereas its positivity can be verified by means of the definition. A square $nv \times nv$ matrix $\mathbf{M}$ is positive-definite if and only if $\mathbf{z}^{\mathrm{T}}\ \mathbf{M}\ \mathbf{z}\ >\ 0\quad \forall \mathbf{z} \in \mathbb{R}^{nv}_{/\{0\}}$, which is equivalent to requiring that $\int_K p^2\ \mathrm{d}\mathbf{x}\ >\ 0$ where $p = \sum\ \mathbf{z}_i\ \varphi_i$.

where $\mathbf{I}$ is the identity matrix and $\mathbf{\Lambda}$ equals $\mathbf{D}^{-\frac{1}{2}} = \mathrm{diag}\,(\ \ldots\ ,\ \mathbf{D}_{i,\,i}^{-\frac{1}{2}}\ ,\ \ldots\ )$.
The $g_K \times nv$ rectangular matrix

$$\mathbf{\Phi}^\star = \mathbf{\Phi}\ (\mathbf{Q}\ \mathbf{\Lambda})\ , \qquad \text{which verifies} \qquad \mathbf{I} = (\mathbf{\Phi}^\star)^{\mathrm{T}}\ \mathbf{W}\ \mathbf{\Phi}^\star\ ,$$

is an orthonormal set of shape functions

$$\mathbf{\Phi}^\star = [\ \ldots\ \boldsymbol{\phi}^\star_{iv}\ \ldots\ ]\ ,$$

and $(\mathbf{Q}\ \mathbf{\Lambda})$ are expansion coefficients of the orthonormal set $\mathbf{\Phi}^\star$ onto the initial set $\mathbf{\Phi}$, *i.e.*

$$\boldsymbol{\phi}^\star_{jv} = \mathbf{\Lambda}_{jv,\,jv} \sum_{iv=1}^{nv} \mathbf{Q}_{iv,\,jv}\ \boldsymbol{\phi}_{iv} \qquad \forall jv \in \mathbb{N}_{(1,\,nv)}\,. \tag{2.28}$$

The linear expansion (2.28) points out that the starting set of shape functions $\mathbf{\Phi}$ affects the orthonormalization procedure.

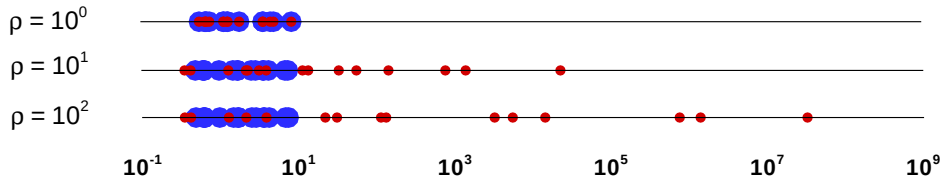
Let us consider any numerical process which takes a non-orthogonal set of basis functions $\mathbf{\Phi}$ and successively, at any step $iv = 1, \ldots, nv$, computes the shape function

$$\boldsymbol{\phi}^\star_{iv} = \sum_{jv=1}^{iv-1} c_{\,iv,\,jv}\ \boldsymbol{\phi}^\star_{jv}\ +\ c_{\,iv,\,iv}\ \boldsymbol{\phi}_{iv}\ , \tag{2.29}$$

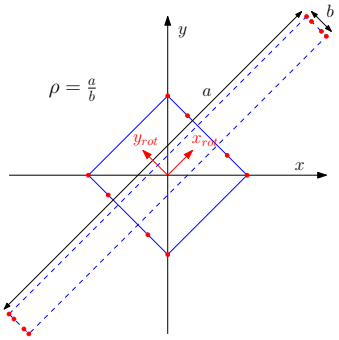
which is orthonormal to all the previous $\left\{\boldsymbol{\phi}^\star_{jv}\right\}_{jv=1}^{iv-1}$.
Eq. (2.11) relates the ratio of maximum to minimum eigenvalues to the condition number of a given mass matrix. Then, a starting shape set, which is associated to an ill-conditioned mass matrix, implies that :

$$\max_{1\le jv\le nv} \{\mathbf{\Lambda}_{jv,\,jv}\} \,/\, \min_{1\le jv\le nv} \{\mathbf{\Lambda}_{jv,\,jv}\} \ \gg\ 1\ ,$$

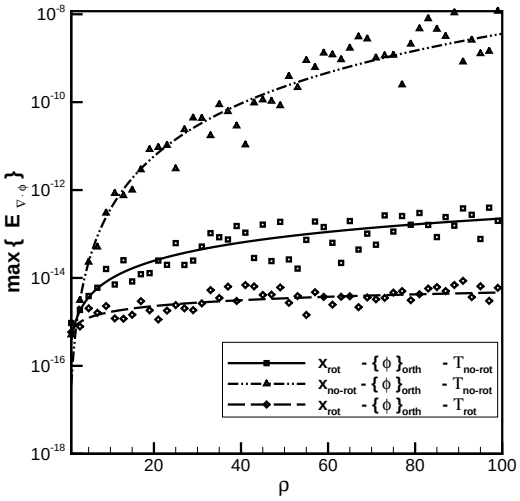
and the magnitude of coefficients appearing in (2.29) could fluctuate very much, causing cancellation errors in the numerical procedure.



(a) $\{\mathbf{\Lambda}_{j,\,j}\}_{j=1}^{15}$ Legend : *blue* = rotated/scaled, *red* = non-rot/non-scaled



(b) Tested quadrangular



(c) Divergence error

Figure 2.5: *Top* : $\{\mathbf{\Lambda}_{jv,\,jv}\}_{jv=1}^{nv}$ coefficients for the scaled/rotated and for the non-scaled/non-rotated starting bases. *Bottom* : distribution of the divergence error of orthonormal shape sets derived by means of MGS procedure from different starting bases. *Machine precision* : 15 decimal digits

Let us consider a rectangular element stretched along the first quadrant bisector (figure 2.5(b)). In figure 2.5(a) are compared the diagonal terms

of $\mathbf{\Lambda}$ for both the rotated/scaled and the non-rotated/non-scaled basis sets on that element. The mass matrix eigenvalues are not much affected by element aspect ratio if local reference frame axes are aligned with elemental principal inertial axes. Conversely, when this precaution is not taken, the mass matrix condition number and its eigenvalues degrade quickly when rising the element ratio ρ. This remark justifies the choice of basis (2.5) as starting set for the orthonormalization process, in facts it yields well-conditioned mass matrices even for straight elements with high aspect ratios.

Previous considerations suggested that the more coefficients $\{\mathbf{\Lambda}_{jv,\,jv}\}_{jv=1}^{nv}$ spread out the more the cancellation errors spoil the numerical orthonormalization procedure. Further numerical results corroborate this supposition and reveal an additional drawback related to defining shape functions and integration rules in the real space.

First we define a proper indicator of "shape function quality", that is the divergence error,

$$\begin{aligned} E_{\boldsymbol{\nabla}\cdot\boldsymbol{\phi}} &= \max_{1\le iv\le nv}\{e_{x,\,iv}+e_{y,\,iv}\} \qquad \text{with :} \\ e_{v,\,iv} &= \left|\int_K \partial_v\varphi_{iv}\;\mathrm{d}\mathbf{x}-\int_{\partial K}\varphi_{iv}\,n_v\;\mathrm{d}\sigma\right| \qquad \forall\, v\in\{x,\,y\}\;, \end{aligned} \tag{2.30}$$

which results from the Gauss-Green theorem. The divergence error is computed numerically by means of integration rules which integrate exactly any polynomial belonging to $\mathbb{P}_4(K)$ and $\mathbb{P}_4(\partial K)$ spaces, where K is the quadrangular in figure 2.5(b) and ∂K is its boundary. It compares boundary integrals of shape functions with volume integrals of their derivatives, *i.e.* it implicates the evaluation of two different functions on two different quadrature node sets. Moreover it represents the numerical conservativity of shape functions with respect of integration rules, which are exact in the considered approximation space.
Then, we compare the divergence errors of orthonormal shape sets which

are derived from different initial bases by means of the same process, see section 2.6.
Results are depicted in figure 2.5(c) and suggest the following considerations :

- When the starting basis (2.5) is defined in the non-rotated local reference frame (x, y), the divergence error on the final orthonormal set grows unaccetably large in stretched elements ($\mathbf{x_{no\ rot}}\ \mathbf{T_{no\ rot}}$). In facts round-off errors in the MGS procedure make shape functions, defined by their evaluations in integration nodes, being no longer polynomials. As a matter of fact, in this case the more ρ increases the more basis coefficients $\{\mathbf{\Lambda}_{jv,\,jv}\}_{jv=1}^{nv}$, red spots in figure 2.5(a), spread out.

- By using the rotated frame (x_{rot}, y_{rot}) to define the starting basis, it is produced an orthonormal set ($\mathbf{x_{rot}}\ \mathbf{T_{no\ rot}}$), which is affected by much lower divergence errors than in the previous case. In facts, local frame rotation makes $\{\mathbf{\Lambda}_{jv,\,jv}\}_{jv=1}^{nv}$ coefficients, blue spots in figure 2.5(a), being independent of element ratio ρ. However, even in this case, the divergence error slightly increases on more and more stretched elements and the starting basis shows the same error trend.

- The initial set is composed by monomials which are evaluated in each cubature node as products of its coordinates, so that the error can be only related to wrong location of integration nodes. The more the element stretches the more integration nodes gather in the direction normal to elongation, affecting the mapping of cubature nodes in the real space, see figure 2.5(b). In order to reduce this effect, we consider the mapping of the canonical quadrangle in the real one expressed in the rotated reference frame. This means that we are relieving the mapping of element rotation, thus increasing the accuracy of cubature node locations in the real space. As expected, this operation results in a further reduction of the divergence error of final orthonormal shape functions, referred as ($\mathbf{x_{rot}}\ \mathbf{T_{rot}}$).

In order to obtain "good orthonormal sets", it is necessary to construct well-conditioned starting bases on macro-elements, thus avoiding amplification of initial "conservativity errors", and to locate accurately cubature nodes on sub-elements by means of suitable mappings. By the way, high levels of divergence error ($1e-5 \sim 1e-6$) do not seem to affect the spatial discretization accuracy in smooth solutions.

It is important to conclude that all these observations are relevant only if shape functions are defined in the real space.

2.6 The orthogonalization process

The procedure to produce a set of orthonormal shape functions on a generic polyhedral element K relies on the modified Gram-Schmidt (MGS) orthogonalization algorithm. The sole requirement of this procedure is the capability to compute the integral of polynomial functions on arbitrarily shaped elements. Thus, the MGS procedure with re-orthogonalization[7] can be simply setup as shown in the following pseudo-code :

MGS ALGORITHM WITH RE-ORTHOGONALIZATION

1 **for** $iv \leftarrow 1$ **to** nv
2 **do for** $n \leftarrow 1$ **to** 2
3 **do for** $jv \leftarrow 1$ **to** $iv - 1$
4 **do** $r^{(n)}_{iv,\,jv} \leftarrow \langle \boldsymbol{\phi}_{iv},\, \boldsymbol{\phi}^{\star}_{jv} \rangle_K$
5 $\boldsymbol{\phi}_{iv} \leftarrow \boldsymbol{\phi}_{iv} - r^{(n)}_{iv,\,jv} \boldsymbol{\phi}^{\star}_{jv}$
6 $r^{(n)}_{iv,\,iv} \leftarrow \sqrt{\langle \boldsymbol{\phi}_{iv},\, \boldsymbol{\phi}_{iv} \rangle_K}$
7 $\boldsymbol{\phi}_{iv} \leftarrow \boldsymbol{\phi}_{iv} / r^{(n)}_{iv,\,iv}$
8 $\boldsymbol{\phi}^{\star}_{iv} \leftarrow \boldsymbol{\phi}_{iv}$

Line 2 indicates that orthogonalization is applied twice.

[7] Refer to [40], for more information concerning the accuracy of the orthogonalization process.

The set of orthonormal shape functions we wish to construct and the starting set of basis functions defined on K, are denoted by $\{\,\boldsymbol{\phi}^{\star}_{iv}\,\}_{iv=1}^{nv}$ and $\{\,\boldsymbol{\phi}_{iv}\,\}_{iv=1}^{nv}$, respectively. Moreover, $\left\langle \boldsymbol{\phi}^{\Delta},\, \boldsymbol{\phi}^{\nabla} \right\rangle_K$ indicates the inner product $\left(\boldsymbol{\phi}^{\Delta}\right)^{\mathrm{T}}\, \mathbf{W}\, \boldsymbol{\phi}^{\nabla}$, as defined in (2.24), where $\boldsymbol{\phi}^{\Delta}$ and $\boldsymbol{\phi}^{\nabla}$ are either orthonormal or non-orthonormal shape functions. Finally we recall that basis (2.5) is the actual choice of the starting set, as thoroughly justified in section 2.5.

As pointed out in (2.29), the above MGS algorithm amounts to constructing the set of orthonormal shape functions $\{\,\boldsymbol{\phi}^{\star}_{iv}\,\}_{iv=1}^{nv}$ according to the following system of equations :

$$\boldsymbol{\phi}^{\star}_{iv} = \sum_{jv=1}^{iv-1} c_{iv,\,jv}\; \boldsymbol{\phi}^{\star}_{jv} \;+\; c_{iv,\,iv}\; \boldsymbol{\phi}_{iv} \qquad \forall\, iv \in \mathbb{N}_{(1,\,nv)}\,, \tag{2.31}$$

where coefficients $c_{iv,\,jv}$ are determined by enforcing each new $\boldsymbol{\phi}^{\star}_{iv}$ to be orthogonal to the $iv-1$ already orthonormalized basis functions, whereas the coefficient $c_{iv,\,iv}$ is the L_2 normalizing factor of the newly created $\boldsymbol{\phi}^{\star}_{iv}$. So that, $\forall\, iv \in \mathbb{N}_{(1,\,nv)}$ and $\forall\, jv \in \mathbb{N}_{(1,\,iv-1)}$, we write :

$$\begin{aligned} c_{iv,\,iv} &= \left\{ \left\langle \boldsymbol{\phi}_{iv},\, \boldsymbol{\phi}_{iv} \right\rangle_K - \sum_{jv=1}^{iv-1} \left\langle \boldsymbol{\phi}_{iv},\, \boldsymbol{\phi}^{\star}_{jv} \right\rangle_K^2 \right\}^{-\frac{1}{2}}\,, \\ c_{iv,\,jv} &= -\, c_{iv,\,iv}\; \left\langle \boldsymbol{\phi}_{iv},\, \boldsymbol{\phi}^{\star}_{jv} \right\rangle_K\,. \end{aligned}$$

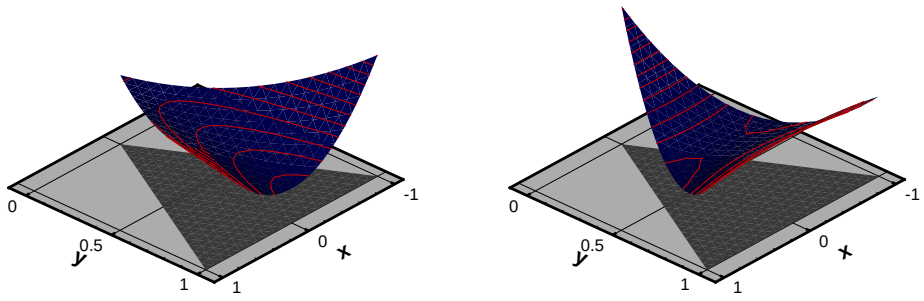
Then, from eq. (2.31), it is clear that the orthonormal set $\{\,\boldsymbol{\phi}^{\star}_{iv}\,\}_{iv=1}^{nv}$ is also hierarchical. In facts, increasing the degree of polynomial approximation entails adding to the existing set of basis functions as many $\boldsymbol{\phi}^{\star}_{iv}$ of the form of eq. (2.31) as the number of new $\boldsymbol{\phi}_{iv}$ up to the required degree, without changing the already existing orthonormal basis functions.

Finally we remark that the MGS algorithm outlined above is also used to compute the values of basis functions (and of their spatial derivatives, if necessary) at any location other than those needed to compute the integrals of lines 4 and 6. In such cases the symbols $\boldsymbol{\phi}$ and $\boldsymbol{\phi}^{\star}$ at lines 5, 7 and 8

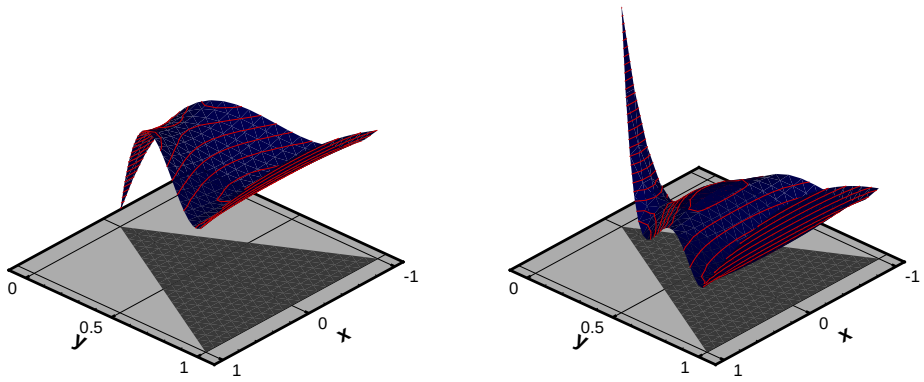
denote values of starting and orthonormalized basis functions (or of their derivatives) at the desired location.

2.7 Conclusions and representation

In sections 2.2 and 2.6, both a suitable starting set of basis functions and a numerical procedure to derive a hierarchic orthonormal set of shape functions were presented. The choice of the starting set was justified in section 2.5, pointing out that it has an important role during the orthonormalization process. The results suggest that, in order to further improve the range of applicability of those shape functions, it necessary to ameliorate the starting set of basis functions. Anyway, the method derived so far works very well for many element shapes and for a wide range of polynomial degrees. Furthermore, the use of polygonal grid discretizations will concern coarse levels in h-multigrid. Those grids are constructed by minimizing the aspect ratio of coarse-elements, which are agglomerations of the fine-elements. Then, the overall quality of the discretization will be "good" even for high polynomial degrees, see section 2.2.
Finally, some orthonormal shape functions are represented on "classic" and "special" element shapes, in order to stress on the geometrical flexibility of the presented discretization.

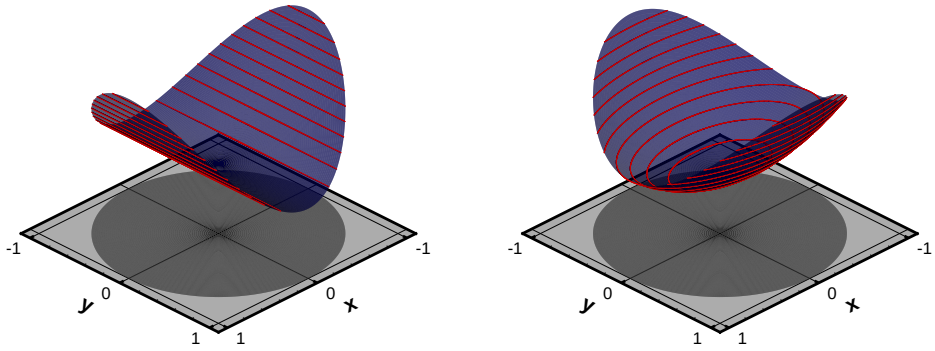


(a) $\phi_3^\star$ (b) $\phi_5^\star$



(c) $\phi_9^\star$ (d) $\phi_{14}^\star$

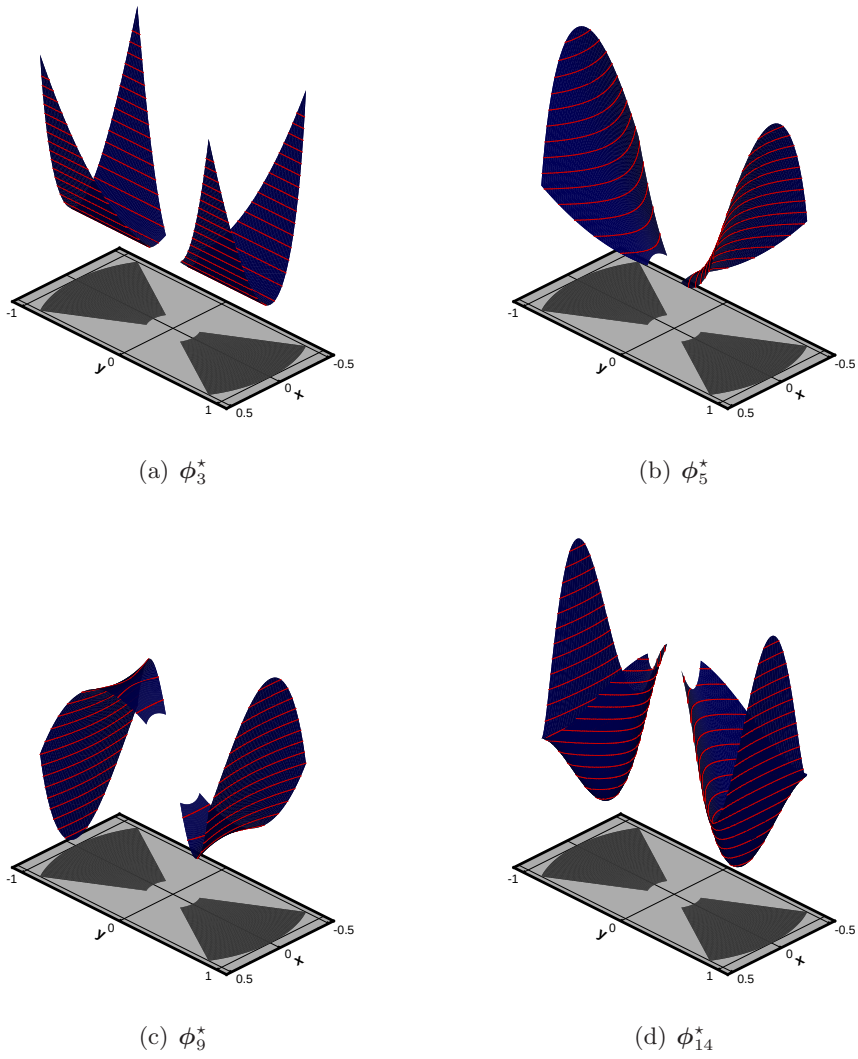
Figure 2.6: Four orthogonal shape functions on a "classic" element. Shape maxima are located in regions of minimum surface along inertial principal axes



(a) $\phi_3^\star$ (b) $\phi_5^\star$

(c) $\phi_9^\star$ (d) $\phi_{14}^\star$

Figure 2.7: Four orthogonal shape functions on a circle. The circle has the optimum aspect ratio, see section 2.2. Furthermore is a polyhedral with numberless faces



(a) $\boldsymbol{\phi}_3^\star$ (b) $\boldsymbol{\phi}_5^\star$

(c) $\boldsymbol{\phi}_9^\star$ (d) $\boldsymbol{\phi}_{14}^\star$

Figure 2.8: Four orthogonal shape functions on a disconnected domain. The shape sets can be defined on any domain

2.8 Governing equations

The fluid flow models which are considered include inviscid and viscous compressible flows governed by Euler and Navier-Stokes equations.
The governing equations of mass, momentum and energy conservation are fully coupled in the compressible case and, written in conservation form, read :

$$\frac{\partial \mathbf{u}}{\partial t} \; + \; \boldsymbol{\nabla} \cdot \mathbf{F}_i(\mathbf{u}) \; - \; \boldsymbol{\nabla} \cdot \mathbf{F}_v(\mathbf{u}) = \mathbf{0} \qquad \text{in} \quad [\,0,\, t\,] \times \Omega \,, \tag{2.32}$$

where $\Omega \subset \mathbb{R}^d,\ d \in \{2, 3\}$ is a bounded and connected Lipschitz domain.

The conservative variables $\mathbf{u}$ and the inviscid flux function $\mathbf{F}_i(\mathbf{u})$, for $d = 2$, equal :

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho \mathbf{v}_x \\ \rho \mathbf{v}_y \\ \rho e_0 \end{bmatrix}, \qquad \mathbf{F}_i(\mathbf{u}) \;=\; \left[\begin{array}{c|c} \rho \mathbf{v}_x & \rho \mathbf{v}_y \\ \rho \mathbf{v}_x^2 + p & \rho \mathbf{v}_x \mathbf{v}_y \\ \rho \mathbf{v}_x \mathbf{v}_y & \rho \mathbf{v}_y^2 + p \\ \rho h_0 \mathbf{v}_x & \rho h_0 \mathbf{v}_y \end{array}\right] \tag{2.33}$$

where ρ is the density, p is the pressure, $\mathbf{v}_a$ is velocity component parallel to a-axis and e_0 and h_0 are the stagnation energy and enthalpy per unit mass, respectively. That is :

$$\begin{aligned} e_0 \;&=\; e + e_k \qquad\quad h_0 = h + e_k \qquad\quad \text{with :} \\ e_k \;&=\; \frac{\mathbf{v}_x^2 + \mathbf{v}_y^2}{2}\,, \qquad e \;= \frac{c^2}{\gamma(\gamma-1)}\,, \qquad h \;= \frac{c^2}{\gamma - 1}\,, \end{aligned} \tag{2.34}$$

where c is the local speed of sound ($c^2 = \gamma R T$) and γ is the ratio of specific heats ($\gamma = c_p / c_v$). It is assumed that the fluid obeys the perfect gas state equation, *i.e.* $p = \rho\, R\, T$ or $p = (\gamma - 1)\, \rho e$.

Finally, the viscous flux function $\mathbf{F}_v(\mathbf{u})$ equals :

$$\mathbf{F}_v(\mathbf{u}) \;=\; \left\{ \begin{array}{c} \mathbf{0} \\ \boldsymbol{\tau}^{\mathbf{v}} \\ \mathbf{v} \cdot \boldsymbol{\tau}^{\mathbf{v}} - \mathbf{q} \end{array} \right\} \tag{2.35}$$

where $\mathbf{v}$ is the velocity vector, $\boldsymbol{\tau}^{\mathbf{v}}$ is a tensor

$$\boldsymbol{\tau}^{\mathbf{v}} = \mu\,(\boldsymbol{\nabla}\mathbf{v} + \boldsymbol{\nabla}\mathbf{v}^T) - \frac{2}{3}\mu\,(\boldsymbol{\nabla}\cdot\mathbf{v})\,\mathbf{I} \tag{2.36}$$

and $\mathbf{q}$ is the heat flux vector, which is related to temperature field by the *Fourier Law* :

$$\mathbf{q} = -k\boldsymbol{\nabla}T, \qquad \text{with} \qquad k = \frac{c_p\mu}{Pr} \tag{2.37}$$

where k is the thermal conductivity, μ is the dynamic viscosity and Pr is the Prandtl number.

For clarity $\mathbf{F}_v(\mathbf{u})$ is expressed in the same form of $\mathbf{F}_i(\mathbf{u})$ in (2.33).

$$\mathbf{F}_v(\mathbf{u}) \;=\; \left[\begin{array}{c|c} 0 & 0 \\ \mu\,\xi_{\mathbf{v}}^{(1)} & \mu\,\xi_{\mathbf{v}}^{(2)} \\ \mu\,\xi_{\mathbf{v}}^{(2)} & \mu\,\xi_{\mathbf{v}}^{(3)} \\ \mu\,\xi_{\mathbf{v}}^{(1)}\mathbf{v}_x + \mu\,\xi_{\mathbf{v}}^{(2)}\mathbf{v}_y + k\,\partial_x T & \mu\,\xi_{\mathbf{v}}^{(2)}\mathbf{v}_x + \mu\,\xi_{\mathbf{v}}^{(3)}\mathbf{v}_y + k\,\partial_y T \end{array}\right]$$

$$\begin{aligned} \xi_{\mathbf{v}}^{(1)} &= 2\partial_x\mathbf{v}_x - \frac{2}{3}\nabla\cdot\mathbf{v} \\ \xi_{\mathbf{v}}^{(2)} &= \partial_y\mathbf{v}_x + \partial_x\mathbf{v}_y \\ \xi_{\mathbf{v}}^{(3)} &= 2\partial_y\mathbf{v}_y - \frac{2}{3}\nabla\cdot\mathbf{v} \end{aligned} \tag{2.38}$$

where ∂_x and ∂_y refer to partial derivatives with respect to variables x and y, *i.e.* $\partial_x = \partial/\partial x$, $\partial_y = \partial/\partial y$.

2.9 Equation discretization

The discretization of equation (2.32) will be derived treating inviscid and viscous terms separately.

2.9.1 Euler equations

The conservation form of Euler equations is obtained by neglecting the $\boldsymbol{\nabla} \cdot \mathbf{F}_v(\mathbf{u})$ in eq. (2.32). Then, by multiplying by a "test function" $\mathbf{v}$[8], integrating over the domain Ω, and performing an integration by parts we obtain the weak statement of the problem

$$\int_\Omega \mathbf{v} \cdot \frac{\partial \mathbf{u}}{\partial t} \, \mathrm{d}\mathbf{x} \; + \; \int_{\partial\Omega} \mathbf{v} \cdot (\mathbf{F}_i(\mathbf{u}) \cdot \mathbf{n}) \; \mathrm{d}\sigma \; - \; \int_\Omega \boldsymbol{\nabla}\mathbf{v} \; : \; \mathbf{F}_i(\mathbf{u}) \, \mathrm{d}\mathbf{x} \; = \; 0$$

$$\forall \mathbf{v} \in \left[H^1(\Omega)\right]^N \tag{2.39}$$

where $\partial\Omega$ denotes the boundary of Ω and N is the number of equations in eq. (2.32). A discrete analogue of equation (2.39) is obtained by subdividing the domain Ω_h, approximation of Ω, into a set of non-overlapping polyhedral elements $K \in \mathcal{T}_h$[9], and considering functions $\mathbf{u}_h$ and $\mathbf{v}_h$, defined within each element, given by combination of nv shape functions φ_{iv}[10],

$$\mathbf{u}_h(\mathbf{x}, t) \; = \; \sum_{iv=1}^{nv} \mathbf{U}_{iv}(t) \; \varphi_{iv}(\mathbf{x}) \, , \quad \mathbf{v}_h(\mathbf{x}) \; = \; \sum_{iv=1}^{nv} \mathbf{V}_{iv} \; \varphi_{iv}(\mathbf{x}) \quad \forall \mathbf{x} \in K \tag{2.40}$$

The expansion coefficients $\mathbf{U}_{iv}(t)$ and $\mathbf{V}_{iv}$ denote the degrees of freedom of the numerical solution and of the test functions, respectively. By splitting integrals over Ω, appearing in eq. (2.39), into sums of integrals over elements K, the semi-discrete equations read :

$$\sum_{K \in \mathcal{T}_h} \; \int_K \mathbf{v}_h \cdot \frac{\partial \mathbf{u}_h}{\partial t} \, \mathrm{d}\mathbf{x} \; + \; \sum_{K \in \mathcal{T}_h} \; \int_{\partial K} \mathbf{v}_h \; \cdot (\mathbf{F}_i(\mathbf{u}_h|_K) \cdot \mathbf{n}) \; \mathrm{d}\sigma \; -$$

$$\sum_{K \in \mathcal{T}_h} \; \int_K \boldsymbol{\nabla}\mathbf{v}_h \; : \; \mathbf{F}_i(\mathbf{u}_h) \, \mathrm{d}\mathbf{x} \; = \; 0 \quad \forall \mathbf{v}_h \in \mathbf{V}_h \tag{2.41}$$

[8] Note that up to now $\mathbf{v}$ will refer to a "test function", not to the velocity vector

[9] $\mathcal{T}_h = \{K_{ie}\}_{ie=1}^{ne}$, triangulation of Ω_h, was introduced at the beginning of this section. Here, in order to ease the notation, let us drop ie.

[10] The ultimate shape functions are defined in (2.5).

where $\mathbf{V}_h$ is the space of test functions, defined in eq. (2.2), and ∂K denotes the boundary of element K.

Before going further it is necessary to introduce some notation. We denote with $\mathcal{E}_h^0$ the set of internal element faces, with $\mathcal{E}_h^\partial$ the set of boundary element faces and with $\mathcal{E}_h$ the union of the previous sets ($\mathcal{E}_h^0 \cup \mathcal{E}_h^\partial$). Then we set :

$$\Gamma_h^0 \stackrel{\text{def}}{=} \bigcup_{e \in \mathcal{E}_h^0} e, \qquad \Gamma_h^\partial \stackrel{\text{def}}{=} \bigcup_{e \in \mathcal{E}_h^\partial} e, \qquad \Gamma_h \stackrel{\text{def}}{=} \Gamma_h^0 \cup \Gamma_h^\partial \tag{2.42}$$

Moreover, having no global continuity requirement for V_h on Ω_h, functions $\mathbf{u}_h$ and $\mathbf{v}_h$ are double-valued on Γ_h^0 and single valued on Γ_h^∂. Therefore, it is convenient to introduce some trace operators which simplify the weak-form of the equations. For all vector quantities $\mathbf{q}$ and scalar quantities ζ such that a (possibly two-valued) trace is available, we define the *average* $\{\cdot\}$ and the *jump* $[\![\cdot]\!]$ as follows. Let e be an interior edge shared by elements K^+ and K^-. Define the unit normal vectors $\mathbf{n}^+$ and $\mathbf{n}^-$ on e pointing exterior to K^+ and K^-, respectively. Being $\mathbf{q}^i \stackrel{\text{def}}{=} \mathbf{q}|_{\partial K^i}$ and $\zeta^i \stackrel{\text{def}}{=} \zeta|_{\partial K^i}$ we set :

$$\begin{aligned}
&\{\mathbf{q}\} = \tfrac{1}{2}(\mathbf{q}^+ + \mathbf{q}^-)\,, \quad &&[\![\mathbf{q}]\!] = \mathbf{q}^+ \cdot \mathbf{n}^+ + \mathbf{q}^- \cdot \mathbf{n}^- \quad &&\text{on } e \in \mathcal{E}_h^0\,,\\
&\{\zeta\} = \tfrac{1}{2}(\zeta^+ + \zeta^-)\,, \quad &&[\![\zeta]\!] = \zeta^+ \mathbf{n}^+ + \zeta^- \mathbf{n}^- \quad &&\text{on } e \in \mathcal{E}_h^0\,,\\
&\{\zeta\} = \zeta\,, &&[\![\mathbf{q}]\!] = \mathbf{q} \cdot \mathbf{n} &&\text{on } e \in \mathcal{E}_h^\partial\,.
\end{aligned} \tag{2.43}$$

Due to the discontinuous approximation of the solution, flux terms in (2.41) are not uniquely defined at element interfaces. It is at this stage that the technique traditionally used in finite volume schemes is borrowed by the discontinuous finite element method. Thus, we substitute the flux $\mathbf{F}_i(\mathbf{u}_h|_K)$ with a suitably defined numerical flux $\hat{\mathbf{F}}_i(\mathbf{u}_h^+, \mathbf{u}_h^-)$. Summing equation (2.41) over the elements we obtain the DG formulation of the problem

(2.39) which then requires to find $\mathbf{u}_h \in \mathbf{V}_h$ such that :

$$\int_{\Omega_h} v_h \frac{\partial \mathbf{u}_h}{\partial t} \, \mathrm{d}\mathbf{x} \; + \; \int_{\Gamma_h} \hat{\mathbf{F}}_i(\mathbf{u}_h^+, \, \mathbf{u}_h^-) \cdot [\![v_h]\!] \, \mathrm{d}\sigma \quad - \quad \int_{\Omega_h} \mathbf{F}_i(\mathbf{u}_h) \cdot \boldsymbol{\nabla} v_h \, \mathrm{d}\mathbf{x} \;\; = \;\; \mathbf{0} \quad \forall v_h \in V_h \, . \tag{2.44}$$

Notice that eq. (2.44) is a system of N equations[11], v_h is a scalar function and $[\![v_h]\!]$ is a vector[12], refer to eq. (2.43).
The states $\mathbf{u}_h^+$ and $\mathbf{u}_h^-$ are calculated by means of the expansion appearing in eq. (2.40), which is evaluated at element interfaces. In order to guarantee the formal accuracy of the scheme, numerical fluxes have to be *consistent* and *conservative*. The *consistency* of the numerical flux $\hat{\mathbf{F}}_i$ reads :

$$\hat{\mathbf{F}}_i(\mathbf{z}, \, \mathbf{z}) = \mathbf{F}_i(\mathbf{z}) \tag{2.45}$$

whenever $\mathbf{z}$ is a smooth function satisfying the Dirichlet boundary conditions. Furthermore, we say that the numerical flux $\hat{\mathbf{F}}_i$ is *conservative* if it is single-valued on Γ_h.

There are several flux functions satisfying the above criteria such as the "exact" Riemann flux function or the approximate Lax-Friedrichs, Roe, Enquist-Osher, Harden-Lax-van Leer (HLLE) flux functions. In this work all computations were performed with the "exact" Riemann flux function. The key idea is to compute $\hat{\mathbf{F}}_i(\mathbf{u}_h^+, \, \mathbf{u}_h^-)$ as $\mathbf{F}_i(\mathbf{u}^*)$, where $\mathbf{u}^*$ denotes the solution of a Riemann problem across the discontinuity[13].

[11] Equation (2.41) is equivalent to the system of equations which is obtained by substituting the vector $\mathbf{v}_h \in \mathbf{V}_h$ with the scalar $v_h \in V_h$. Indeed, notice that the expression $\forall \mathbf{v}_h \in \mathbf{V}_h$ is equivalent to the expression $\forall (\mathbf{v}_h)_i \in V_h, \; \forall i \in \mathbb{N}_{(1, \, N)}$. The substitution could be avoided by using a proper *jump* operator for $\mathbf{v}_h$, as presented in [2] and in [3]. However, in the author opinion, equation (2.44) is much more clear in this form.

[12] In (2.43) was not defined the *scalar jump* operator $[\![\cdot]\!]$ on Γ_h^∂. However, for consistency with (2.41), $[\![v_h]\!]$ equals $v_h \mathbf{n}$ on Γ_h^∂.

[13] First velocity vectors $\mathbf{v}^+$ and $\mathbf{v}^-$ are projected onto the face unit normal vector $\mathbf{n}^-$, getting u_n^+ and u_n^-, respectively. Then the obtained one-dimensional Riemann problem

Despite the upwind treatment of fluxes at the element interfaces, the numerical solutions computed with the method described so far (excluding the special case of a single constant shape function φ_0 for each element) are oscillatory in the vicinity of flow discontinuities. As shown by Godunov theorem [10], this is a typical behaviour of any high-order accurate numerical scheme in the vicinity of a discontinuous solution. In order to compute physically relevant solutions, it is therefore necessary to introduce into the discontinuous finite element method some non linear dissipative mechanism which does not destroy the order of accuracy of the scheme, such as local projection (or slope limiter) limiting strategies proposed by Cockburn in [7].
An alternative stabilization approach is the use of a shock-capturing term similar to that commonly considered in the SUPG finite element method, see [6], [4]. This approach adds an artificial viscosity term of the form

$$\sum_{K\in\mathcal{T}_h}\int_K \epsilon|_K\,\boldsymbol{\nabla}\mathbf{u}\cdot\boldsymbol{\nabla}v_h\;\mathrm{d}\mathbf{x}\,, \tag{2.46}$$

with

$$\epsilon|_K \;=\; C_\epsilon\left(\sum_i\left(|K|\int_K\frac{\mathbf{s}_i}{\mathbf{u}_{h,i}}\;\mathrm{d}\mathbf{x}\right)^2\right)^{-\frac{1}{2}},$$

where C_ϵ is a positive parameter, i runs over all components of $\mathbf{u}_h$ and $\mathbf{s}$ verifies the weak form

$$\int_K v_h\;\mathbf{s}\;\mathrm{d}\mathbf{x}\;=\;\int_{\partial K} v_h\left(\mathbf{F}_i(\mathbf{u}_h^-)-\mathbf{F}_i(\mathbf{u}^*)\right)\cdot\mathbf{n}\;\mathrm{d}\sigma\quad\forall v_h|_K\in\mathbb{P}_k(K)\,, \tag{2.47}$$

being $\mathbf{F}_i$ the flux function defined in (2.33), $\mathbf{u}_h^-$ the trace of $\mathbf{u}_h|_K$ on ∂K, $\mathbf{u}^*$ the solution of the Riemann problem across the discontinuity on ∂K and $\mathbb{P}_k(K)$ the space of polynomials of global degree at most k on the element K. Notice that, due to the specific choice of $\mathbf{s}$ in eq. (2.47), the artificial

is solved by means of the exact iterative Riemann solver of Gottlieb and Groth (1978). Finally the face tangential velocity component of the state $\mathbf{u}^*$ is taken to be equal to the upwind value.

viscosity coefficient C_ϵ vanishes when a stationary shock wave is located on the element boundary. As matter of fact, the Rankine-Hugoniot jump condition states that

$$\mathbf{F}_i(\mathbf{u}) - \mathbf{F}_i(\mathbf{u}^*) = c\,(\mathbf{u} - \mathbf{u}^*)\,,$$

thus, $\mathbf{F}_i(\mathbf{u}) - \mathbf{F}_i(\mathbf{u}^*)$ vanishes when the wave speed c vanishes.

2.9.2 Navier-Stokes equations

Many techniques are available for the DG space discretizations of diffusive terms and a complete survey can be found in [1]. In this work it was used the method proposed in [6] and we describe it for the discretization of the viscous part of the Navier-Stokes equations in considerable detail.
In order to focus on the viscous term of the equations, we consider the model problem :

$$\begin{cases} \boldsymbol{\nabla} \cdot \mathbf{F}_v(\mathbf{u}, \boldsymbol{\nabla}\mathbf{u}) = \mathbf{0} & \text{in} \quad \Omega\,, \\ \\ \mathbf{u} = \mathbf{u}^b & \text{on} \quad \partial\Omega, \end{cases} \tag{2.48}$$

which may be reformulated as the first order system :

$$\begin{cases} \mathbf{z}_k = \boldsymbol{\nabla}\mathbf{u}_k, \quad \forall k \in \mathbb{N}_{(1,\,N)} \\ \\ \boldsymbol{\nabla} \cdot \mathbf{F}_v(\mathbf{u}, \mathbf{z}_1, \ldots, \mathbf{z}_N) = \mathbf{0} \qquad \text{in} \quad \Omega, \end{cases} \tag{2.49}$$

with the Dirichlet boundary conditions

$$\mathbf{u}_k = \mathbf{u}_k^b \qquad \text{on} \quad \partial\Omega \qquad \forall k \in \mathbb{N}_{(1,\,N)}\,,$$

where $\mathbf{u}_k$ is the k^{th} component of the state vector $\mathbf{u}$.
The weak statement of first N equations of system (2.49) is derived by multiplying by a *test function* $\mathbf{g} \in \mathbb{R}^2$, integrating over the domain Ω and performing an integration by parts. That is :

$$\int_\Omega \mathbf{g} \cdot \mathbf{z}\; \mathrm{d}\mathbf{x} \;=\; -\; \int_\Omega u\, \boldsymbol{\nabla} \cdot \mathbf{g}\; \mathrm{d}\mathbf{x} \;+\; \int_{\partial\Omega} u^b\, \mathbf{g} \cdot \mathbf{n}\; \mathrm{d}\sigma \qquad \forall \mathbf{g} \in \mathbb{R}^2, \tag{2.50}$$

where notations u, $\mathbf{z}$ and u^b replace $\mathbf{u}_k$, $\mathbf{z}_k$ and $\mathbf{u}_k^b$ respectively.
Then, repeating the procedure used to derive eq. (2.41) from eq. (2.39), we obtain the discrete analogue of eq. (2.50).
That is, find $u_h \in V_h$ and $\mathbf{z}_h \in \mathbf{G}_h$ so that :

$$\int_{\Omega_h} \mathbf{g}_h \cdot \mathbf{z}_h \, \mathrm{d}\mathbf{x} \;=\; - \int_{\Omega_h} u_h \, \boldsymbol{\nabla} \cdot \mathbf{g}_h \, \mathrm{d}\mathbf{x} \;+\; \int_{\Gamma_h^0} \hat{u}_h \, [\![\mathbf{g}_h]\!] \, \mathrm{d}\sigma \;+ \\ + \int_{\Gamma_h^\partial} u^b \, \mathbf{g}_h \cdot \mathbf{n} \, \mathrm{d}\sigma \qquad \forall \mathbf{g}_h \in \mathbf{G}_h \,, \tag{2.51}$$

where

$$V_h \stackrel{\text{def}}{=} \left\{ v_h \in L^2(\Omega_h) : \quad v_h|_K \in \mathbb{P}_k(K) \; \forall K \in \mathcal{T}_h \right\} ,$$

$$\mathbf{G}_h \stackrel{\text{def}}{=} \left\{ \mathbf{g}_h \in [L^2(\Omega_h)]^2 : \quad \mathbf{g}_h|_K \in [\mathbb{P}_k(K)]^2 \; \forall K \in \mathcal{T}_h \right\} ,$$

and where the scalar numerical flux function $\hat{u}_h$ was introduced to make univocal the evaluation of contour integrals on Γ_h^0.

A very natural choice for the discretization of purely elliptic problems is the centered numerical flux $\hat{u}_h = \{u_h\}$, where the average operator $\{\,\cdot\,\}$ was defined in eq. (2.43). So that eq. (2.51) becomes :

$$\int_{\Omega_h} \mathbf{g}_h \cdot \mathbf{z}_h \, \mathrm{d}\mathbf{x} \;=\; - \int_{\Omega_h} u_h \, \boldsymbol{\nabla} \cdot \mathbf{g}_h \, \mathrm{d}\mathbf{x} \;+\; \int_{\Gamma_h^0} \{u_h\} \, [\![\mathbf{g}_h]\!] \, \mathrm{d}\sigma \;+ \\ + \int_{\Gamma_h^\partial} u^b \, \mathbf{g}_h \cdot \mathbf{n} \, \mathrm{d}\sigma \qquad \forall \mathbf{g}_h \in \mathbf{G}_h \,. \tag{2.52}$$

However, the property, which is stated and demonstrated below, further simplifies the form of eq. (2.52).

Property 1. *Being $x_h \in V_h$ a arbitrary discontinuous scalar function in Ω_h and $\mathbf{y}_h \in \mathbf{G}_h$ an arbitrary vector function in Ω_h, the following identity*

holds :

$$\int_{\Omega_h} (x_h \, \boldsymbol{\nabla} \cdot \mathbf{y}_h \, + \, \boldsymbol{\nabla} x_h \cdot \mathbf{y}_h) \, \mathrm{d}\mathbf{x} \;\; = \tag{2.53}$$
$$\int_{\Gamma_h^0} (\{x_h\} \, [\![\mathbf{y}_h]\!] \;\; + \;\; \{\mathbf{y}_h\} \cdot [\![x_h]\!] \,) \, \mathrm{d}\sigma \;\; + \;\; \int_{\Gamma_h^\partial} x_h \, \mathbf{y}_h \cdot \mathbf{n} \, \mathrm{d}\sigma$$

where Γ_h^0 *and* Γ_h^∂ *where defined in (2.42).*

Proof *Eq. (2.53) is a consequence of the divergence theorem and of the relation :*

$$(x_h \mathbf{y}_h \cdot \mathbf{n})^- \, + \, , (x_h \mathbf{y}_h \cdot \mathbf{n})^+ \, = \, \{x_h\} [\![\mathbf{y}_h]\!] \, + \, \{\mathbf{y}_h\} \cdot [\![x_h]\!] \, , \tag{2.54}$$

which can be verified by a simple algebraic calculation. By using the relation $\mathbf{n}^+ + \mathbf{n}^- = \mathbf{0}$ *the first term in eq. (2.54) becomes :*

$$x_h^+ \mathbf{y}_h^+ \cdot \mathbf{n}^+ \, + \, \frac{x_h^+ \mathbf{y}_h^- \cdot (\mathbf{n}^- + \mathbf{n}^+)}{2} \, + \, x_h^- \mathbf{y}_h^- \cdot \mathbf{n}^- \, + \, \frac{x_h^- \mathbf{y}_h^+ \cdot (\mathbf{n}^+ + \mathbf{n}^-)}{2} =$$
$$= \frac{x_h^+}{2} [\![\mathbf{y}_h]\!] \, + \, \{\mathbf{y}_h\} x_h^+ \mathbf{n}^+ \, + \, \frac{x_h^-}{2} [\![\mathbf{y}_h]\!] \, + \, \{\mathbf{y}_h\} x_h^- \mathbf{n}^- = \{x_h\} [\![\mathbf{y}_h]\!] \, + \, \{\mathbf{y}_h\} \cdot [\![x_h]\!]$$

The divergence theorem reads :

$$\int_{\Omega_h} (\boldsymbol{\nabla} \cdot \mathbf{y}_h \, x_h \, + \, \mathbf{y}_h \cdot \boldsymbol{\nabla} x_h) \, \mathrm{d}\mathbf{x} \;\; = \;\; \int_{\Omega_h} \boldsymbol{\nabla} \cdot (x_h \, \mathbf{y}_h) \, \mathrm{d}\mathbf{x} \;\; =$$
$$= \;\; \int_{\Gamma_h^0} \left((x_h \mathbf{y}_h \cdot \mathbf{n})^- \, + \, (x_h \mathbf{y}_h \cdot \mathbf{n})^+ \right) \, \mathrm{d}\sigma \, + \, \int_{\Gamma_h^\partial} x_h \, \mathbf{y}_h \cdot \mathbf{n} \, \mathrm{d}\sigma \, , \tag{2.55}$$

Then, substituting eq. (2.54) in the right side of eq. (2.55), we obtain eq. (2.53) □

By replacing the first integral on the right hand side of eq. (2.52) with eq. (2.53), where x_h and $\mathbf{y}_h$ stand for u_h and $\mathbf{g}_h$ respectively, we obtain a new form of eq. (2.51) which reads :

$$\int_{\Omega_h} \mathbf{g}_h \cdot (\mathbf{z}_h - \boldsymbol{\nabla} u_h) \;\; \mathrm{d}\mathbf{x} \;\; = \;\; - \;\; \int_{\Gamma_h} \{\mathbf{g}_h\} \cdot [\![u_h]\!]^\partial \, \mathrm{d}\sigma \, , \tag{2.56}$$

where $[\![\cdot]\!]^\partial$ is the extended definition of the jump operator in eq. (2.43), which automatically takes care of Dirichlet boundary conditions.

That is :

$$[\![x]\!]^{\partial} = \begin{cases} (x - x^b)\mathbf{n} & \text{on} \quad \Gamma_h^{\partial} \\ [\![x]\!] & \text{on} \quad \Gamma_h^{0} \end{cases} \tag{2.57}$$

Eq. (2.59) shows that the auxiliary variable $\mathbf{z}_h$ is equal to the internal gradient $\boldsymbol{\nabla} u_h$ plus an additional term accounting for the jumps in u_h, occurring at inner element interfaces, and for the jumps between u_h and u^b, occurring at those boundaries where Dirichlet conditions are prescribed. This interpretation suggests to introduce the global lifting operator

$$\int_{\Omega_h} \mathbf{g}_h \cdot \mathbf{R}_h \left([\![u_h]\!]^{\partial} \right) \, \mathrm{d}\mathbf{x} \;=\; -\int_{\Gamma_h} \{\mathbf{g}_h\} \cdot [\![u_h]\!]^{\partial} \, \mathrm{d}\sigma \,, \tag{2.58}$$

which is a function that spreads, on the whole domain Ω_h, the effect of interface and boundary jumps of the state u_h. By virtue of the lifting operator $\mathbf{R}_h$, $\mathbf{z}_h$ may be expressed in weak sense as :

$$\int_{\Omega_h} \mathbf{g}_h \cdot \mathbf{z}_h \, \mathrm{d}\mathbf{x} \;=\; \int_{\Omega_h} \mathbf{g}_h \cdot \left(\boldsymbol{\nabla} u_h \;+\; \mathbf{R}_h \left([\![u_h]\!]^{\partial} \right) \right) \, \mathrm{d}\mathbf{x} \quad \forall \mathbf{g}_h \in \mathbf{G}_h \,. \tag{2.59}$$

The above relation implies that the first N equations of system (2.49) are automatically verified in *weak-finite sense* by taking

$$\mathbf{z}_{h,k} \;=\; \boldsymbol{\nabla} \mathbf{u}_{h,k} \;+\; \mathbf{R}_h \left([\![\mathbf{u}_{h,k}]\!]^{\partial} \right) \qquad \forall k \in \mathbb{N}_{(1,\,N)} \,. \tag{2.60}$$

Now, elaborating the last equation in (2.49) as done for the Euler equations, we obtain the DG formulation of the viscous term of the Navier-Stokes equations, which requires to find $\mathbf{u}_h \in \mathbf{V}_h$ such that :

$$\begin{aligned} &\int_{\Gamma_h} \hat{\mathbf{F}}_v \left(\mathbf{u}_h^+, \mathbf{z}_{h,1}^+, \ldots, \mathbf{z}_{h,N}^+, \mathbf{u}_h^-, \mathbf{z}_{h,1}^-, \ldots, \mathbf{z}_{h,N}^- \right) \cdot [\![v_h]\!] \, \mathrm{d}\sigma \;- \\ &\qquad \int_{\Omega_h} \mathbf{F}_v \left(\mathbf{u}_h, \mathbf{z}_{h,1}, \ldots, \mathbf{z}_{h,N} \right) \cdot \boldsymbol{\nabla} v_h \, \mathrm{d}\mathbf{x} \;=\; \mathbf{0} \quad \forall v_h \in V_h \,, \end{aligned} \tag{2.61}$$

where $[\![v_h]\!]$ equals $v_h\,\mathbf{n}$ on Γ_h^∂ [14] and $\hat{\mathbf{F}}_v$ is the centered numerical flux,

$$\hat{\mathbf{F}}_v\left(\mathbf{u}_h^+,\,\mathbf{z}_{h,1}^+,\ldots\,\mathbf{z}_{h,N}^+,\mathbf{u}_h^-,\,\mathbf{z}_{h,1}^-,\ldots\,\mathbf{z}_{h,N}^-\right) \;=\; \left\{\,\mathbf{F}_v\left(\mathbf{u}_h,\,\mathbf{z}_{h,1},\ldots,\,\mathbf{z}_{h,N}\right)\right\}\,. \tag{2.62}$$

The first term in eq. (2.61) shows that the scheme derived so far[15] has a non-compact support. As a matter of fact, the global lifting $\mathbf{R}_h^+\left([\![\mathbf{u}_h]\!]^\partial\right)$ sums the jump contributions on the whole K^+ element boundary, ∂K^+, and $\mathbf{R}_h^-\left([\![\mathbf{u}_h]\!]^\partial\right)$ does the same on ∂K^-. Thus, the numerical vector flux related to face e

$$\hat{\mathbf{F}}_v\left(\mathbf{u}_h^+,\,\mathbf{z}_{h,1}^+,\ldots,\,\mathbf{z}_{h,N}^+,\,\mathbf{u}_h^-,\,\mathbf{z}_{h,1}^-,\ldots,\,\mathbf{z}_{h,N}^-\right)\Big|_e\,,$$

where $e \stackrel{\text{def}}{=} \partial K^- \cap \partial K^+$, depends on the jump of $\mathbf{u}_h$ on $\partial K^- \cup \partial K^+$. Furthermore in [1], the **BR1** scheme was showed to be unstable for purely elliptic problems.
Subsequently in [6], by modifying the definition of the lifting operator, it was derived a stable scheme with a compact support, commonly referred as **BR2** scheme. In order to define it, let us consider local lifting operators $\mathbf{r}_h^e$ for each interface or boundary face $e \in \Gamma_h$ defined as :

$$\int_{\Omega_h^e} \mathbf{g}_h \cdot \mathbf{r}_h^e\left([\![u_h]\!]^\partial\right)\,\mathrm{d}\mathbf{x} \;=\; -\int_e \{\mathbf{g}_h\}\cdot[\![u_h]\!]^\partial\,\mathrm{d}\sigma \quad \forall e\in\Gamma_h\,,\quad \forall \mathbf{g}_h \in \mathbf{G}_h\,, \tag{2.63}$$

where Ω_h^e is either the union of the elements which share the same face e, or one boundary element, whereas u_h is the k^{th} component of the discrete state vector $\mathbf{u}_h$. Thus, all local lifting operators are associated to one face in Γ_h and each of them, $\mathbf{r}_h^e$, is null out of its domain of definition, Ω_h^e. Futhermore, local lifting operators represent face contributions to the previously defined

[14] In eq. (2.43) was not defined the scalar jump operator $[\![\,\cdot\,]\!]$ on Γ_h^∂. However, we preferred to use this notation because it is consistent with that used in eq. (2.44) and because, in order to write $[\![v_h]\!]^\partial$, we should define the right state of v_h first.

[15] Commonly referred as **BR1** scheme.

global lifting operator, $\mathbf{R}_h$. In facts, by combining eq. (2.63) with eq. (2.58), we write

$$\int_{\Omega_h} \mathbf{g}_h \cdot \mathbf{R}_h \left([\![u_h]\!]^\partial \right) \, \mathrm{d}\mathbf{x} \;=\; -\int_{\Gamma_h} \{\mathbf{g}_h\} \cdot [\![u_h]\!]^\partial \, \mathrm{d}\sigma \;=$$
$$-\sum_{e \in \Gamma_h} \int_e \{\mathbf{g}_h\} \cdot [\![u_h]\!]^\partial \, \mathrm{d}\sigma \;=\; \sum_{e \in \Gamma_h} \int_{\Omega_h^e} \mathbf{g}_h \cdot \mathbf{r}_h^e \left([\![u_h]\!]^\partial \right) \, \mathrm{d}\sigma \,,$$

which is equivalent to the weak equality

$$\mathbf{R}_h \left([\![u_h]\!]^\partial \right) \;=\; \sum_{e \in \Gamma_h} \mathbf{r}_h^e \left([\![u_h]\!]^\partial \right) \,, \tag{2.64}$$

since $\mathbf{g}_h$ is an arbitrary vector function in $\mathbf{G}_h$.

The **BR2** scheme is obtained by replacing the numerical flux defined in eq. (2.62) with

$$\hat{\mathbf{F}}_v \left(\mathbf{u}_h^+, \ldots, \mathbf{u}_h^-, \ldots \right)\big|_e \;=\; \{ \, \mathbf{F}_v \left(\mathbf{u}_h|_e, \, \mathbf{z}_{h,1}|_e, \, \ldots, \, \mathbf{z}_{h,N}|_e \right) \} \,, \tag{2.65}$$

where

$$\mathbf{z}_{h,i}|_e \;=\; \boldsymbol{\nabla} \mathbf{u}_{h,i}|_e \;+\; \eta_e \, \mathbf{r}_h^e \left([\![\mathbf{u}_{h,i}]\!]^\partial \right) \,,$$

with η_e a positive stabilizing coefficient, which has to be chosen sufficiently large.[16] Finally, by summing together eq. (2.44) and eq. (2.61) we obtain the DG formulation of the Navier-Stokes equations.
Find $\mathbf{u}_h \in \mathbf{V}_h$ such that :

$$\int_{\Omega_h} v_h \, \frac{\partial \mathbf{u}_h}{\partial t} \, \mathrm{d}\mathbf{x} + \int_{\Gamma_h} \hat{\mathbf{F}}_i(\mathbf{u}_h^+, \, \mathbf{u}_h^-) \cdot [\![v_h]\!] \, \mathrm{d}\sigma +$$
$$- \int_{\Omega_h} \left[\mathbf{F}_i(\mathbf{u}_h) - \mathbf{F}_v \left(\mathbf{u}_h, \, \boldsymbol{\nabla} \mathbf{u}_{h,1} + \mathbf{R}_h \left([\![\mathbf{u}_{h,1}]\!]^\partial \right), \ldots \right) \right] \cdot \boldsymbol{\nabla} v_h \, \mathrm{d}\mathbf{x} +$$
$$- \sum_{e \in \Gamma_h} \int_{\Gamma_h} \left\{ \mathbf{F}_v \left(\mathbf{u}_h, \, \boldsymbol{\nabla} \mathbf{u}_{h,1}|_e + \eta_e \, \mathbf{r}_h^e \left([\![\mathbf{u}_{h,1}]\!]^\partial \right), \ldots \right) \right\} \cdot [\![v_h]\!] \, \mathrm{d}\sigma = \mathbf{0}$$
$$\forall v_h \, \in V_h \,, \tag{2.66}$$

where $[\![v_h]\!]$ equals $v_h \, \mathbf{n}$ on Γ_h^∂.

[16] Concerning the stabilizing coefficient η_e it is better to refer to [1].

2.9.3 Boundary conditions

The DG discretization is best suited for a weak enforcement of boundary conditions. This can be achieved by properly defining a boundary state, which, together with the internal state, allows to compute the numerical fluxes and the lifting operators on Γ_h^∂.

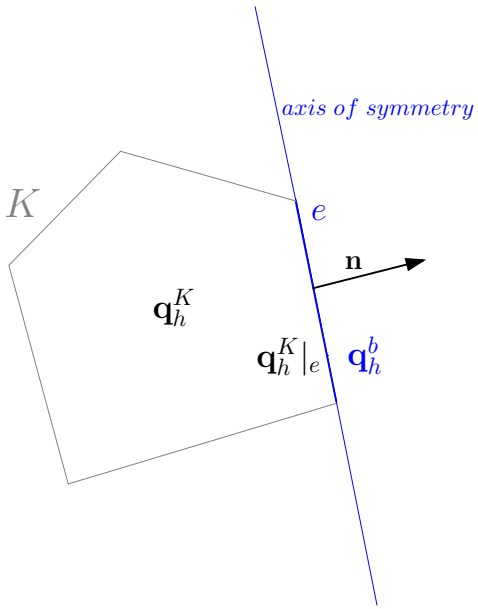


Figure 2.9: Imposition of symmetry boundary condition.

For example, the symmetry-type boundary conditions have been implemented by defining the boundary state in primitive variables $\mathbf{q}_h^b$ as a function of $\mathbf{q}_h^K|_e$, which is the trace on $e \in \Gamma_h^\partial$ of the discrete state $\mathbf{u}_h|_K$ in

primitive variables.[17] The relation reads :

$$\begin{bmatrix} p_h^b \\ t_h^b \\ u_h^b \\ v_h^b \end{bmatrix} = \begin{bmatrix} p_h^K \\ t_h^K \\ u_h^K - 2\,v_n\,n_x \\ v_h^K - 2\,v_n\,n_y \end{bmatrix} \quad \text{and} \quad \boldsymbol{\nabla}\mathbf{q}_{h,i}^b = \boldsymbol{\nabla}\mathbf{q}_{h,i}^b - 2\,(\boldsymbol{\nabla}\mathbf{q}_{h,i}^b \cdot \mathbf{n})\,\mathbf{n}\,,$$

where v_n is the velocity component normal to face e and i runs over all components of vector $\mathbf{q}_h^b$.
Notice that the external boundary state $\{\mathbf{q}_h^b, \boldsymbol{\nabla}\mathbf{q}_h^b\}$ is all we need to define viscous and inviscid numerical flux functions on Γ_h^∂.

This very natural way of imposing boundary conditions is another great advantage of DG methods, especially if they are coded in primitive variables.

2.10 Time discretization

All integrals appearing in the space discrete problem (2.66) are evaluated by means of the integration method presented in sections 2.3 and 2.4. By assembling together all the elemental contributions, the system of ordinary differential equations which governs the evolution in time of the discrete solution can be written as :

$$\mathbf{M}\,\frac{d\mathbf{U}}{dt} \;+\; \mathbf{R}(\mathbf{U}) \;=\; \mathbf{0}, \tag{2.67}$$

where $\mathbf{U}$ is the global vector of the degrees of freedom (see eq. (2.4) or eq. (2.40)), $\mathbf{M}$ denotes the block diagonal mass matrix, and $\mathbf{R}(\mathbf{U})$ is the residual vector. Due to the block diagonal structure of $\mathbf{M}$, the time integration of the above system of ODEs can be easily accomplished by means of an explicit method for initial value problems. That is, we could associate to any o^{th}-order accurate space discretization an o^{th}-order accurate Runge-Kutta time discretization.

[17] In order to distinguish the vector of primitive variables from the vector of conservative ones $\mathbf{u}$, we write $\mathbf{q} \stackrel{\text{def}}{=} (p, t, u, v)^T$

Explicit multistage Runge-Kutta scheme

1 $\mathbf{U}^0 \leftarrow \mathbf{U}^i$
2 **for** $j \leftarrow 1$ **to** N_{sts}
3 **do** $\mathbf{U}^j \leftarrow \mathbf{U}^0 \; - \; \alpha_j \, \Delta t \, \mathbf{M}^{-1} \mathbf{R}(\mathbf{U}^{j-1})$
4 $\mathbf{U}^{i+1} \leftarrow \mathbf{U}^{N_{sts}}$

where N_{sts} is the number of Runge-Kutta scheme stages.

N_{sts}	α_1	α_2	α_3	α_4	α_5
1	1	-	-	-	-
3	1/4	2/3	1	-	-
5	1/5	1/4	1/3	2/5	1

Table 2.1: Explicit Runge-Kutta scheme coefficients

This scheme results slow to converge to the stationary solution. As a matter of fact each element K marches with the maximum Δt_K which verifies the following heuristic condition for the CFL number :

$$\text{CFL} \; \leq \; \frac{1}{2\,k+1}\,, \tag{2.68}$$

being k the maximum degree of shape functions in element K.

A locally-implicit residual smoother is a reasonable compromise between :

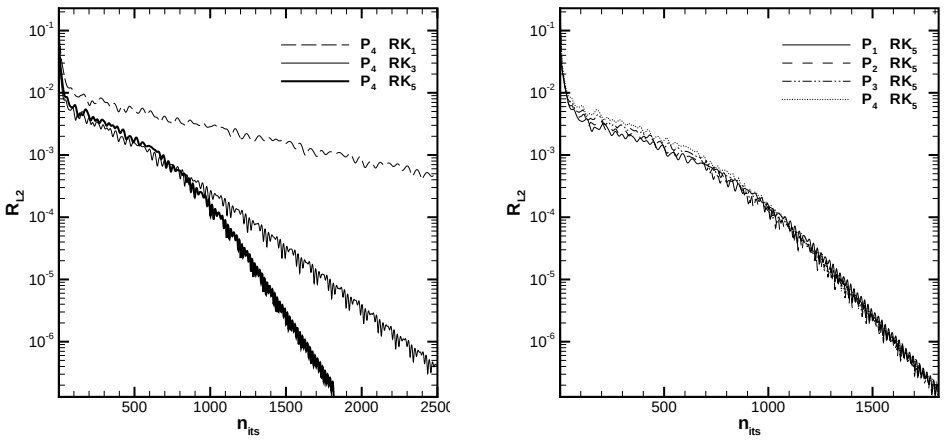
- fully explicit DG schemes which are inefficient due to the very restrictive CFL condition, eq. (2.68)
- and fully implicit solvers which are very expensive in terms of memory requirements

Thus, replacing $\mathbf{R}(\mathbf{U}^{j-1})$ in line 3 with $\mathbf{D}^0 \, \delta\mathbf{U}^j + \mathbf{R}(\mathbf{U}^{j-1})$, where $\mathbf{D}^0$ is the elemental jacobian "frozen" at the first Runge-Kutta stage, we obtain :

LOCALLY-IMPLICIT MULTISTAGE SCHEME

$\mathbf{U}^0 \leftarrow \mathbf{U}^i$
$\Delta\mathbf{U} \leftarrow \mathbf{0}$
for $j \leftarrow 1$ **to** N_{sts}
  do $\mathbf{A} \leftarrow \mathbf{M} + \alpha_k \, \Delta t \, \mathbf{D}^0$
    $\delta\mathbf{U}^j \leftarrow - \mathbf{A}^{-1}(\mathbf{M}\,\Delta\mathbf{U} + \alpha_k \, \Delta t \, \mathbf{R}(\mathbf{U}^{j-1}))$
    $\Delta\mathbf{U} \leftarrow \Delta\mathbf{U} + \delta\mathbf{U}^j$
    $\mathbf{U}^j \leftarrow \mathbf{U}^{j-1} + \delta\mathbf{U}^j$
$\mathbf{U}^{i+1} \leftarrow \mathbf{U}^{N_{sts}}$

The locally-implicit multistage scheme, derived so far, allows to rise the CFL number up to arbitrary values, provided that it is sufficiently small in the first iterations. In the present work, the CFL number was increased from 0.5 to 50 linearly with the residual reduction, in order to help alleviate transients during the solution process.



(a) N_{sts} comparison (b) k comparison

Figure 2.10: Locally-implicit multistage scheme performances

Moreover, updating the residual more than once during an iteration enhances the "communication" between neighbouring elements. As a matter of fact the number of iterations to converge reduces when increasing the number of stages in the scheme, see figure 2.10(a), and it is independent of the polynomial degree of the discretization, see figure 2.10(b).

Note that the α_k coefficients in the locally-implicit multistage scheme equal those used in the explicit Runge-Kutta scheme, which are tabulated in table 2.1. This is just a preliminary choice, because the locally-implicit scheme is stable for a wide range of coefficients α_k and their effect has to be investigated.

Chapter 3

Numerical results

In order to asses the accuracy of the discontinuous Galerkin approximation on polyhedral grids, uniform grid refinement was performed for three different problems. The transonic Ringleb flow and the subsonic flow over a gaussian bump were considered as representative cases for smooth solutions of $2D$ compressible Euler equations. The Helmholtz problem was studied to verify the accuracy of the viscous numerical flux. The results indicate that the DG discretization on polyhedral grids attains a full $O(h^{p+1})$ order of convergence for smooth solutions.
Moreover two problems were used to test the method in the viscous case : the NACA 0012 airfoil with $\mathrm{Re}_{\infty,1} = 5000$, 0^o incidence and $\mathrm{M}_\infty = 0,5$ and the NACA 0012 airfoil with $\mathrm{Re}_{\infty,1} = 73$, 10^o incidence and $\mathrm{M}_\infty = 0,8$.
Finally we studied the effects of reducing integration rules by means of the solution of compressible Euler equations around the NACA 0012 airfoil with 0^o incidence and $\mathrm{M}_\infty = 0,5$.

3.1 Hexagonal grid generation

The hexagon was chosen as representative element to asses the accuracy of the discontinuous Galerkin discretization on polyhedral grids. The con-

struction of regular hexagonal grids was supported by an uniform quad grid. The hexagonal element vertices and the additional points, which are needed to represent accurately curved boundaries, are calculated from the auxiliary quad grid. As shown in figure 3.1 the hexagonal element shape

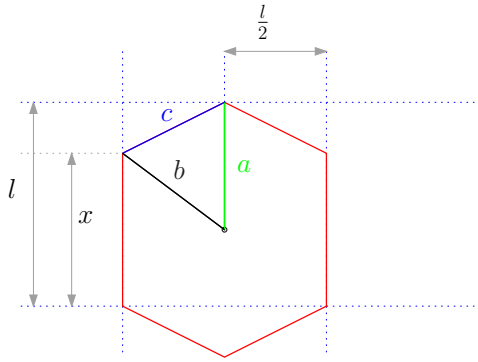


Figure 3.1: Hexagonal element from quad grid

is defined by choosing $x = \xi\, l$, where l is the length of any quadrilateral side of the auxiliary grid and $\xi \in (0, 1)$ is a scalar. Then, we required x to minimize the function

$$f(x) = |a(x) - x| + |b(x) - x| + |c(x) - x|\,,$$

and we found $x = 0.625\, l$.

The hexagonal grid derived so far is strongly related with its auxiliary quad grid. Notice that, considering uniform grids, the hexagon surface equals the respective quadrilateral one (figure 3.1). This observation suggests that the refinement of a hexagonal grid can be achieved by refining its auxiliary grid and then generating the hexagonal grid from it. As a matter of fact the hexagonal grid size scales with its auxiliary mesh size. Notice that the coarse hexagon is composed by three fine nested hexagons and three slices of fine non-nested hexagons, which verify $A = B = C = a + b + c$, supposing uniform the auxiliary grid (figure 3.2). Finally, in order to mesh the

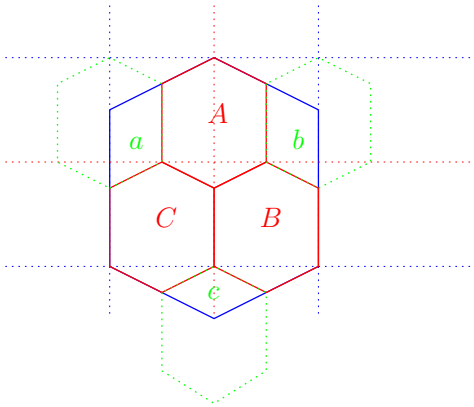


Figure 3.2: One-step refinement by means of *quasi-nested* hexagonal grids. Auxiliary quad grids: coarse (dotted blue) and fine (dotted red). Hexagonal grids: coarse (blue), fine (green and red)

portion of Ω_h contiguous to its boundary $\partial\Omega_h$ some collapsed hexagonal elements were considered (figure 3.3). Notice that, after refining, the col-

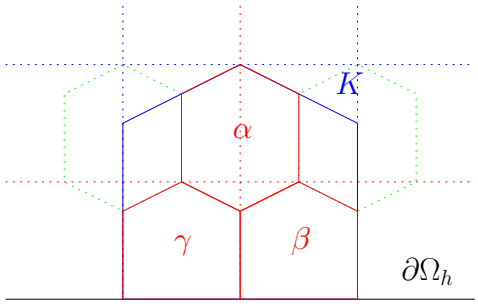
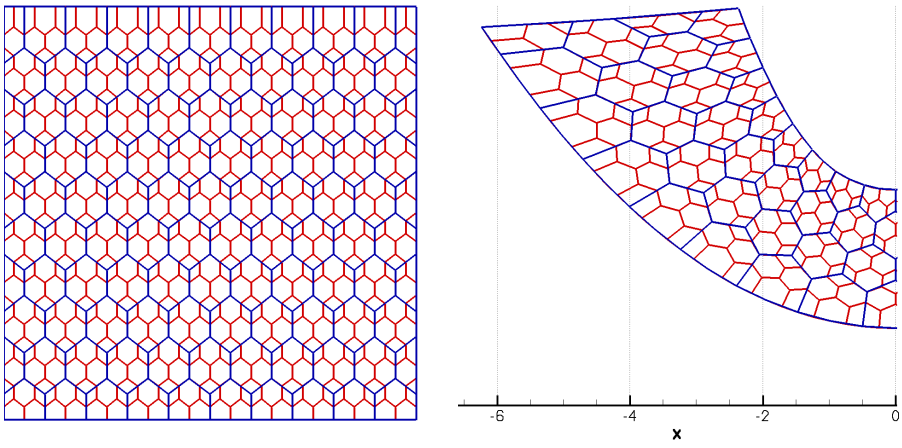


Figure 3.3: Collapsed hexagon and the corresponding refined grid.

lapsed hexagon K is splitted into two collapsed hexagons (β and γ), one complete hexagon α and two slices.

The grid generation method described so far, produces very good *quasi-nested* hexagonal meshes in any domain which can be decomposed by means of a structured mesh. Figure 3.4(a) and figure 3.4(b) show respectively the superimposition of two *quasi-nested* uniform hexagonal grids and the superimposition of two non-uniform hexagonal grids.



(a) Uniform case

(b) Non-uniform case

Figure 3.4: Hexagonal grid superimposition

3.2 Ringleb flow

We consider the solution of the Ringleb flow problem, that is one of the few non-trivial problems of $2D$ compressible Euler equations for which a (smooth) analytical solution is known. For this case the analytical solution may be obtained using the hodograph transformation, see [16]. The problem represents a transonic flow which turns around a symmetric obstacle, see figure 3.5, with inflow and outflow boundaries given by the left and right boundaries of the domain, and non absorbing, *i.e.* reflective boundaries with normal velocity $\mathbf{v} \cdot \mathbf{n} = 0$, on the lower and upper boundary.

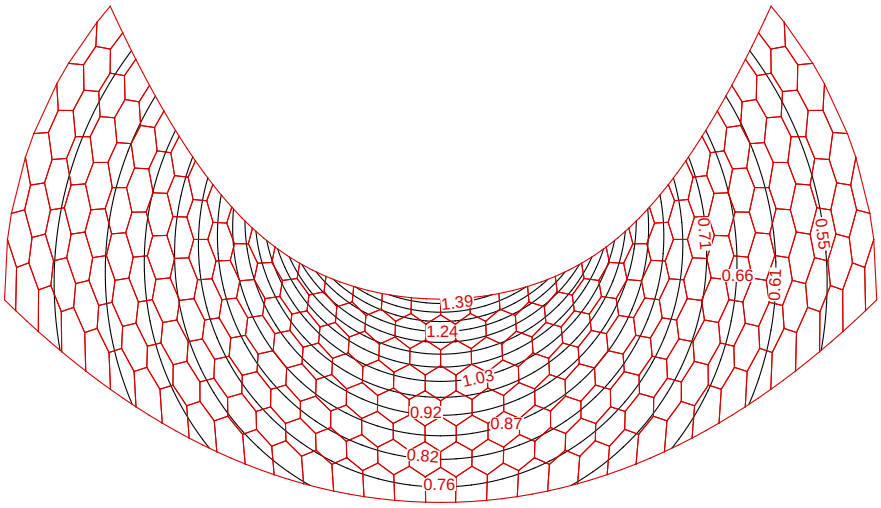


Figure 3.5: Intermediate hexagonal grid and Mach number iso-lines ($\mathbb{P}_6$) for the Ringleb solution

The solution to this flow problem is smooth yet transonic, with a small supersonic region around the nose. The exact solution of the Ringleb flow

problem reads :

$$\begin{aligned} x(q,k) &= \pm\frac{1}{k\,q\,\rho}\left(1-\frac{q^2}{k^2}\right)^{\frac{1}{2}}, \\ y(q,k) &= \frac{1}{2\rho}\left(\frac{1}{q^2}-\frac{2}{k^2}\right)+\frac{1}{2}J, \\ J &= \frac{1}{c}+\frac{1}{3c^3}+\frac{1}{5c^5}-\frac{1}{2}\ln\left(\frac{1+c}{1-c}\right), \end{aligned} \tag{3.1}$$

Then the computational domain is taken to be :

$$\Omega \;=\; \{\; \forall(x,y) : k \in (0.7, 1.2) \quad \text{et} \quad q \in (0.5, k)\;\}\,. \tag{3.2}$$

where k is a *stream-function*, *i.e.* it is constant along the streamlines, and can be expressed as :

$$k \;=\; \frac{q}{\sin\theta}\,,$$

being θ the angle between the velocity vector and the vertical direction in figure 3.5. Moreover q, c and ρ are dimensionless quantities, which equal, respectively, the ratio between local velocity magnitude and the speed of sound at the stagnation condition, c_0, the local speed of sound divided by c_0 and the ratio between local and stagnation fluid densities. Then, in order to express eq. (3.1) as a function of the sole q and k, we can use relations for an ideal gas undergoing an adiabatic process, which read :

$$\begin{aligned} \rho &= T^{\frac{1}{\gamma-1}} = c^{\frac{2}{\gamma-1}}, \\ p &= T^{\frac{\gamma}{\gamma-1}} = c^{\frac{2\gamma}{\gamma-1}}, \end{aligned}$$

where p and T are ratios of local values to stagnation ones, whereas

$$c = \left(1+\frac{\gamma-1}{2}M^2\right)^{-\frac{1}{2}},$$

which can be reformulated as

$$c = \left(1-\frac{\gamma-1}{2}q^2\right)^{\frac{1}{2}}\,.$$

Note that low order approximations of reflective boundaries reduce the order of convergence. To suppress this effect, given boundary conditions were enforced on the whole domain boundary, *i.e.* the exact solution, in eq. (3.1), was weakly imposed on $\partial\Omega$.

	Grid size	**triangles** $\|e_{\mathbf{u}}\|_2$	order	**quadrilaterals** $\|e_{\mathbf{u}}\|_2$	order	***hexagons*** $\|e_{\mathbf{u}}\|_2$	order
$\mathbb{P}_1$	16×4	1.07e−2	-	2.33e−2	-	8.62e−3	-
	32×8	2.87e−3	**1.90**	7.56e−3	**1.62**	1.93e−3	***2.15***
	64×16	7.46e−4	**1.94**	2.21e−3	**1.77**	4.51e−4	***2.10***
	128×32	1.90e−4	**1.97**	6.01e−4	**1.88**	1.09e−4	***2.04***
$\mathbb{P}_2$	16×4	5.57e−4	-	1.27e−3	-	7.83e−4	-
	32×8	7.17e−5	**2.96**	2.05e−4	**2.63**	1.13e−4	***2.79***
	64×16	9.14e−6	**2.97**	3.00e−5	**2.77**	1.34e−5	***3.08***
	128×32	1.13e−6	**3.01**	4.06e−6	**2.88**	1.70e−6	***2.98***
$\mathbb{P}_3$	16×4	5.48e−5	-	1.57e−4	-	1.13e−4	-
	32×8	3.64e−6	**3.91**	1.10e−5	**3.82**	9.38e−6	***3.60***
	64×16	2.30e−7	**3.98**	7.46e−7	**3.89**	7.23e−7	***3.70***
	128×32	1.46e−8	**3.98**	4.77e−8	**3.97**	5.26e−8	***3.78***
$\mathbb{P}_4$	16×4	8.03e−6	-	2.64e−5	-	1.80e−5	-
	32×8	2.83e−7	**4.82**	1.10e−6	**4.58**	7.12e−7	***4.66***
	64×16	8.90e−9	**4.99**	3.80e−8	**4.86**	2.38e−8	***4.90***
	128×32	2.75e−10	**5.01**	1.20e−9	**4.99**	8.51e−10	***4.81***
$\mathbb{P}_5$	16×4	1.08e−6	-	5.46e−6	-	3.15e−6	-
	32×8	2.30e−8	**5.56**	1.21e−7	**5.49**	7.43e−8	***5.40***
	64×16	3.55e−10	**6.01**	2.12e−9	**5.84**	1.40e−9	***5.72***
	128×32	5.57e−12	**6.00**	3.40e−11	**5.97**	2.55e−11	***5.78***
$\mathbb{P}_6$	16×4	1.53e−7	-	1.24e−6	-	6.32e−7	-
	32×8	1.75e−9	**6.45**	1.54e−8	**6.33**	8.63e−9	***6.20***
	64×16	1.49e−11	**6.89**	1.34e−10	**6.85**	1.10e−10	***6.28***
	128×32	1.17e−13	**6.98**	1.10e−12	**6.93**	1.19e−12	***6.55***

Table 3.1: Convergence results for $\|e_{\mathbf{u}}\|_2$

For the convergence study were used twelve grids, four recursively nested grids, 16×4, 32×8, 64×16 and 128×32 elements, for three kinds of element, triangles, quadrangles and hexagons. Hexagonal grids are just partially nested and boundary hexagons are collapsed on the boundary, see figure 3.8. The output of interest in this case was the L_2 norm of the state error, which reads :

$$\|e_{\mathbf{u}}\|_2 = \left(\frac{\int_{\Omega_h} (\mathbf{u}_h - \mathbf{u}) \cdot (\mathbf{u}_h - \mathbf{u}) \, \mathrm{d}\mathbf{x}}{|\Omega_h|} \right)^{\frac{1}{2}} , \tag{3.3}$$

where $\mathbf{u}$ and $\mathbf{u}_h$ are respectively exact and numerical solution vectors in primitive variables.
The results in figure 3.9 and in table 3.1 show that optimal error convergence, $O(h^{p+1})$, is attained. Even though hexagonal meshes do not attain the $O(h^{p+1})$ order of convergence for $\mathbb{P}_5$ and $\mathbb{P}_6$ discretizations, the trend suggests that, reducing further the mesh size, the optimal error of convergence should be attained.

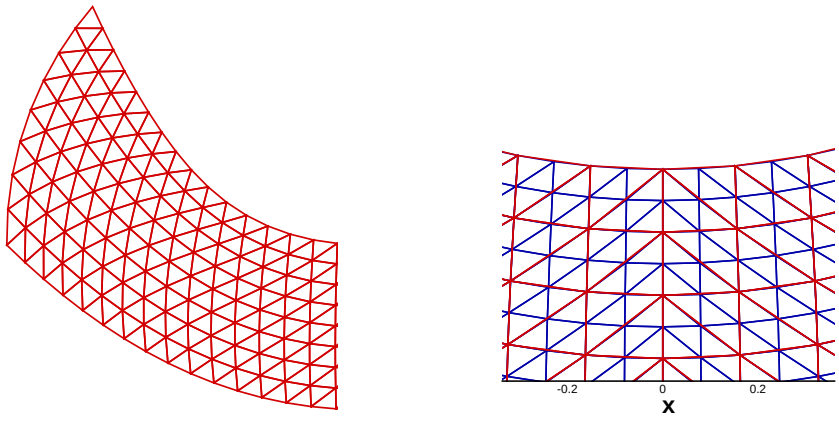


Figure 3.6: Triangular grids

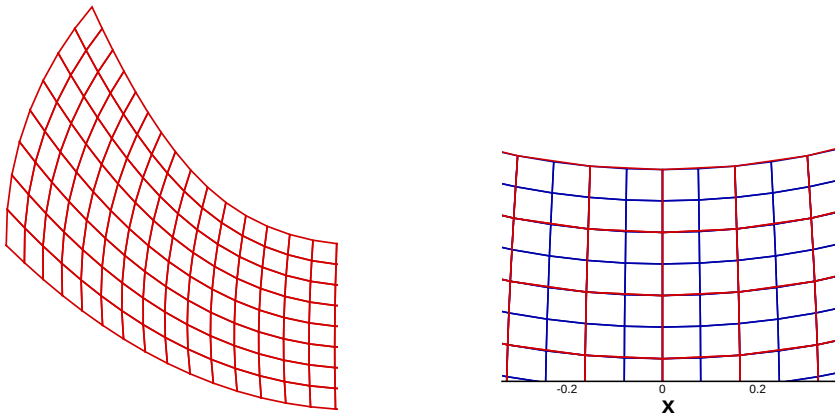


Figure 3.7: Quadrangular grids

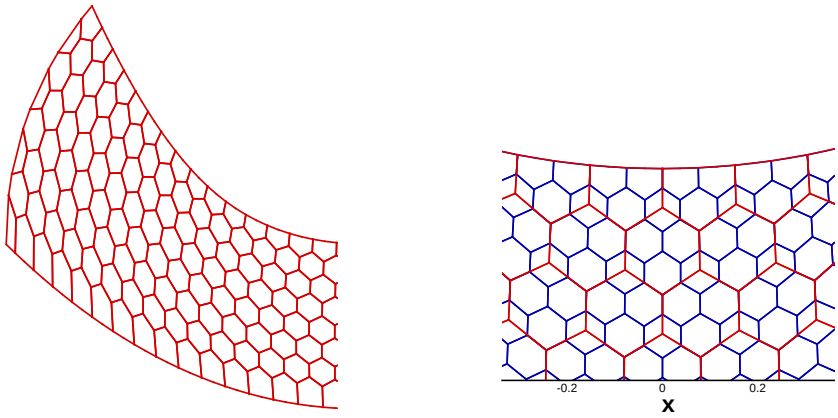
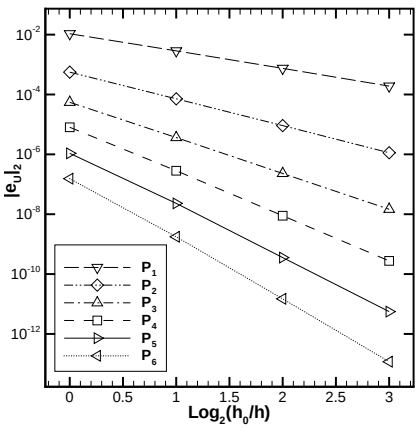
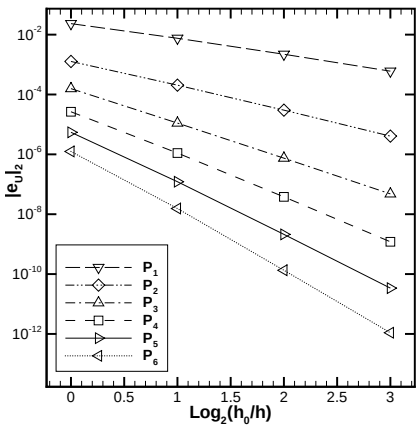


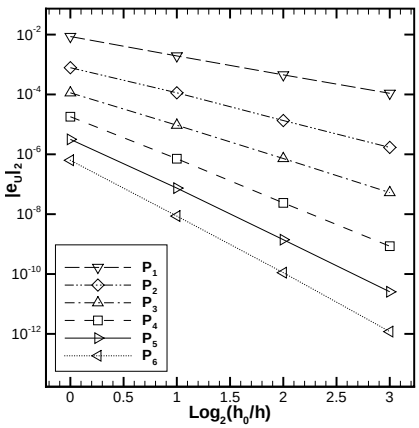
Figure 3.8: Hexagonal grids



(a) triangles

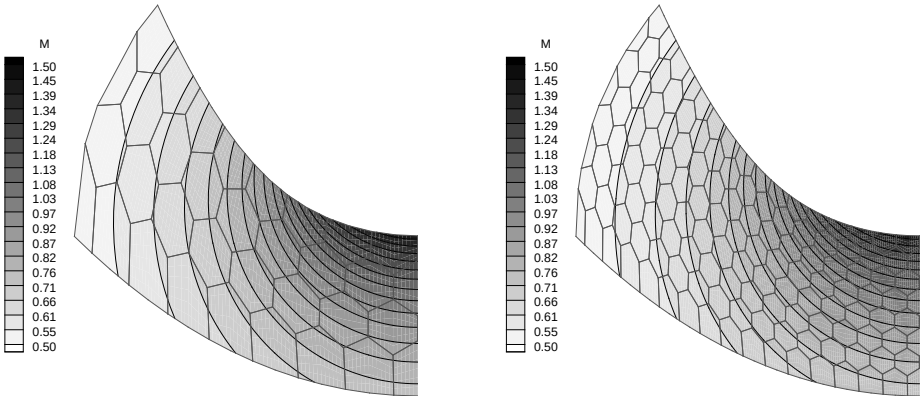


(b) quadrangles



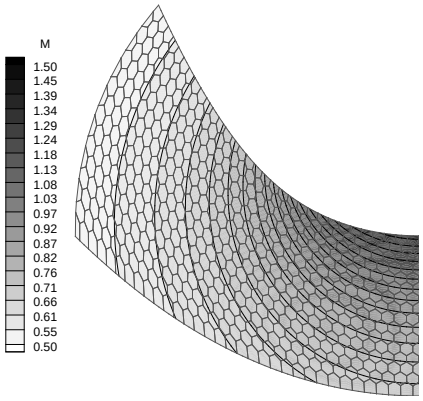
(c) hexagons

Figure 3.9: $\|e_{\mathbf{u}}\|_2$ convergence curves.



(a) 16×4 grid

(b) 32×8 grid



(c) 64×16 grid

Figure 3.10: Mach number iso-contours for hexagonal grids ($\mathbb{P}_6$).

3.3 Gaussian bump perturbation

Another representative, smooth inviscid case is that of duct flow over a Gaussian bump perturbation. The domain,

$$\Omega = \left\{ \forall (x,y) : x \in (-1,1) \quad \text{et} \quad y \in (0.05e^{-50x^2}, 1) \right\} \tag{3.4}$$

and an intermediate hexagonal mesh are depicted in figure 3.11.
Non absorbing wall boundary conditions were enforced on the top and on the bottom of the channel. At the outflow, the static pressure was set, and at the inflow, the total temperature, total pressure and flow angle (0^o) were prescribed, resulting in a free stream Mach number of $0,5$.

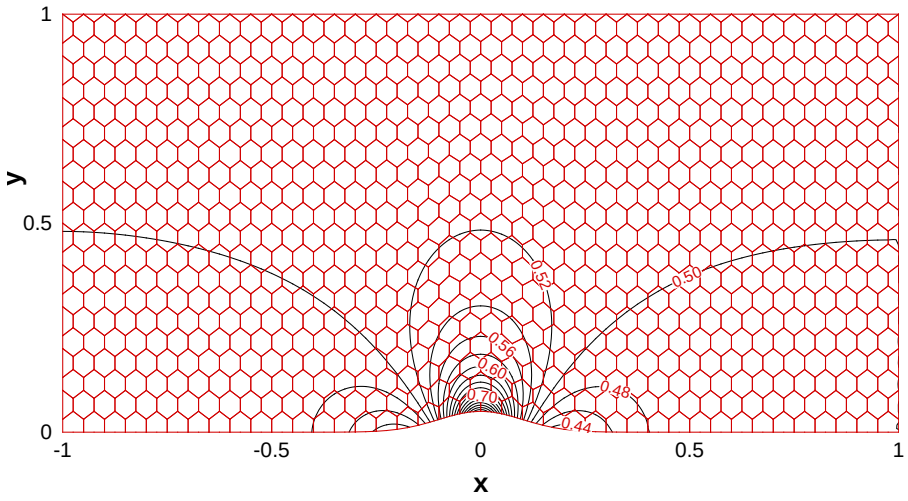


Figure 3.11: Intermediate hexagonal grid and Mach number iso-lines ($\mathbb{P}_4$) for the Gaussian bump perturbation problem

For the convergence study were used four recursively partially nested hexagonal grids, 220, 840, 3280 and 12960 elements. Furthermore the bump

geometry was represented using high order polynomials.
The output of interest in this case is the L_2 norm of the entropy error,

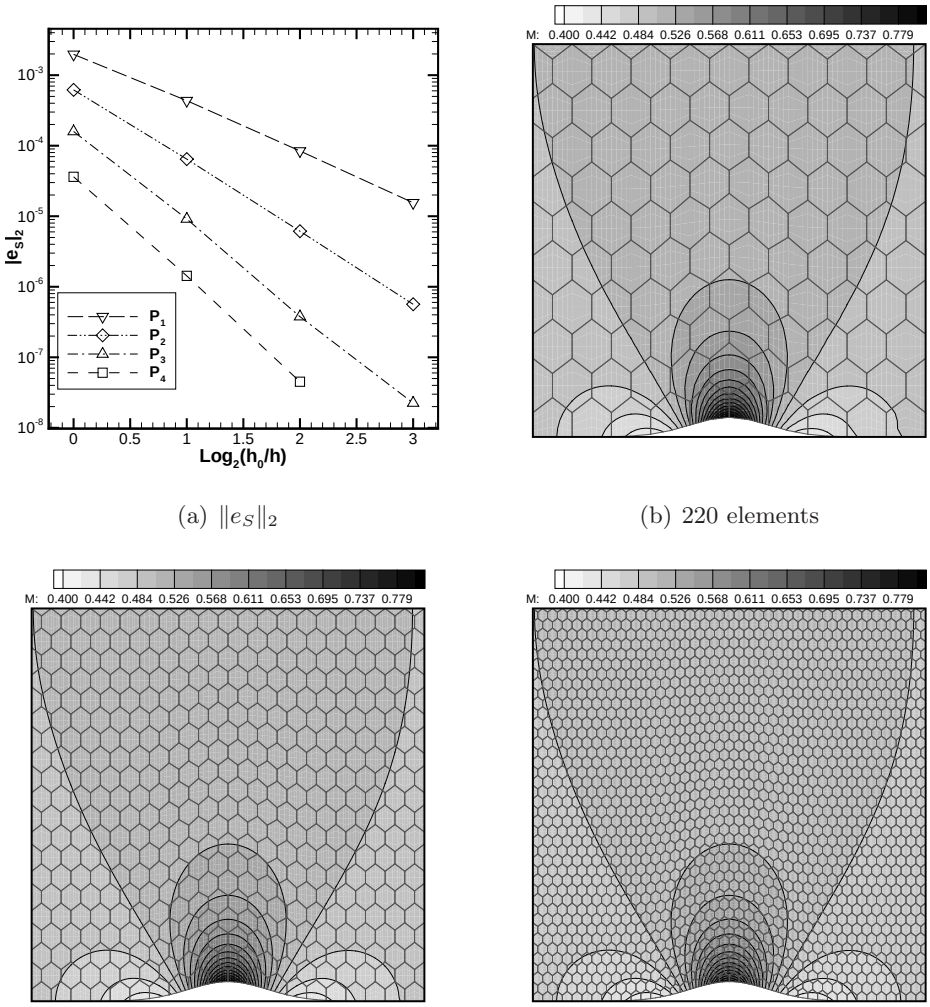
$$\|e_S\|_2 \;=\; \left(\frac{\int_{\Omega_h} (S - S_{fs})^2 \, \mathrm{d}\mathbf{x}}{|\Omega_h|} \right)^{\frac{1}{2}} , \tag{3.5}$$

where S_{fs} is the free stream entropy.

Results in figure 3.12 and in table 3.2 show that optimal error convergence, $O(h^{p+1})$, is attained. This test assesses the accuracy of the discontinuous Galerkin approximation of inviscid problems on polyhedral grids, when enforcing "classic" boundary conditions.

Grid size		$\|e_S\|_2$	order		$\|e_S\|_2$	order
220	$\mathbb{P}_1$	1.97e−3	-	$\mathbb{P}_2$	6.18e−4	-
840		4.35e−4	**2.18**		6.45e−5	**3.26**
3280		8.42e−5	**2.37**		6.13e−6	**3.40**
12960		1.55e−5	**2.44**		5.64e−7	**3.44**
220	$\mathbb{P}_3$	1.59e−4	-	$\mathbb{P}_4$	3.62e−5	-
840		9.14e−6	**4.12**		1.43e−6	**4.66**
3280		3.77e−7	**4.60**		4.51e−8	**4.99**
12960		2.24e−8	**4.06**		-	-

Table 3.2: Convergence results for $\|e_S\|_2$



(a) $\|e_S\|_2$

(b) 220 elements

(c) 840 elements

(d) 3280 elements

Figure 3.12: $\|e_S\|_2$ and Mach number iso-contours ($\mathbb{P}_4$) for hexagonal grids

3.4 Helmholtz problem

Laking of an analytical non-trivial solution of compressible Navier-Stokes equations, it was necessary to choose a simple model problem to assess the order of convergence of discontinuous Galerkin approximations for problems associated to second order PDEs. Thus, the Helmholtz problem, in eq. (3.6), was chosen to verify the spatial accuracy of viscous numerical fluxes on polyhedral grids. The figure below depicts both the domain Ω and the intermediate polygonal grid, which was used for the error, see eq. (3.7), convergence study.

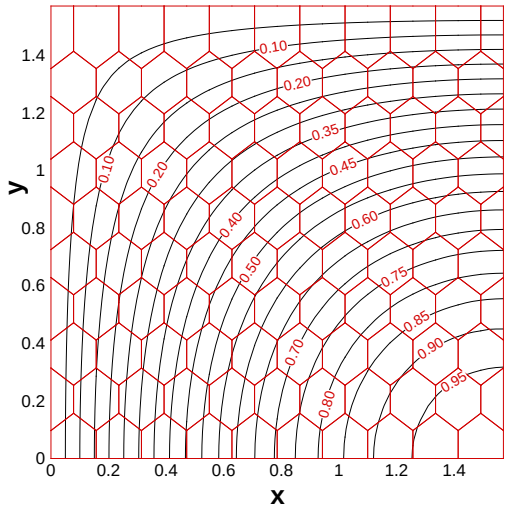


Figure 3.13: Intermediate hexagonal grid and solution iso-lines ($\mathbb{P}_3$) for Helmholtz problem

The considered linear reaction-diffusion problem reads :

$$\begin{aligned} \boldsymbol{\nabla}^2 u(x,y) + 2\,u(x,y) = 0 \qquad & \forall (x,y) \in \Omega, \\ u(x,y) = sin(x)cos(y) \qquad & \forall (x,y) \in \partial\Omega\,, \end{aligned} \tag{3.6}$$

where

$$\Omega = \left(0, \frac{\pi}{2}\right) \times \left(0, \frac{\pi}{2}\right)\,.$$

Given boundary conditions were enforced on the whole boundary of the domain, $\partial\Omega$. For the convergence study were used three recursively partially nested hexagonal grids, 27, 105 and 410 elements.

The output of interest in this case was the L_2 norm of the state error,

$$\|e_u\|_2 \;=\; \left(\frac{\int_{\Omega_h} (u_h - u)^2 \; \mathrm{d}\mathbf{x}}{|\Omega_h|} \right)^{\frac{1}{2}}\,, \tag{3.7}$$

where u and u_h are respectively the exact and the numerical solution of the problem.

Results in figure 3.14 and in table 3.3 show that optimal error convergence, $O(h^{p+1})$, is attained.

ne		$\|e_u\|_2$	order		$\|e_u\|_2$	order		$\|e_u\|_2$	order
27		3.12e−3	-		1.28e−4	-		4.74e−6	-
105	$\mathbb{P}_1$	7.88e−4	**1.98**	$\mathbb{P}_2$	1.51e−5	**3.08**	$\mathbb{P}_3$	2.63e−7	**4.17**
410		2.00e−4	**1.98**		1.83e−6	**3.04**		1.64e−8	**4.00**

Table 3.3: Convergence results for $\|e_u\|_2$

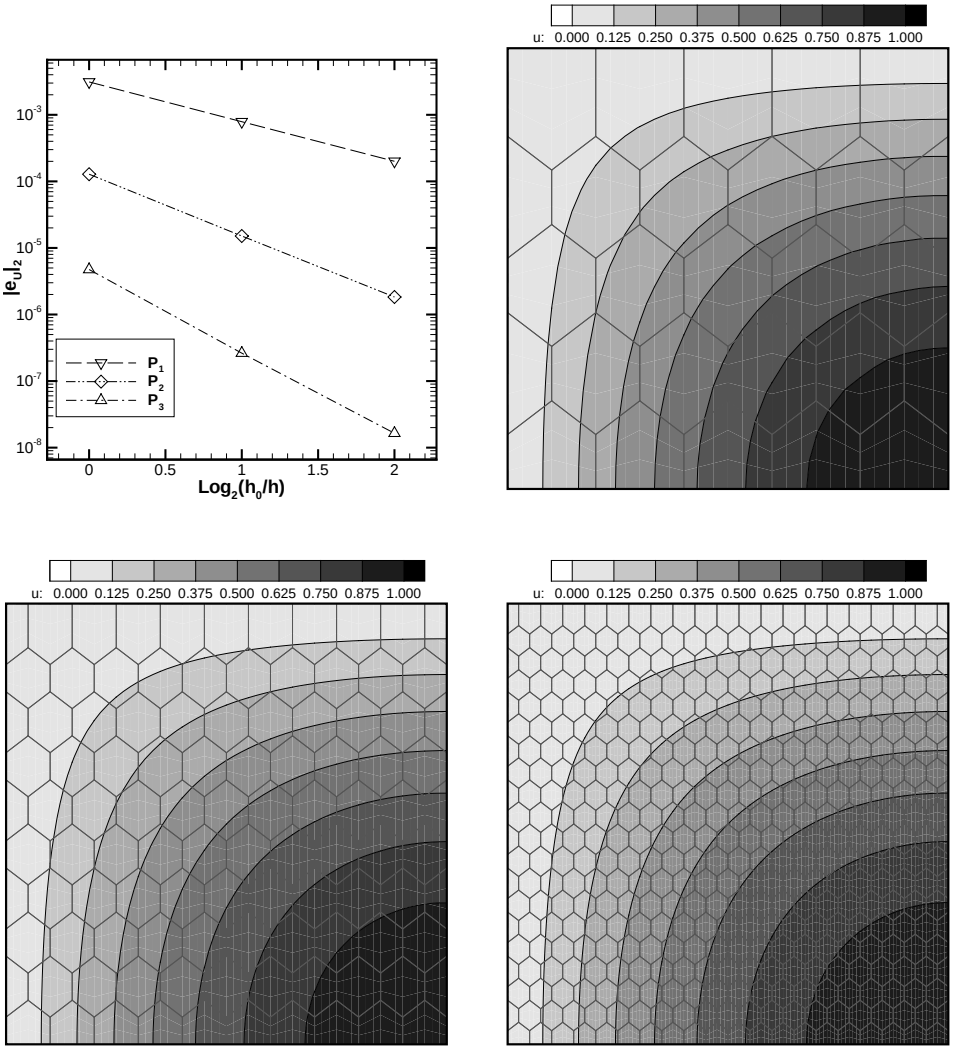


Figure 3.14: $\|e_u\|_2$ and u iso-contours ($\mathbb{P}_3$) for hexagonal grids (27, 105 and 410 elements)

3.5 Viscous flow over a NACA 0012 airfoil

The discontinuous Galerking approximation of compressible Navier-Stokes equations on polygonal grids was tested in two viscous flow conditions around the NACA 0012 airfoil. The two cases, referred as *Re5k* and *Re73*, are respectively characterized by $\mathrm{Re}_{\infty,1} = 5000$, $\alpha = 0^o$, $\mathrm{M}_\infty = 0,5$ and $\mathrm{Re}_{\infty,1} = 73$, $\alpha = 10^o$, $\mathrm{M}_\infty = 0,8$, where α denotes the angle of flow incidence.

In both the cases the free-stream boundary was located at 5 chords from the leading edge and no-slip adiabatic boundary conditions were imposed on the airfoil boundary. Futhermore, in order to achieve high-order accuracy, solid wall boundaries of the airfoil were represented by means of piecewise cubic polynomials for all orders of solution approximation.

The polyhedral meshes were derived from an unstructured grid of 2048 triangles, which will be referred as the source grid. Triangles were merged into polygons by optimizing a function that captures the overall quality of the fused elements. Therefore it was employed an algorithm which minimizes both the maximum and the average of all elemental aspect ratios of the resulting grid [17].
The agglomeration process was controlled by setting the maximum number of sub-elements which compose a polygonal element. Then, in order to solve accurately the boundary layer, the grid generation for cases *Re5k* and *Re73* was performed by limiting the number of sub-elements per polygon to three and to five, respectively. The process resulted in two polygonal grids of 853 and 461 elements.

In figures 3.15, 3.16, 3.17 and 3.18, are displayed both the Mach number iso-contours and the polygonal grids, resulting in the two considered cases, *Re5k* and *Re73*. In both cases, the discontinuities across element interfaces diminish when increasing the order of the discretization, suggesting that accuracy increases with the order of the polynomial approximation.

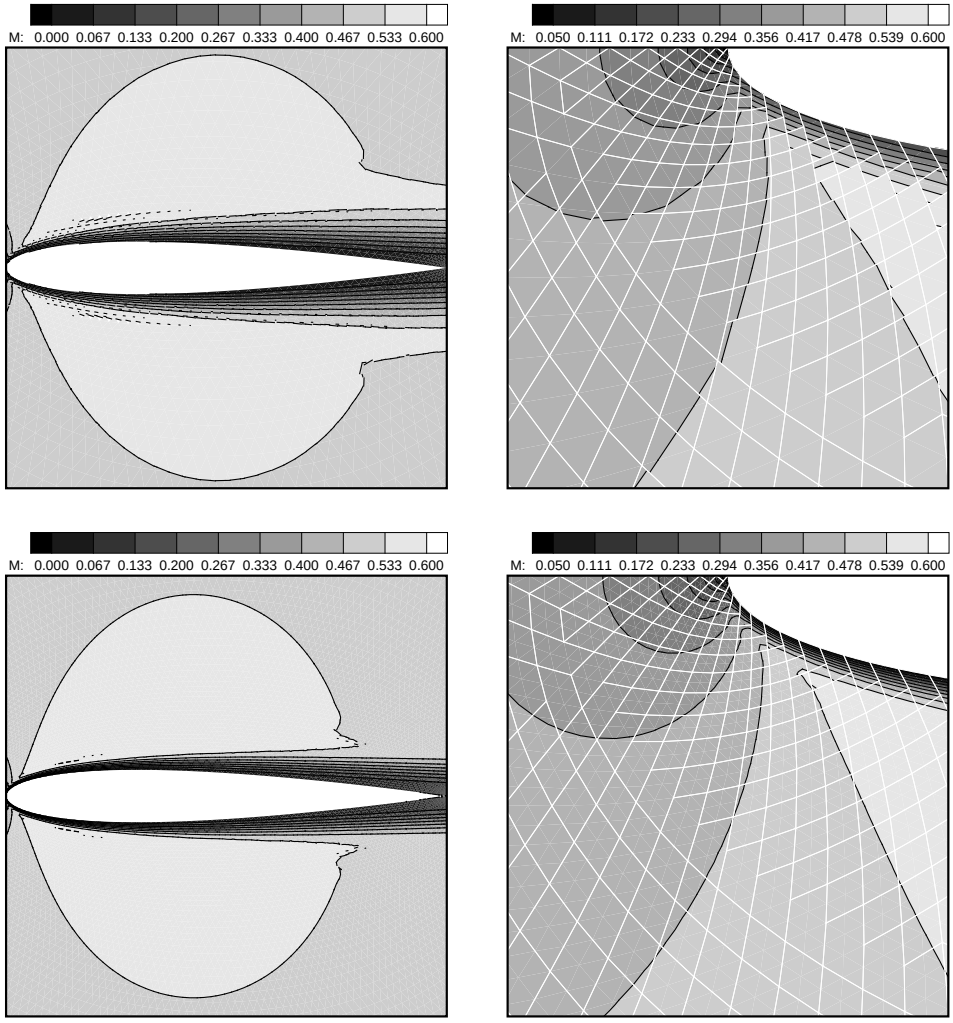


Figure 3.15: Mach number iso-contours, case *Re5k*. From the top to the bottom : $\mathbb{P}_1$ and $\mathbb{P}_2$, discretizations.

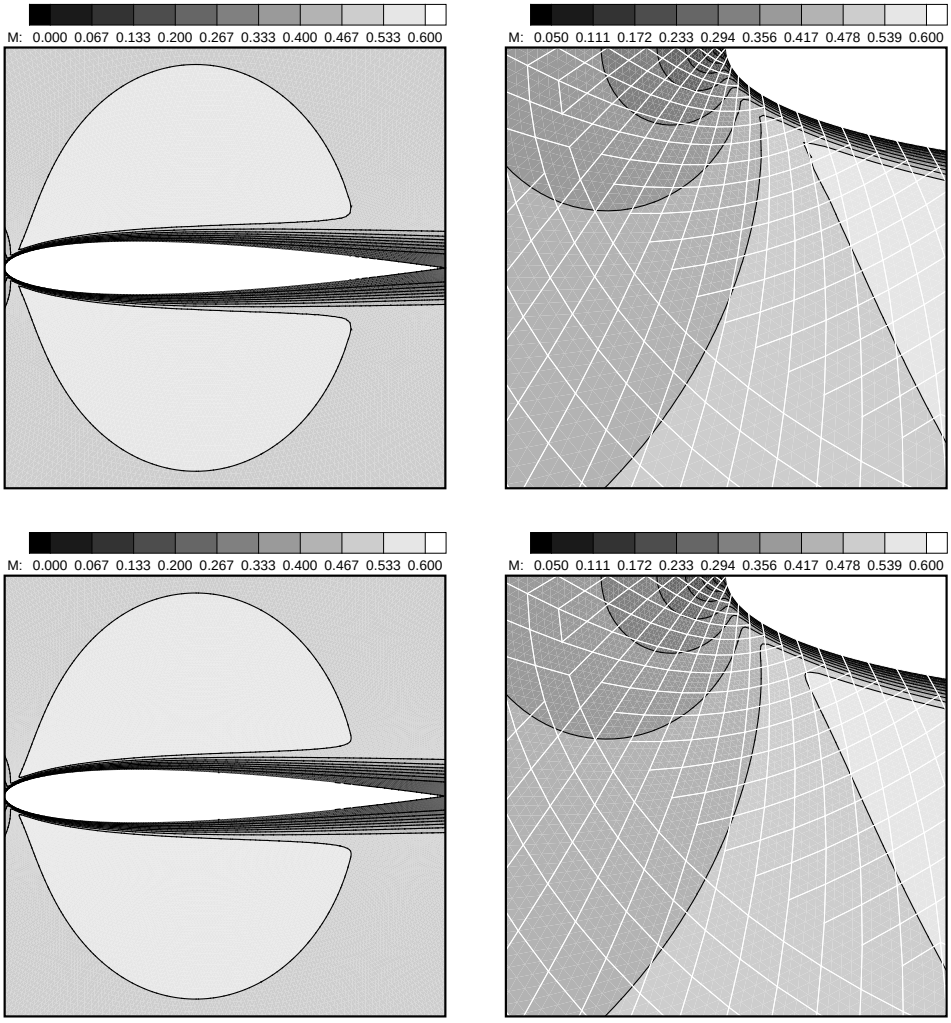


Figure 3.16: Mach number iso-contours, case *Re5k*. From the top to the bottom : $\mathbb{P}_3$ and $\mathbb{P}_4$ discretizations.

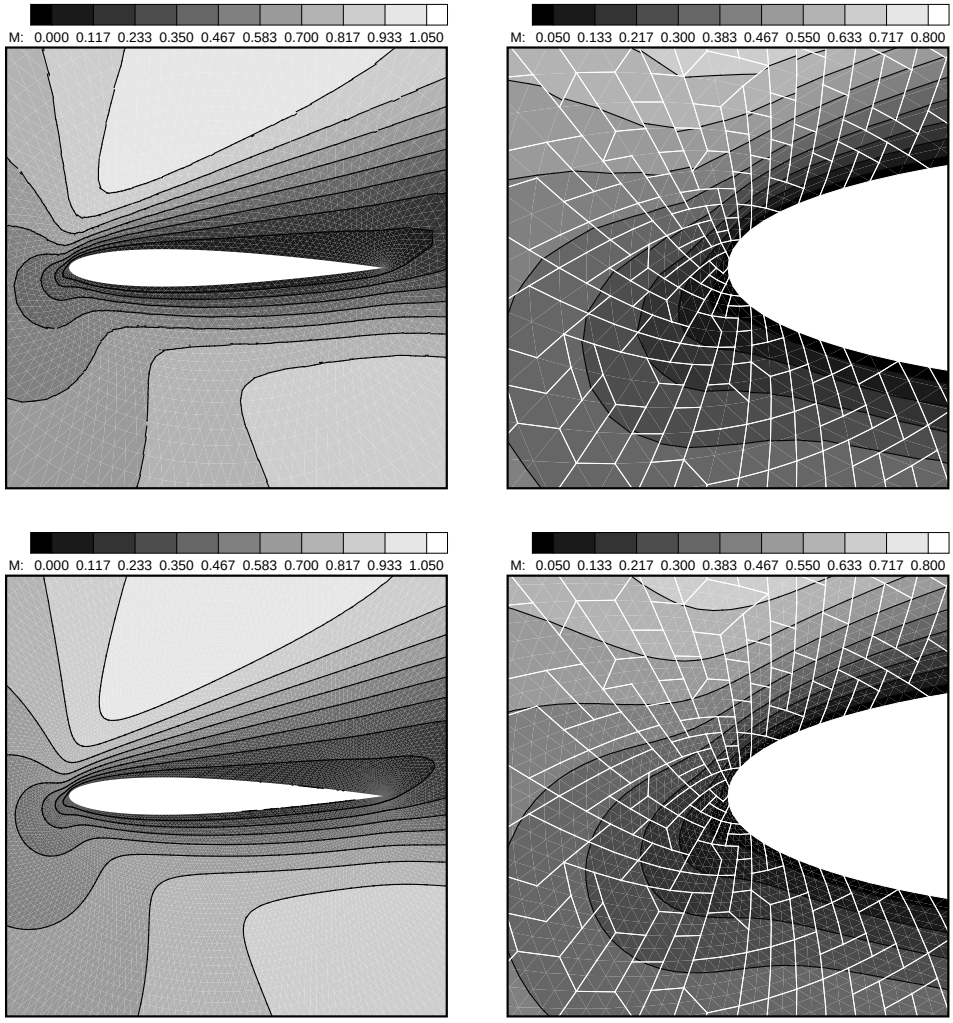


Figure 3.17: Mach number iso-contours, case *Re73*. From the top to the bottom $\mathbb{P}_1$ and $\mathbb{P}_2$ discretizations.

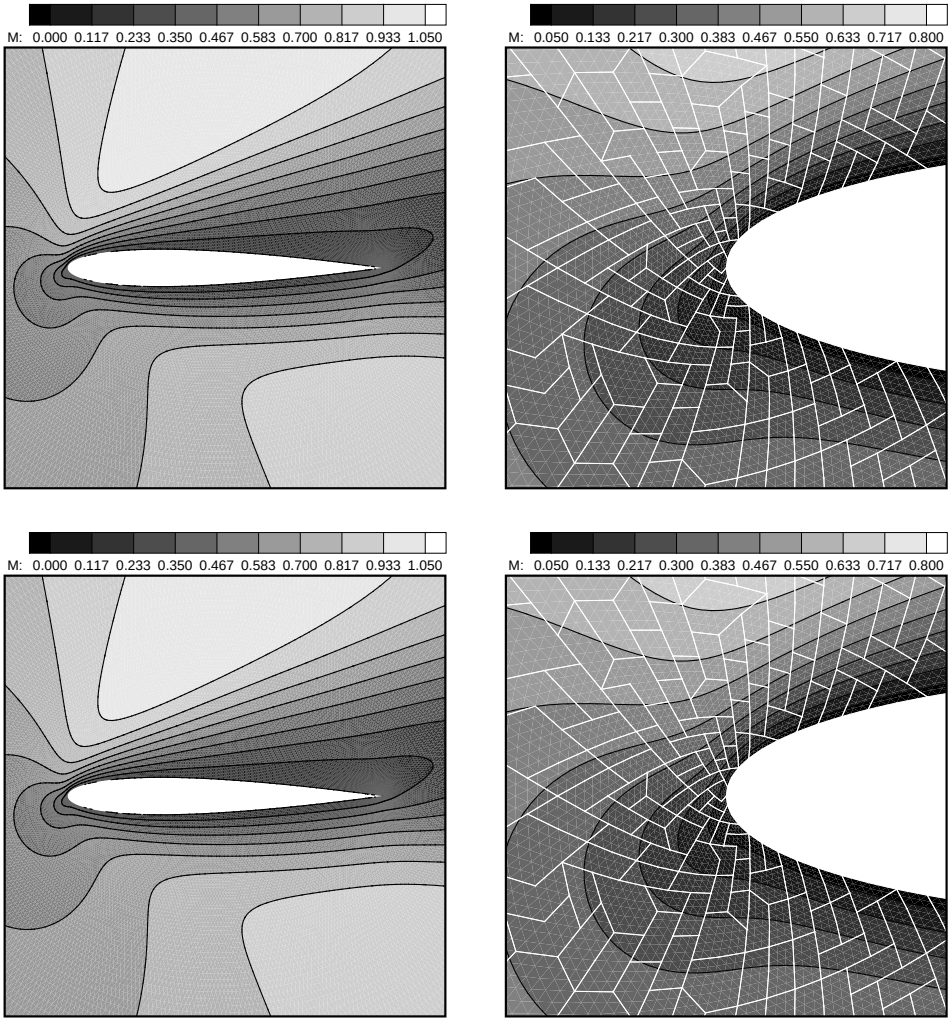


Figure 3.18: Mach number iso-contours, case *Re73*. From the top to the bottom $\mathbb{P}_3$ and $\mathbb{P}_4$ discretizations.

3.6 Reduced integration rules

In order to test the effect of reduced integration rules on both the accuracy of the approximated solution and the CPU time, the inviscid subsonic flow around the NACA 0012 airfoil was considered. Indeed, in this case, the accuracy of the state discretization can be evaluated by means of the entropy error, as defined in eq. (3.5).
The problem is that of an inviscid subsonic flow at zero incidence, with a free-stream Mach number equal to 0,5. Free-stream boundary conditions were located at 50 chords from the leading edge, whereas non-absorbing wall boundary conditions were enforced on the airfoil solid wall, which was represented by piecewise 4^{th} order polynomials, for all approximation degrees of the solution.

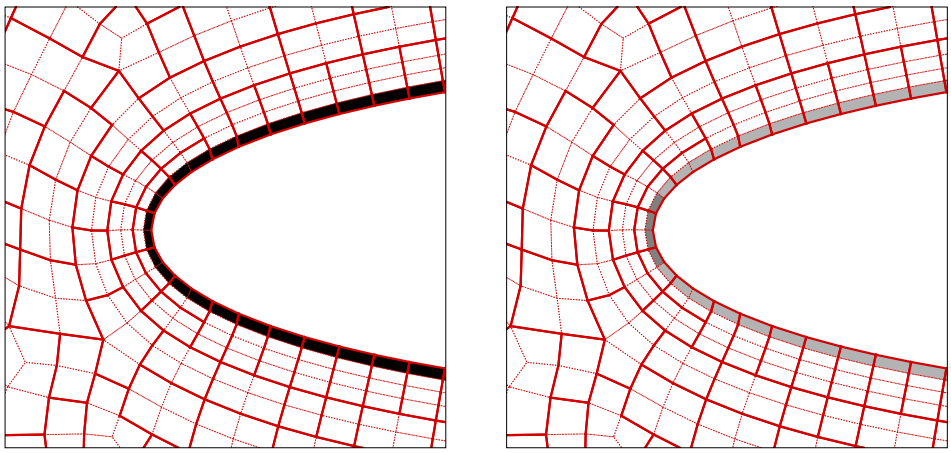


Figure 3.19: Comparison of cubatures used for exact (left) and reduced (right) integration of the $\mathbb{P}_1$ solution approximation on a polygonal grid. Legend : *white = 4 nodes, light gray = 9 nodes, dark gray = 16 nodes, black = 81 nodes*

The computational grid was constructed by agglomerating the 2249 ele-

ments of a hybrid unstructured mesh in polygons of five sub-regions maximum. As done in previous section, the merging procedure was performed by the **MGridGen 1.0** algorithm, which minimizes the overall grid aspect ratio [17]. A detail of the polyhedral grid (thick red line) and of the sub-elements used for integration (dotted red line) is depicted in figure 3.19.

Results, obtained on the previously described grid, are presented for five polynomial approximations using both exact and reduced integration rules. The actual cubature reduction method relies on the criterion defined in eq. (2.21), where scalar *tol* was set to 5e−6. Furthermore, it is important to point out that integration rules were reduced not only on elements, but also on their faces. The performed cubature order reduction does not affect the entropy error and, thus, the accuracy. Conversely, being constant the admitted error *tol*, the more the order of the polynomial approximation increases the more the cubature order reduction enhances computational efficiency.

	CPU time MC		CPU time RC		$\text{CPU}_{MC}/\text{CPU}_{RC}$		
	RK3	NEJ	RK3	NEJ	RK3	NEJ	$\|e_S\|_2$
$\mathbb{P}_1$	0.164s	0.125s	0.121s	0.102s	1.355	1.225	1.62e−5
$\mathbb{P}_2$	0.664s	0.523s	0.470s	0.386s	1.413	1.356	1.72e−6
$\mathbb{P}_3$	1.75s	1.50s	1.22s	1.07s	1.431	1.402	5.13e−7
$\mathbb{P}_4$	6.33s	5.27s	4.24s	3.64s	1.500	1.450	3.50e−7
$\mathbb{P}_5$	12.6s	11.4s	8.18s	7.59s	1.544	1.502	2.21e−7

Table 3.4: CPU time reduction table. RK3 and NEJ refer to the three steps Runge Kutta explicit and to the non-linear element jacobi time integration schemes, respectively

Table 3.4 compares the computational efficiency of reduced integration rules

with that of exact ones for five polinomial degree approximations, using both explicit and block implicit schemes. For both time integration schemes the reduced number of cubature nodes is beneficial to efficiency.
Figures 3.20 and 3.21 show that the more the polynomial degree of the discretization increases the more the discontinuities across the element interfaces diminish. Furthermore the entropy error in the L_2 norm, (2.21), decreases when rising the polynomial degree of the discretization, table 3.4. Then accuracy grows likewise the approximation degree does.

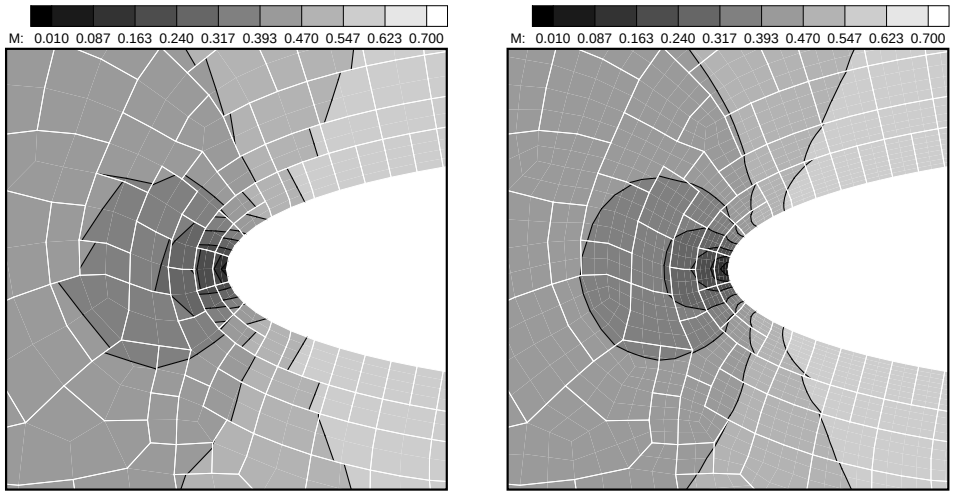


Figure 3.20: Mach number iso-contours, $\mathbb{P}_1$, $\mathbb{P}_2$ discretizations.

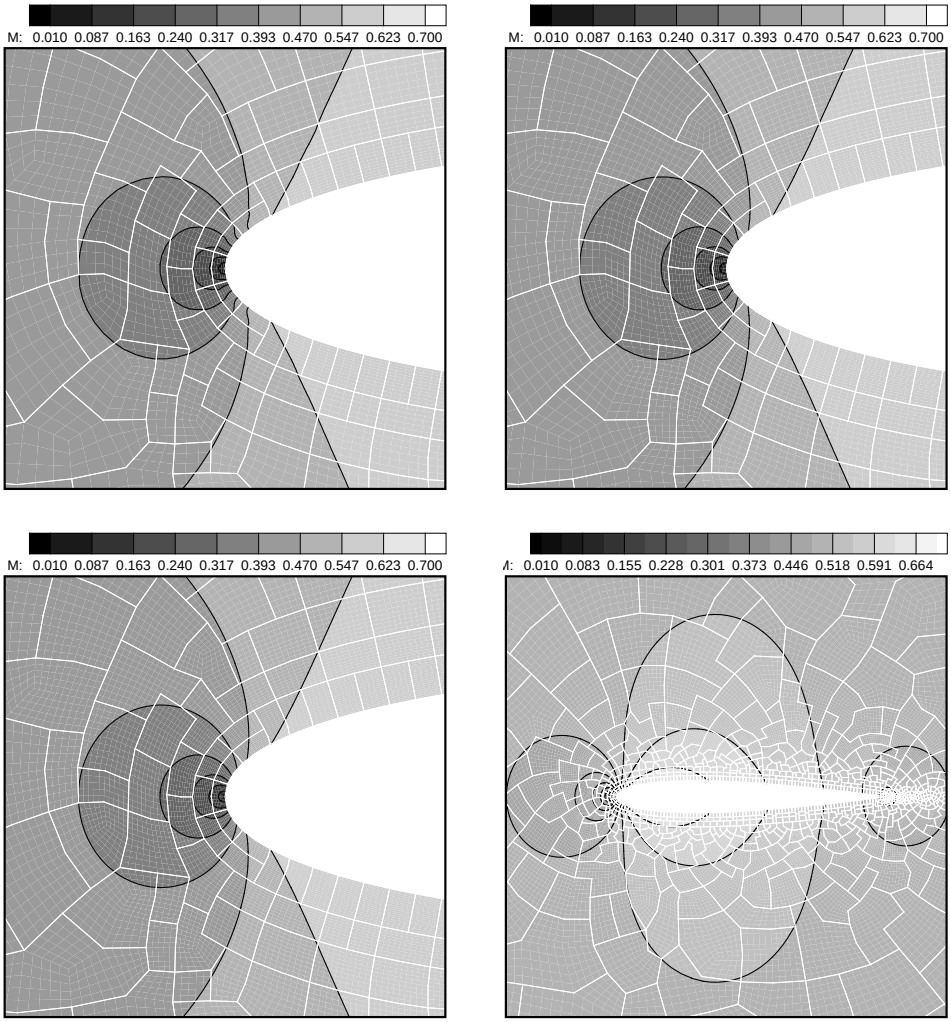


Figure 3.21: Mach number iso-contours, $\mathbb{P}_3$, $\mathbb{P}_4$ and $\mathbb{P}_5$ discretizations.

Chapter 4

Multigrid Solver

Hereinafter the discontinuous Galerkin discretization on polyhedral grids is used to develop an h-multigrid method at constant polynomial degree.
First an elegant yet practical set of transfer operators, which are well-suited for hp-multigrid approaches, is derived for general space settings. Then, their properties for current space settings, *i.e.* for hierarchic orthonormal sets of shape functions, are presented. After that, a quasi-implicit multistage h-multigrid iteration strategy for the discontinuous Galerkin discretization of the steady Euler equations is developed and investigated experimentally. Coarse level discretizations are constructed by coarsening the grid at constant polynomial degree. Fine elements are merged by optimizing a function that capture the overall quality of the fused elements, those elements are polygons in $2D$ and polyhedra in $3D$. The agglomeration strategy was developed by I. Moulitsas and G. Karypis [17], some details about this process were presented in section 3.5.
Finally, a multistage V-cycle is developed using the non-linear FAS (full approximation storage) multigrid scheme. Results are presented for an uniform flow over a NACA 0012 airfoil at 2^o of incidence and $\mathrm{M}_\infty = 0,5$. In section 4.4 are presented preliminary tests which point out some of the most considerable properties of the developed multigrid scheme and its

differences with the "classic" p-multigrid scheme. In section 4.5, those considerations are summarized and suitable improvements are presented.

4.1 Definition of transfer operators

[1] In this section we present in detail an elegant yet practical set of *transfer operators*, which are well-suited for both h- and p- multigrids applied to discontinuous Galerkin finite element methods.
Transfer operators aim to "relate" two nested interpolation spaces. Let us consider a fine interpolation space $\mathbf{V}_h$, see eq. (2.2), and another interpolation space, coarser than $\mathbf{V}_h$, such that $\mathbf{V}_H \subseteq \mathbf{V}_h$. In order to derive coarse space $\mathbf{V}_H$ from $\mathbf{V}_h$, we can both reducing the degree of the polynomial approximation and agglomerating the elements of the fine triangulation $\mathcal{T}_h = \{K_i^h\}_{i=1}^n$, getting a less resolved triangulation $\mathcal{T}_H = \{K_j^H\}_{j=1}^m$, *i.e.* we

[1] This section aims to define transfer operators in very detail for both h- and p-multigrid approaches and it requires a quite heavy notation.
Then we summarize here all the indices which will be introduced in the text :

- h , fine-level
- H, coarse-level
- K_i^h, element on the fine-level
- K_j^H, element on the coarse-level
- nv_i, number of *d.o.f.*s in fine-element K_i^h
- nv_j, number of *d.o.f.*s in coarse-element K_j^H
- n, number of fine-elements in $\mathcal{T}_h$, or in coarse-element K_j^H
- m, number of coarse elements in $\mathcal{T}_H$
- a, e, indices for fine *d.o.f.*s
- o, s, indices for coarse *d.o.f.*s
- i, p, indices for fine-elements
- j , q, indices for coarse-elements

assume the following space settings :

$$\begin{aligned} \mathbf{u}_H \in \mathbf{V}_H &\stackrel{\text{def}}{=} [\,V_H\,]^N \qquad \text{and} \\ \mathbf{u}_h \in \mathbf{V}_h &\stackrel{\text{def}}{=} [\,V_h\,]^N \ , \end{aligned} \tag{4.1}$$

where N is the number of unknowns,

$$\begin{aligned} V_H &\stackrel{\text{def}}{=} \left\{ v_H \in L^2(\Omega_H) : \ v_H|_{K_j^H} \in \mathbb{P}_{k_H}(K_j^H) \ \forall K_j^H \in \mathcal{T}_H \right\} \quad \text{and} \\ V_h &\stackrel{\text{def}}{=} \left\{ v_h \in L^2(\Omega_h) : \ v_h|_{K_i^h} \in \mathbb{P}_{k_h}(K_i^h) \ \forall K_i^h \in \mathcal{T}_h \right\} , \end{aligned}$$

being $\mathbb{P}_{k_H}(K_j^H)$ and $\mathbb{P}_{k_h}(K_i^h)$ the spaces of polynomials of global degrees at most k_H on K_j^H and k_h on K_i^h, respectively.

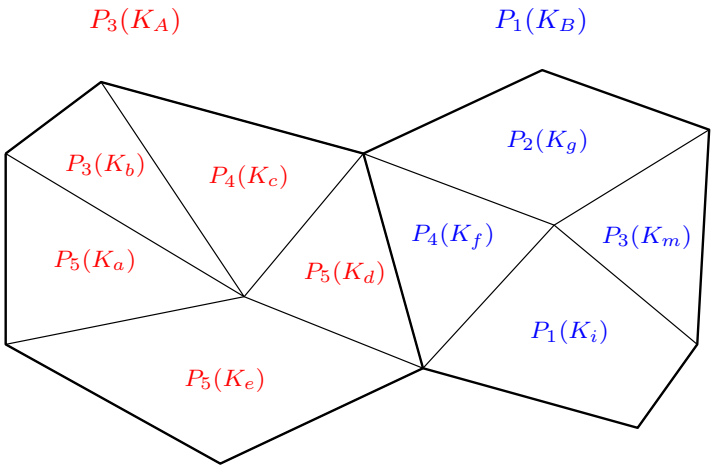


Figure 4.1: Superimposition of fine and coarse nested interpolation spaces

We note that, as sketched in figure 4.1, in order to obtain nested spaces, the global degree of the polynomial approximation k_H in any coarse element

K_j^H is such that $k_H \leq k_h$ for all $K_i^h \subseteq K_j^H$, where k_h are polynomial approximation degrees in fine elements K_i^h.
Hereinafter, we first present the principal transfer operators in the general case and then we reduce them for a set of hierarchic orthonormal shape functions.

4.1.1 State transfer operators

Both *prolongation* and *restriction operators* for the solution (state) are based on the L_2 or Galerkin projection. The restriction operator aims to project the fine state

$$\mathbf{u}_h = \sum_{i=1}^{n} \sum_{a=1}^{nv_i} \mathbf{U}_{a,\,i}^h \; \varphi_{a,\,i}^h \,, \tag{4.2}$$

belonging to $\mathbf{V}_h$, onto the coarse space $\mathbf{V}_H$. Thus, the restriction of $\mathbf{u}_h$ is equivalent to find the coarse state

$$\mathbf{u}_H = \sum_{j=1}^{m} \sum_{o=1}^{nv_j} \mathbf{U}_{o,\,j}^H \; \varphi_{o,\,j}^H \,, \tag{4.3}$$

such that

$$\left\langle \mathbf{u}_h - \mathbf{u}_H,\, \varphi_{o,\,j}^H \right\rangle_{K_j^H} \;=\; \mathbf{0} \qquad \forall o \in \mathbb{N}_{(1,\,nv_j)} \quad \text{and} \quad \forall j \in \mathbb{N}_{(1,\,m)} \,, \tag{4.4}$$

where the notation $\langle \cdot\,,\,\cdot \rangle_K$[2] indicates the L_2 inner product on element K.
Let us consider the projection in one coarse element K_j^H composed of n fine elements K_i^h, then eq. (4.4) reads :

$$\sum_{i=1}^{n} \sum_{a=1}^{nv_i} \mathbf{U}_{a,\,i}^h \left\langle \varphi_{a,\,i}^h,\, \varphi_{s,\,j}^H \right\rangle_{K_i^h} = \sum_{o=1}^{nv_j} \mathbf{U}_{o,\,j}^H \left\langle \varphi_{o,\,j}^H,\, \varphi_{s,\,j}^H \right\rangle_{K_j^H} \quad \forall s \in \mathbb{N}_{(1,\,nv_j)} \,,$$

where $nv_j \leq nv_i$ for all fine elements $K_i^h \in K_j^H$. This is induced by requiring that the coarse discretization is nested in the fine one. Expressing

[2] $\langle a\,(\mathbf{x}),\, b\,(\mathbf{x}) \rangle_K \overset{\text{def}}{=} \int_K a\,(\mathbf{x})\; b\,(\mathbf{x})\; \mathrm{d}\mathbf{x}$

the previous equation in a more and more compact form, we write first

$$\mathbf{U}_j^H \;=\; \sum_{i=1}^{n} \mathbf{M}_{H_j}^{-1}\, \mathbf{M}_{h_i}^{H_j}\, \mathbf{U}_i^h\,, \quad \text{where} \quad \left(\mathbf{M}_{h_i}^{H_j}\right)_{s,\,a} = \left\langle \varphi_{a,\,i}^h,\, \varphi_{s,\,j}^H \right\rangle_{K_i^h} \tag{4.5}$$

and $\mathbf{M}_{H_j}$ is the coarse element mass matrix. Then we write

$$\mathbf{U}_j^H \;=\; \widetilde{I}_h^{H_j}\, \mathbf{U}_j^h\,,$$

where

$$\begin{aligned} \widetilde{I}_h^{H_j} &= \left[\begin{array}{ccc} \dots & \mathbf{M}_{H_j}^{-1}\, \mathbf{M}_{h_i}^{H_j} & \dots \end{array}\right] \\ \mathbf{U}_j^h &= \left[\begin{array}{ccc} \dots & \left(\mathbf{U}_i^h\right)^{\mathrm{T}} & \dots \end{array}\right]^{\mathrm{T}}. \end{aligned}$$

Finally we derive the global state restriction operator

$$\widetilde{I}_h^H \;=\; \left[\begin{array}{ccc} \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widetilde{I}_h^{H_j} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots \end{array}\right] \tag{4.6}$$

which is a $m \times n^3$ block matrix such that $\mathbf{U}^H \;=\; \widetilde{I}_h^H\, \mathbf{U}^h$, where $\mathbf{U}^H$ and $\mathbf{U}^h$ are vectors of the degrees of freedom in coarse and fine spaces, respectively.

The prolongation of the coarse state $\mathbf{u}_H$, see eq. (4.3), is equivalent to find the fine state $\mathbf{u}_h$, see eq. (4.2), such that

$$\left\langle \mathbf{u}_h - \mathbf{u}_H,\, \varphi_{a,\,i}^h \right\rangle_{K_i^h} \;=\; \mathbf{0} \qquad \forall a \in \mathbb{N}_{(1,\,nv_i)} \quad \text{and} \quad \forall i \in \mathbb{N}_{(1,\,n)}\,. \tag{4.7}$$

[3] Note that each block, $\mathbf{M}_{H_j}^{-1}\, \mathbf{M}_{h_i}^{H_j}$, is a $nv_j \times nv_i$ rectangular matrix, where $nv_i \geq nv_j$. In facts we are considering transfer operators for *hp*-multigrid methods.

Handling eq. (4.7) as done for eq. (4.4), we obtain the global state prolongation operator

$$\widetilde{I}_H^h \;=\; \begin{bmatrix} \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widetilde{I}_{H_j}^h & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots \end{bmatrix} \quad \text{with} \quad \widetilde{I}_{H_j}^h = \begin{bmatrix} \vdots \\ \mathbf{M}_{h_i}^{-1}\,(\mathbf{M}_{h_i}^{H_j})^{\mathrm{T}} \\ \vdots \end{bmatrix}, \quad (4.8)$$

which is a $n \times m^4$ block matrix such that $\mathbf{U}^h \;=\; \widetilde{I}_H^h\,\mathbf{U}^H$.

4.1.2 Residual restriction operator

The residual restriction operation is more delicate than the solution one. In facts, the fine solution $\mathbf{u}_h$ is a linear combination of shape functions, see eq. (4.2), then can be restricted by means of the Galerkin projection, as shown. Conversely the fine residual $\mathbf{r}^h(\mathbf{u}_h)$ is a vector, whose components,

$$\mathbf{r}_{a,\,i}^h(\mathbf{u}_h) \;\stackrel{\text{def}}{=}\; \mathbf{B}\,(\mathbf{u}_h, \varphi_{a,\,i}^h) \qquad \forall a \in \mathbb{N}_{(1,\,nv_i)} \quad \text{and} \quad \forall i \in \mathbb{N}_{(1,\,n)}\,, \qquad (4.9)$$

are non-linear functions of $\mathbf{u}_h$. Thus, in order to restrict the residual, we can either project on the coarse-level the fine-level decomposition of $\mathbf{r}^h(\mathbf{u}_h)$ [12] or use a fine representation of coarse shape functions [8], profiting by the linearity of the second term in $\mathbf{B}\,(\,\cdot\,,\,\cdot\,)$.
Fidkowski, in [8], developed a residual restriction operator for nested spaces by using the correspondence between shape functions. In facts, since $\mathbf{V}_H$ is nested in $\mathbf{V}_h$, any $\varphi_{o,\,j}^H$ can be expressed in terms of $\varphi_{a,\,i}^h$,

$$\varphi_{o,\,j}^H \;=\; \sum_{i=1}^{n} \sum_{a=1}^{nv_i} \alpha_{o,\,j,\,a,\,i}\;\varphi_{a,\,i}^h \qquad \forall o \in \mathbb{N}_{(1,\,nv_j)} \quad \text{and} \quad \forall j \in \mathbb{N}_{(1,\,m)}\,, \qquad (4.10)$$

[4]Note that each block, $\mathbf{M}_{h_i}^{-1}\,(\mathbf{M}_{h_i}^{H_j})^{\mathrm{T}}$, is a $nv_i \times nv_j$ rectangular matrix, where $nv_i \geq nv_j$.

where n is the number of elements K_i^h in K_j^H. Then, the coarse representation of $\mathbf{r}^h(\mathbf{u}_h)$ is expressed as

$$\mathbf{r}^H_{o,\,j}(\mathbf{u}_h) \;=\; \sum_{i=1}^{n} \sum_{a=1}^{nv_i} \alpha_{o,\,j,\,a,\,i}\, \mathbf{r}^h_{a,\,i}(\mathbf{u}_h) \qquad \forall o \in \mathbb{N}_{(1,\,nv_j)} \quad \text{and} \quad \forall j \in \mathbb{N}_{(1,\,m)}\,,$$

by replacing eq. (4.10) in eq. (4.9) and using the linearity of $\mathbf{B}\,(\,\cdot\,,\,\cdot\,)$ with respect to its second term. Fidkowski[5] showed that the coefficients $\alpha_{o,\,j,\,a,\,i}$, which set up the residual restriction operator, are the terms of state prolongation transpose $(\widetilde{I}_H^h)^{\mathrm{T}}$.
Thus, the global residual restriction operator reads :

$$I_h^H \;=\; \begin{bmatrix} \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_h^{H_j} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots \end{bmatrix} \tag{4.11}$$

with

$$I_h^{H_j} \;=\; \begin{bmatrix} \;\ldots & \mathbf{M}_{h_i}^{H_j}\, \mathbf{M}_{h_i}^{-1} & \ldots\; \end{bmatrix},$$

which is a $m \times n$ block matrix such that $\mathbf{r}^H(\mathbf{u}_h) \;=\; I_h^H\; \mathbf{r}^h(\mathbf{u}_h)$, where $\mathbf{r}^H(\mathbf{u}_h)$ and $\mathbf{r}^h(\mathbf{u}_h)$ are fine residual vectors in the coarse and fine spaces, respectively.
Note that, in eq. (4.11) the residual restriction matrix is definitively the state prolongation transpose. In facts, the transpose of block matrices in eq. (4.8) verify relation :

$$\left(\mathbf{M}_{h_i}^{-1}\; (\mathbf{M}_{h_i}^{H_j})^{\mathrm{T}}\right)^{\mathrm{T}} \;=\; \mathbf{M}_{h_i}^{H_j}\; \mathbf{M}_{h_i}^{-1}\,,$$

owing to the symmetry of $\mathbf{M}_{h_i}$ and to the following properties of transpose

$$(\mathbf{A}\;\mathbf{B})^{\mathrm{T}} \;=\; \mathbf{B}^{\mathrm{T}}\; \mathbf{A}^{\mathrm{T}}\,, \qquad (\mathbf{C}^{-1})^{\mathrm{T}} \;=\; (\mathbf{C}^{\mathrm{T}})^{-1}\,,$$

[5] Refer to section 3.4.4 in [8].

where $\mathbf{C}$ is any invertible square matrix.

This approach may be generalized to non-nested spaces in the following way. Let us express the residual, associated to any element K_i^h, as :

$$\mathbf{r}_{a,\,i}^h(\mathbf{u}_h) \;=\; \int_{K_i^h} \boldsymbol{\nabla}\cdot\mathbf{f}\,(\mathbf{u}_h)\;\varphi_{a,\,i}^h\;\mathrm{d}\mathbf{x} \qquad \forall a\in\mathbb{N}_{(1,\,nv_i)}\,, \tag{4.12}$$

where $\mathbf{f}\,(\mathbf{u}_h)$ is the flux function of any PDE in the conservative form. The expansion of the divergence of the flux function in $\mathbf{V}_h$, reads :

$$\boldsymbol{\nabla}\cdot\mathbf{f}\,(\mathbf{u}_h) \;\sim\; \sum_{a=1}^{nv_i} \mathcal{R}_{\,a,\,i}\,\varphi_{a,\,i}^h \qquad \forall i\in\mathbb{N}_{(1,\,n)}\,, \tag{4.13}$$

where $\mathcal{R}$ is the vector of degrees of freedom of the aimed decomposition, whereas symbol $\sim$ is used to point out that, owing to the non-linearity of $\mathbf{f}\,(\mathbf{u}_h)$, the two sides of eq. (4.13) are not equal. Then, requiring the right-hand side in eq. (4.13) to be "weakly" equal to the left one, the following expression for $\mathcal{R}$ is derived :

$$\mathbf{M}_{h_i}^{-1}\,\mathbf{r}_i^h(\mathbf{u}_h) \;=\; \mathcal{R}_{\,i} \qquad \forall i\in\mathbb{N}_{(1,\,n)}\,, \tag{4.14}$$

where n is the number of elements in $\mathbf{V}_h$. Vector $\mathcal{R}$, which is the linear expansion of $\boldsymbol{\nabla}\cdot\mathbf{f}\,(\mathbf{u}_h)$ in $\mathbf{V}_h$, can be restricted to the coarse space by means of the global state restriction operator. Thus, restricting $\mathcal{R}$, as done for $\mathbf{U}^h$ in the previous section, and using eq. (4.14) in both $\mathbf{V}_h$ and $\mathbf{V}_H$, we obtain again operator (4.11).

This result lets drop the hypothesis of nested spaces, which was imposed by Fidkowski, furthermore it shows that the residual restriction operator is different from the state one even for linear flux functions.

4.1.3 Jacobian restriction operator

The Jacobian restriction operator is constructed in such a way that it is "compatible" with the solution / residual transfer operators.

Considering the first order Taylor expansion of the fine residual $\mathbf{r}^h(\mathbf{u}_h)$,

$$\left[\frac{\partial\ \mathbf{r}^h(\mathbf{u}_h^{(1)})}{\partial\ \mathbf{u}_h}\right]\ \Delta\,\mathbf{u}_h = \Delta\,\mathbf{r}^h(\mathbf{u}_h)\,, \tag{4.15}$$

where

$$\Delta\,\mathbf{u}_h\ =\ \mathbf{u}_h^{(2)} - \mathbf{u}_h^{(1)} \quad \text{and} \quad \Delta\,\mathbf{r}^h(\mathbf{u}_h)\ =\ \mathbf{r}^h(\mathbf{u}_h^{(2)}) - \mathbf{r}^h(\mathbf{u}_h^{(1)})\,,$$

the relation

$$I_h^H\ \left[\frac{\partial\ \mathbf{r}^h(\mathbf{u}_h^{(1)})}{\partial\ \mathbf{u}_h}\right]\ \widetilde{I}_H^h\ \ \widetilde{I}_h^H\ \Delta\,\mathbf{u}_h\ =\ I_h^H\ \Delta\,\mathbf{r}^h(\mathbf{u}_h) \tag{4.16}$$

must be verified in order to make the coarse expression (4.16) being consistent with its fine counterpart (4.15).
Then the Jacobian restriction operator reads

$$\left[\frac{\partial\ \mathbf{r}^H(\mathbf{u}_h^{(1)})}{\partial\ \mathbf{u}_H}\right]\ =\ I_h^H\ \ \left[\frac{\partial\ \mathbf{r}^h(\mathbf{u}_h^{(1)})}{\partial\ \mathbf{u}_h}\right]\ \widetilde{I}_H^h\,. \tag{4.17}$$

Note that matrix $\widetilde{I}_H^h$ is some "generalized inverse" of matrix $\widetilde{I}_h^H$. In facts, considering a fine state $\mathbf{s}_h$ which can be accurately expressed on the coarse-level, we have that $\mathbf{s}_h\ =\ \widetilde{I}_H^h\ \widetilde{I}_h^H\ \mathbf{s}_h\,.$

Finally it is important to point out that, in h-multigrid approach, restricting to the coarse-level the Jacobian block diagonal entries is not equivalent to selecting the diagonal blocks of the restricted Jacobian. Conversely, in p-multigrid approach, the two processes produce the same result. In facts, expanding eq. (4.17), any j^{th} diagonal block of the restricted Jacobian is written as :

$$\mathbf{J}_{j,\,j}^H\ =\ \sum_{i,\,p=1}^{n}\ \mathbf{M}_{h_i}^{H_j}\ \mathbf{M}_{h_i}^{-1}\ \ \mathbf{J}_{i,\,p}^h\ \ \mathbf{M}_{h_p}^{-1}\ \left(\mathbf{M}_{h_p}^{H_j}\right)^{\mathrm{T}}\,, \tag{4.18}$$

where n is the number of fine elements $K_i^h \in K_j^H$ and $\mathbf{J}_{i,\,p}^h$ are blocks of the fine Jacobian $[\partial\,\mathbf{r}^h/\partial\,\mathbf{u}_h]$. Instead, the restriction of fine-level Jacobian block diagonal entries produces a matrix $\mathbf{D}^H$, whose non-null blocks are

$$\mathbf{D}_{j,\,j}^H \;=\; \sum_{i=1}^{n} \mathbf{M}_{h_i}^{H_j}\;\mathbf{M}_{h_i}^{-1}\;\;\mathbf{J}_{i,\,i}^h\;\;\mathbf{M}_{h_i}^{-1}\;\left(\mathbf{M}_{h_i}^{H_j}\right)^{\mathrm{T}}. \tag{4.19}$$

Hence matrix $\mathbf{D}^H$ differs from $\mathbf{J}^H$ in those off-diagonal blocks, of fine-level Jacobian, which "connect" fine elements belonging to the same coarse element. The "lack of consistency" of $\mathbf{D}^H$ in respect of $\mathbf{J}^H$ does not occur in p-multigrid approach, in which the coarsening process is done by reducing the sole polynomial degree of the discretization.

The h-multigrid scheme, further presented in section 4.2, performs the first coarse level iteration using $\mathbf{D}^H$. This scheme do not show any stability problem when solving subsonic Euler equations.

4.1.4 Properties for hierarchic orthonormal shape sets

In previous sections it was introduced a set transfer operators which is well-suited for DG *hp*-multigrid approaches regardless of the shape set used in the discretization. Actually, transfer operators become much simpler by considering sets of hierarchic orthonormal shape functions. Furthermore they get a interesting property which is the "cornerstone" of adaptive *hp*-multigrid approaches for the discontinuous Galerkin finite element method.

Orthonormal sets of shape functions verify :

$$\left\langle \varphi_{a,\,i}^h,\, \varphi_{e,\,i}^h \right\rangle_{K_i^h} \;=\; \delta_{a,\,e} \quad \forall a, e \in \mathbb{N}_{(1,\,nv_i)} \quad \text{and} \quad \forall i \in \mathbb{N}_{(1,\,n)}\,, \tag{4.20}$$

on the fine-level, and

$$\left\langle \varphi_{o,\,j}^H,\, \varphi_{s,\,j}^H \right\rangle_{K_j^H} \;=\; \delta_{o,\,s} \quad \forall o, s \in \mathbb{N}_{(1,\,nv_j)} \quad \text{and} \quad \forall j \in \mathbb{N}_{(1,\,m)}\,, \tag{4.21}$$

on the coarse-level, where $\delta_{\circ,\bullet}$ is the Kronecker delta.
That is, eq. (4.20) and eq. (4.21) state that elemental mass matrices are identity matrices. Then transfer operators become :

$$\begin{array}{rcl} \widetilde{I}_h^H &=& \mathbf{T}_h^H \\ \widetilde{I}_H^h &=& \left(\mathbf{T}_h^H\right)^{\mathrm{T}} \quad , \qquad \text{where} \quad \mathbf{T}_h^H \;=\; \left[\begin{array}{ccc} \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_h^{H_j} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots \end{array}\right] , \\ I_h^H &=& \mathbf{T}_h^H \end{array} \tag{4.22}$$

with

$$\mathbf{T}_h^{H_j} \;=\; \left[\begin{array}{ccc} \dots & \mathbf{M}_{h_i}^{H_j} & \dots \end{array}\right] .$$

Actually, the choice of orthonormal shape sets reduces memory requirements, because only $\mathbf{T}_h^H$ has to be stored. Moreover, the fact that shape functions are even hierarchic makes the leading minors[6] $[\,\mathbf{M}_{h_i}^{H_j}\,]_{o,\,a}$[7] being $\mathbf{M}_{h_i}^{H_j}$ matrices for polynomial approximations of lower global degrees.

Let us define the coarse spaces V_H^α and V_H^β, where

$$V_H^k \;\stackrel{\text{def}}{=}\; \left\{ v_H \in L^2(K^H) :\; v_H \in \mathbb{P}_k(K^H) \right\} ,$$

and the fine spaces V_h^γ and V_h^ζ, where $\gamma > \zeta \geq \alpha > \beta$ and

$$V_h^k \;\stackrel{\text{def}}{=}\; \left\{ v_h \in L^2(K^h) :\; v_h \in \mathbb{P}_k(K^h) \right\} ,$$

where K^h is one fine element belonging to K^H, see figure 4.2.
The hierarchic shape set defined on V_H^β is a subset of that defined on V_H^α,

[6] Given a $n \times m$ matrix $\mathbf{A}$, the leading minor $[\,\mathbf{A}\,]_{i,\,j}$ indicates the upper-left part of $\mathbf{A}$. That is the $i \times j$ matrix whose rows are the first i rows of $\mathbf{A}$ and whose columns are the first j columns of $\mathbf{A}$.

[7] Note that o indicates the o^{th} shape function of coarse element K_j^H and a indicates the a^{th} shape function of fine element K_i^h. Thus, in order to nest the coarse space in the fine space, $o \leq a$.

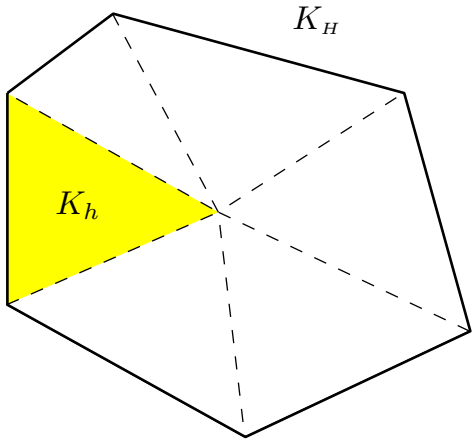


Figure 4.2: Regions for spaces V_H^α, V_H^β (*white*) and V_h^γ, V_h^ζ (*yellow*)

see section 2.6. That is, the first $nv_\beta = (\beta+1)(\beta+2)/2$ shape functions in V_H^α are the shape set in V_H^β and the first $nv_\zeta = (\zeta+1)(\zeta+2)/2$ shape functions in V_h^γ are the shape set in V_h^ζ. Thus, being $\mathbf{M}_\gamma^\alpha$ the $nv_\alpha \times nv_\gamma$ rectangular matrix, defined in eq. (4.5) , which transfers state and residual from V_h^γ to V_H^α, the following relations are set :

$$\mathbf{M}_\zeta^\alpha \;=\; [\mathbf{M}_\gamma^\alpha]_{nv_\alpha,\, nv_\zeta} \quad \text{and} \quad \mathbf{M}_\zeta^\beta \;=\; [\mathbf{M}_\gamma^\alpha]_{nv_\beta,\, nv_\zeta}\,, \tag{4.23}$$

where $\mathbf{M}_\zeta^\alpha$ and $\mathbf{M}_\zeta^\beta$ transfer state and residual from V_h^ζ to V_H^α and from V_h^ζ to V_H^β, respectively.
In order to point out how property (4.23) is the "cornerstone" of a *p*-adaptive *hp*-multigrid approach, let us make the following example. We consider a coarse element c, decomposed in two fine elements f and d, see figure 4.3, and the operator $\mathbf{T}_h^c$, which transfers the $\mathbb{P}_2$ approximation on the fine triangulation to the $\mathbb{P}_2$ approximation on the coarse element c and

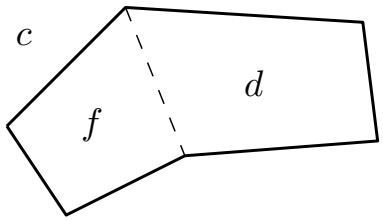


Figure 4.3: Decomposition of element c

which reads :

$$\mathbf{T}_h^c \;=\; \left[\begin{array}{cc} \mathbf{M}_f^c & \mathbf{M}_d^c \end{array}\right]. \tag{4.24}$$

Then we start to compute by means of the $\mathbb{P}_1$ polynomial approximation, using the operator

$$\mathbf{T_1}_h^c \;=\; \left[\begin{array}{cc} [\mathbf{M}_f^c]_{3,3} & [\mathbf{M}_d^c]_{3,3} \end{array}\right],$$

which is a subset of $\mathbf{T}_h^c$. At each iteration, the residuals on $\mathbb{P}_1(f)$ and $\mathbb{P}_1(d)$ are compared to those on $\mathbb{P}_2(f)$ and $\mathbb{P}_2(d)$[8] by means of some adaptive criterion. When it is necessary, the degree of the approximation on f is increased and the operator

$$\mathbf{T_2}_h^c \;=\; \left[\begin{array}{cc} [\mathbf{M}_f^c]_{3,6} & [\mathbf{M}_d^c]_{3,3} \end{array}\right]$$

is used. Furthermore, if it was necessary to increase even the degree of the approximation on d, we would increase the degree of the approximation on c[9] too and use the operator in eq. (4.24).
Note that, in the whole process, it was necessary to store only matrix $\mathbf{T}_h^c$.

[8] The state prolongation from $\mathbb{P}_1$ to $\mathbb{P}_2$ does not need any further computation, because, as shown in [8], [12],[13] and [14], $\widetilde{I}_1^2$ is the identity matrix with 3 zero rows appended, when using hierarchic orthonormal shape functions.

[9] In coarse elements we choose the highest polynomial approximation which assures that the coarse discretization is nested in the fine one.

4.1.5 Properties for bases defined in reference spaces

As done at the end of section 2.5, it is important to compare previous transfer operators with those arising form the "classic" reference space discretization. As a matter of fact, if both the state and the residual are defined in the reference space, the same is done for transfer operators. Then, residual and state transfer operators are required just for canonical element shapes, thus strongly reducing memory requirements. However, we remember that actual orthonormal sets of shape functions, defined in the real space, make transfer operators, for p-multigrid methods, being identity matrices with some zero rows or columns appended. Therefore they do not need to be stored.

4.2 FAS and two-level multigrid

To solve the nonlinear system of equations (2.67), the *Full Approximation Storage* (FAS) scheme was chosen as the multigrid method. Much of the following description is adapted from Fidkowski [9] and Mavriplis [13]. Consider the discretized system of equations given by

$$\mathbf{R}^h(\mathbf{U}^h) \; + \; \mathbf{P}^h \; = \; \mathbf{0},$$

where $\mathbf{U}^h$ is the discrete solution vector for the fine grid level, $\mathbf{R}^h(\mathbf{U}^h)$ is the associated non-linear system and $\mathbf{P}^h$ is the forcing term, which is null for the finest level problem[10]. Let $\mathbf{U}^h_n$ be an approximation to the solution vector and define the discrete residual, $\mathbf{r}^h(\mathbf{U}^h_n)$, by

$$\mathbf{r}^h(\mathbf{U}^h_n) = \mathbf{R}^h(\mathbf{U}^h_n) \; + \; \mathbf{P}^h \, .$$

In a basic two-level multigrid method, the exact solution on a coarse level is used to correct the solution on the fine level. The two level correction scheme is given as follows :

- Iterate on the fine level with the locally-implicit multistage scheme presented in section 2.10
- Obtain a new state approximation $\mathbf{U}^h_{n+1}$
- Compute both the new residual $\mathbf{r}^h(\mathbf{U}^h_{n+1})$ and the new jacobian block diagonal entries $\mathbf{D}^h(\mathbf{U}^h_{n+1})$
- Restrict both the state and the residual to the coarse level :

$$\begin{aligned} \mathbf{U}^H_0 &= \widetilde{I}^H_h \; \mathbf{U}^h_{n+1} \\ \mathbf{R}^H_0 &= I^H_h \; \mathbf{r}^h(\mathbf{U}^h_{n+1}) \, , \end{aligned}$$

  by means of restriction operators defined in sections 4.1.1 and 4.1.2

[10] Note that even if we are interested in the stationary solution, the time dependent term $\mathbf{M}_h \; d\,\mathbf{U}^h/dt$ is used to alleviate transients during the solution process

- Compute restricted state residual and coarse level forcing term :

$$\mathbf{P}^H = \mathbf{R}_0^H \; - \; \mathbf{R}^H(\mathbf{U}_0^H)\,.$$

- Restrict jacobian block diagonal entries, $\mathbf{D}^h(\mathbf{U}_{n+1}^h)$, in the coarse level

$$\mathbf{D}_0^H \; = \; I_h^H \; \mathbf{D}^h(\mathbf{U}_{n+1}^h) \; \widetilde{I}_H^h$$

- Solve the coarse level problem

$$\mathbf{R}^H(\mathbf{U}^H) \; + \; \mathbf{P}^H \; = \; \mathbf{0}\,, \tag{4.25}$$

  starting from the coarse state $\mathbf{U}_0^H$. The solution method consists in k iterations with the locally-implicit multistage residual smoother presented in section 2.10, using the restricted jacobian for the first iteration.

- Compute the coarse grid error

$$\mathbf{E}^H = \mathbf{U}_k^H - \mathbf{U}_0^H\,,$$

  and correct the fine grid approximation

$$\mathbf{U}_{n+1}^h \leftarrow \mathbf{U}_{n+1}^h + \widetilde{I}_H^h \; \mathbf{E}^H$$

Note that, the FAS coarse level equation (4.25) can be written as

$$\mathbf{R}^H(\mathbf{U}^H) \; + \; I_h^H \; \mathbf{r}^h(\mathbf{U}_{n+1}^h) \; - \; \mathbf{R}^H(\mathbf{U}_0^H) \; = \; \mathbf{0}\,, \tag{4.26}$$

so that, if the fine grid problem converged, $\mathbf{r}^h(\mathbf{U}_{n+1}^h) \; = \; \mathbf{0}$ and eq. (4.26) is identically verified, since $\mathbf{U}^H \; = \; \mathbf{U}_0^H$.
The restricted residual $I_h^H \; \mathbf{r}^h(\mathbf{U}_{n+1}^h)$ "controls" coarse level iterations. In facts, considering the simple non-linear element Jacobi scheme, the first two iterations on the coarse level read :

$$\begin{aligned}
\mathbf{U}_1^H \; &= \; \left(\frac{\mathbf{M}_H}{\Delta t} \; + \; \mathbf{D}_0^H \right)^{-1} (\mathbf{R}^H(\mathbf{U}_0^H) \; + \; \mathbf{P}^H) \; + \; \mathbf{U}_0^H\,, \qquad (4.27)\\
\mathbf{U}_2^H \; &= \; \left(\frac{\mathbf{M}_H}{\Delta t} \; + \; \mathbf{D}^H(\mathbf{U}_1^H) \right)^{-1} (\mathbf{R}^H(\mathbf{U}_1^H) \; + \; \mathbf{P}^H) \; + \; \mathbf{U}_1^H\,.
\end{aligned}$$

Note that if $\mathbf{r}^h(\mathbf{U}^h_{n+1})$ vanishes so does its restriction on the coarse level,

$$\mathbf{R}^H(\mathbf{U}^H_0) \; + \; \mathbf{P}^H \; = \; \mathbf{R}^H_0 \to \mathbf{0}\,.$$

Then, supposing that (4.27) is a stable iteration, the state change vanishes

$$\mathbf{U}^H_1 \; - \; \mathbf{U}^H_0 \to \mathbf{0}\,,$$

and the next residual "gets close" to the restricted residual

$$\mathbf{R}^H(\mathbf{U}^H_1) \; - \; \mathbf{R}^H(\mathbf{U}^H_0) \; + \; \mathbf{R}^H_0 \to \mathbf{R}^H_0\,,$$

which vanishes.

In order to obtain a stable two level h-multigrid scheme, it is necessary to use the restricted jacobian $\mathbf{D}^H_0$, instead of the jacobian of the restricted state $\mathbf{D}^H(\mathbf{U}^H_0)$, for the first coarse iteration (4.27). This consideration concerns exclusively the h-multigrid method derived so far. Conversely, pure p-multigrid methods seem stable even if the jacobian of the restricted state $\mathbf{D}^H(\mathbf{U}^H_0)$ is used.

4.3 V-cycles

To make multigrid practical, the basis two level correction scheme could be extended to a V-cycle and to full multigrid (FMG). In a V-cycle, a sequence of one or more coarse levels is used to correct the solution on the fine level. Descending from the finest level to the coarsest, a certain number of pre-smoothing iterations can be performed on each level before the problem is restricted to the next coarser level. On the coarsest level, the problem is either solved directly or smoothed a relatively large number of times, as presented in [9] and in [12]. Ascending back to the finest level, some post-smoothing iterations can be performed on each level. Each such V-cycle constitutes a multigrid iteration.

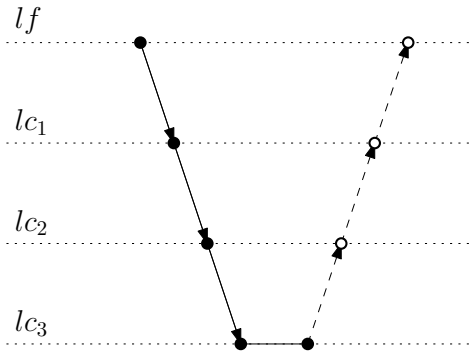


Figure 4.4: Example of a V-cycle multigrid iteration with three coarse levels. $\longrightarrow$ refers to residual, state and jacobian restriction, $\bullet$ represents one smoothing iteration and $\dashrightarrow\!\circ$ is the correction

Some representative tests, in section 4.4.1, show that, if coarse levels are constructed by coarsening the triangulation at constant polynomial approximation, V-cycle performance does not depend very much on pre-smoothing iteration number. Conversely, if coarse levels are constructed by reducing the polynomial approximation on the same triangulation, V-cycle performance improves by increasing the number of pre/post-smoothing iterations. In terms of CPU time, p-multigrid V-cycles attain optimal convergence rates for a given number of pre/post-smoothing iterations, nevertheless this limit occurs for a quite large number of iterations, see [39]. As a matter of fact, in section 2.10 we observed that the number of iterations to converge of the locally-implicit multistage scheme is independent of the polynomial degree. The error dumping speed depends on some grid characteristic dimension, which is independent of the polynomial approximation. Then, in order to overcome this limit in "pure" p-multigrid schemes, it is necessary to increase the number of smoothing iterations on coarse levels, where they are less expensive. Conversely, in h-multigrid schemes, the dumping speed limit is overcome by coarsening elements in coarse levels and increasing the number of iterations is not necessary.

Using plain V-cycles to obtain a high-order solution requires to starting the smoothing iterations on the highest order approximation. As this level contains the largest number of degrees of freedom, smoothing on it is the most expensive. An alternative is to obtain an approximation to the solution using the coarser levels before smoothing on the finest level. This is the premise behind FMG in which V-cycles on successively finer levels are used to approximate the solution on the finest level. By the time the solution is prolongated to the finest level, it is usually a close approximation to the final solution with the exception of certain high frequency errors that can be smoothed efficiently on that level.
A decision that has to be made in the FMG algorithm is when to start iterating on the next finer level. Converging the solution fully on each level is not practical because the discretization error on the coarser level is usually well above machine zero. We can perform a constant number of V-cycles on each level, [13] and [14], or we can prolongate the state when a residual-based criterion is met, [8].

Both FMG approaches were implemented and tested, but they are less efficient than the "classic" V-cycle. In h-multigrid at constant polynomial degree the discretization error on coarse grids is generally high, then coarse level solution is not a good initial state for the finer level. On the other hand, h-multigrid show a very efficient V-cycle, if combined with the quasi-implicit multistage scheme. Thus, we suggest a FMG in which the solution is discretized by successively higher polynomial degrees on the same triangulation, as for p-multigrid approaches, but the convergence is accelerated by means of the h-multigrid V-cycle. The polynomial degree should be increased in each fine element when a local residual-based criterion is met. At the end of a V-cycle, the current residual and its L_1 norm, $\|\mathbf{r}^h(\mathbf{U}^k)\|_{L_1}$, are known in each fine grid element. The solution vector is prolongated to $k+1$ level[11] and the residual is calculated along with its

[11] The state prolongation from $\mathbb{P}_k$ to $\mathbb{P}_{k+1}$ does not need any further computation, because $\widetilde{I}_k^{k+1}$ is the identity matrix with $k+1$ zero rows appended, when using hierarchic

L_1 norm $\|\mathbf{r}^h(\mathbf{U}^{k+1})\|_{L_1}$. The polynomial degree is increased when the condition $\|\mathbf{r}^h(\mathbf{U}^k)\|_{L_1} < \eta_r \|\mathbf{r}^h(\mathbf{U}^{k+1})\|_{L_1}$ is reached, where $\eta_r \in (0,1)$ could depend on some global convergence parameter. Conversely, the polynomial degree of the discretization is kept constant in those elements in which this condition is not met. Furthermore, in V-cycle coarse level elements we choose the highest polynomial degree which assures that the coarse level discretization is nested in the finer one. As a matter of fact, increasing the degree of the discretization is beneficial to the convergence rate, see section 4.4.3, however reducing the global number of degrees of freedom is necessary for computational efficiency.

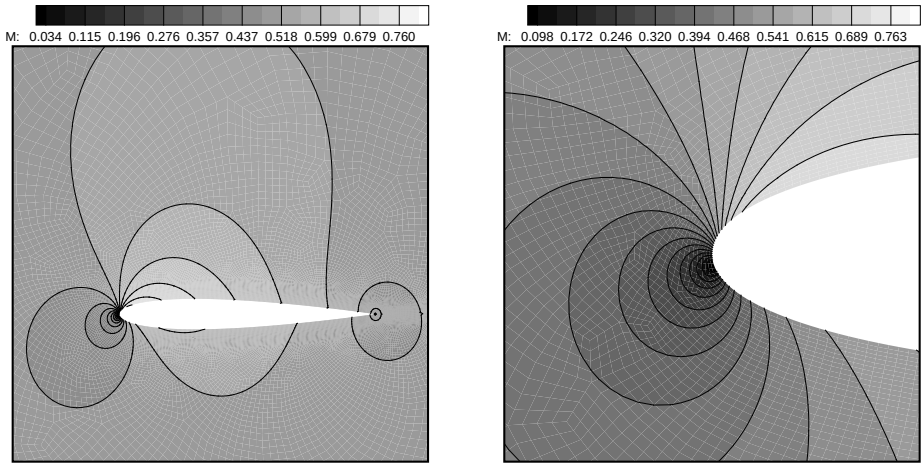
4.4 Preliminary tests

In order to point out some of the most considerable properties of the h-multigrid at constant polynomial degree approach, several tests were conducted on the same problem. The solution of compressible Euler equations around the NACA 0012 airfoil with 2^o of incidence and $\mathrm{M}_\infty = 0,5$ was discretized on a hybrid unstructured mesh of 1124 elements.
The boundary conditions were set as done in section 3.6 except for the inflow angle which was set at 2^o. Furthermore, in all tests, integration rules were reduced by means of the criterion presented in section 2.4, imposing $tol = 5\mathrm{e}{-6}$ for the finest level grid and $tol = 1\mathrm{e}{-2}$ for all the coarse grids. Hereinafter are presented preliminary yet meaningful tests, which should be followed by a precise two-level convergence analysis.

4.4.1 Pre-smoothing iterations

Let us consider the effect of pre-smoothing iteration number on the convergence rate of h-multigrid. All the computions start from the uniform initialization of the flow field with the free-stream flow condition. Then, as introduced in section 4.3, the V-cycle at constant $\mathbb{P}_4$ approximation is used

orthonormal shape functions.



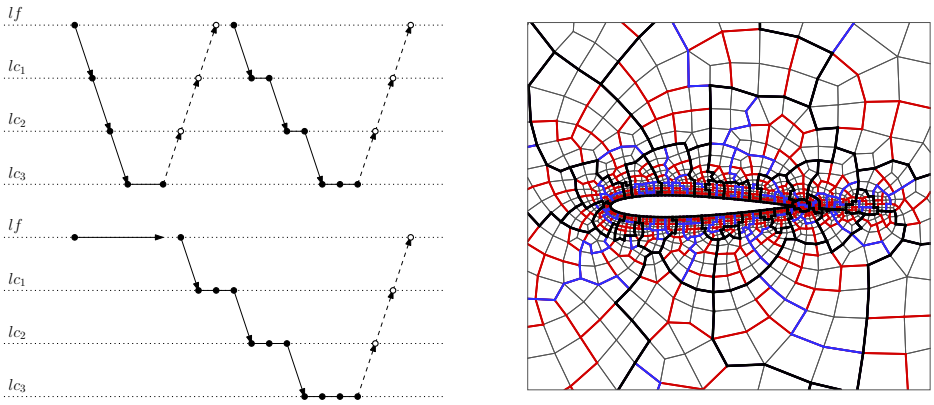
(a) $\mathbb{P}_4$ Mach iso-contours (b) $\mathbb{P}_4$ Mach iso-contours

Figure 4.5: $\mathbb{P}_4$ solution approximation on the considered 1124 element grid

without any FMG approach.

In figure 4.6(a) are depicted four V-cycle schemes, which differ from one other in the number of pre-smoothing iterations on each level. Moreover, in all schemes, the coarsest level is characterized by an additional smoothing iteration with respect to finer levels. Note that basic V-cycles, figures 4.4 and 4.9, perform at least two interations on the coarsest level. As a matter of fact, the first iteration on any coarse level is performed by using the restricted jacobian, see section 4.2, which is computed on the finer level. Thus, on the coarsest level, the second iteration has a negligible computational cost, nevertheless it has a beneficial effect to convergence, especially in first iterations. Finally, the case without pre-smoothing iterations coincides with using the locally-implicit multistage scheme directly on the finest grid.

In figure 4.6(b) are represented the three coarse grids together with the finest level grid. Each coarse element is obtained by agglomerating 5 elements of the finer level grid using the algorithm in [17]. As shown in figure 4.6(b), coarse level triangulations are composed by polygons which can be very irregular. Thus, in order to obtain a coarse level jacobian which is "compatible" with the restricted residual, it is necessary to construct it by means of the consistency relation in eq. (4.17). That makes the first iteration in eq. (4.27) being stable, as required by a correct "fine residual control".

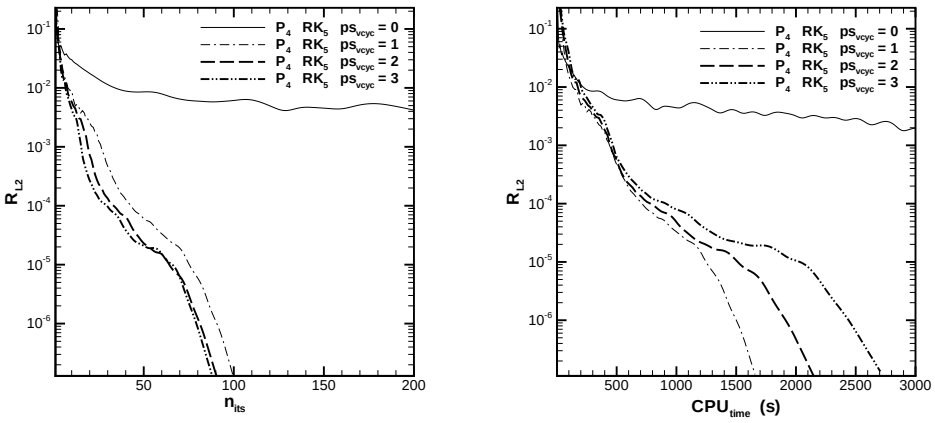


(a) V-cycle schemes

(b) Superimposition of 4 grid levels

Figure 4.6: *Left* : representation of V-cycle scheme for 0, 1, 2 and 3 pre-smoothing iterations. *Right* : superimposition of grid levels lf (*fine gray*), lc_1 (*red*), lc_2 (*blue*) and lc_3 (*black*)

Increasing the number of pre-smoothing iterations has no significant effect on the convergence rate, thus it is detrimental to computational efficiency owing to the CPU cost of the additional iterations. In figure 4.7 it is also represented the convergence curve when no V-cycle is used. In this case, the h-multigrid method converges approximately in one twentieth of the



(a) Iteration number (b) CPU time

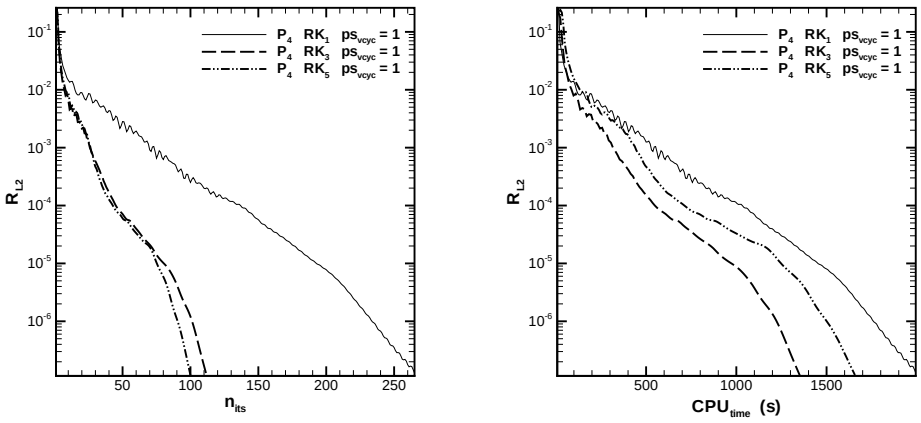
Figure 4.7: Effect of pre-smoothing on the convergence rate in terms iterations and CPU time

iterations needed to converge with the simple locally-implicit multistage scheme. This result will be improved in the next sections, but it is still far from its optimum which requires FMG and "large meshes".

By the way, this simple test points out an important difference with the p-multigrid approach, which is significantly improved by the use of several pre/post-smoothing iterations on coarse multigrid levels. For the interpretation refer to section 4.3.

4.4.2 Smoother number of stages

In previous section we decided to use a V-cycle which makes use of one smoothing iteration on each coarse level and two in the coarsest one, figure 4.6(a) top-left. In this section it is investigated the effect of locally-implicit multistage schemes on the h-multigrid.
Figure 4.8(a) shows that the number of iterations to converge reduces when



(a) Iteration number (b) CPU time

Figure 4.8: Effect of number of stages on the convergence rate in terms iterations and CPU time

increasing the number of stages in the scheme. As remarked in section 2.10, updating more than once during an iteration enhances the "communication" between neibouring elements. Anyway, in terms of computational cost, the scheme with five stages is less efficient than that with three stages.

The locally-implicit multistage scheme is still a work in progress, thus those are preliminary results.

4.4.3 Polynomial degree and macro-element dimension

Aim of this section is evaluating the effect of both polynomial degree and macro-element dimension on h-multigrid efficiency. Then we compare convergence rates of a three level V-cycle, figure 4.9, when changing the degree of the polynomial discretization and the number of fine elements making up a coarse element.
In this test convergence rates are compared in terms of number of iterations.

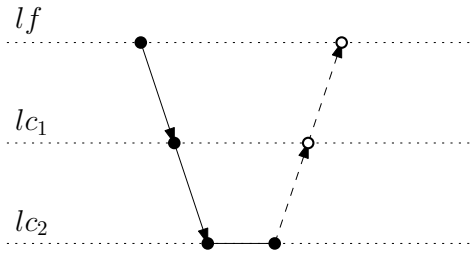
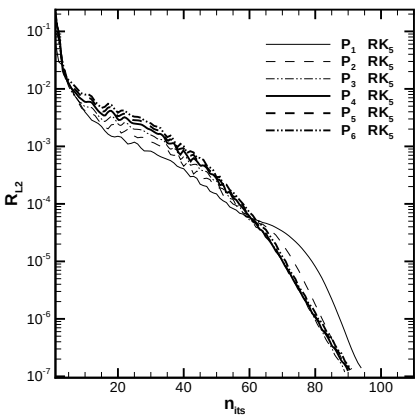
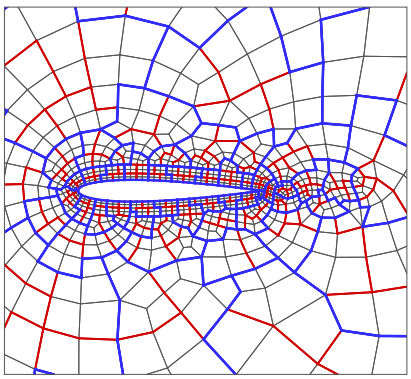


Figure 4.9: V-cycle multigrid iteration with two coarse levels

As a matter of fact, increasing the degree of the polynomial approximation is equivalent to rise the computational cost of each iteration, then it does not make sense to compare convergence rates in terms of CPU time.



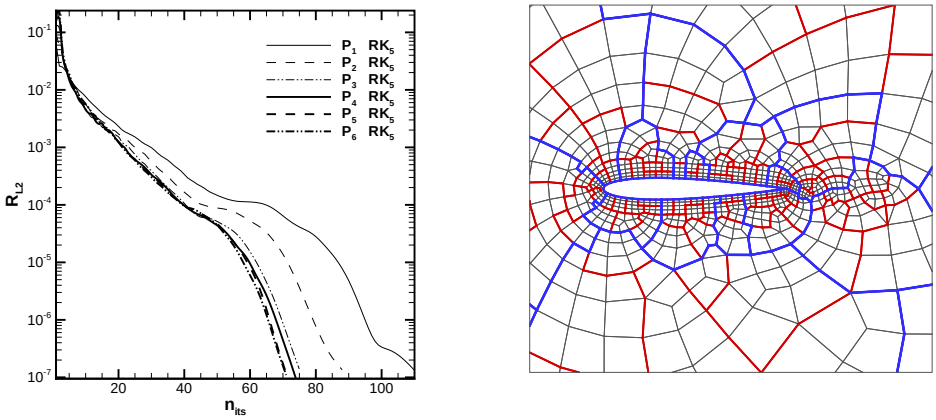
(a) Iteration number



(b) 5×5 level grids

Figure 4.10: *Left* : convergence rates for different polynomial approximations. *Right* : superimposition of grid levels lf (*fine gray*), lc_1 (*red*), lc_2 (*blue*)

In figure 4.10 is depicted the effect of polynomial degree on the convergence rate of the V-cycle, in which coarse level elements are constructed by agglomerating five elements of the finer grid. In this case, the order of the polynomial approximation has a negligible effect on convergence rate and the h-multigrid converges in approximately 90 iterations.

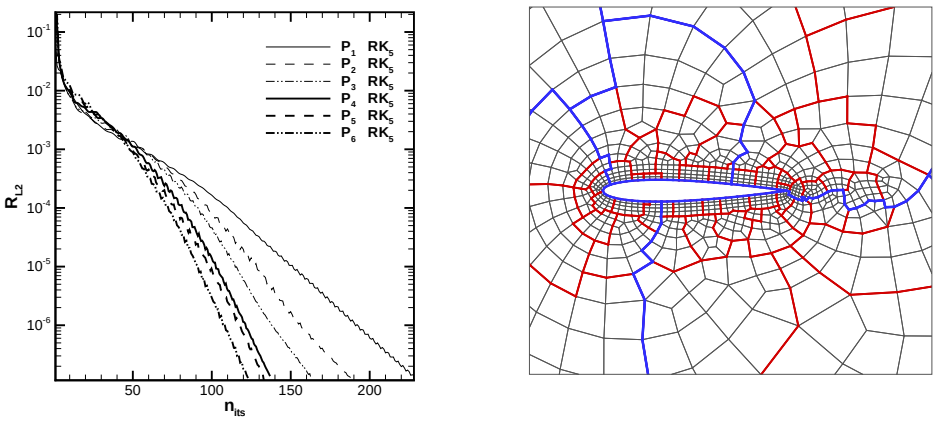


(a) Iteration number

(b) 10×10 level grids

Figure 4.11: *Left* : convergence rates for different polynomial approximations. *Right* : superimposition of grid levels lf (*fine gray*), lc_1 (*red*), lc_2 (*blue*)

Let us increase the number of fine elements composing a macro-element from 5 to 10, constructing two coarse levels, figure 4.11(b), whose elements are larger than in the previous case. Then, see figure 4.11(a), we observe that the order of the polynomial approximation has a beneficial effect on h-multigrid convergence rate. In facts the number of iterations to converge decreases of 21%, 17% and 7% when increasing the approximation degree from $\mathbb{P}_1$ to $\mathbb{P}_2$, from $\mathbb{P}_2$ to $\mathbb{P}_3$ and from $\mathbb{P}_3$ to $\mathbb{P}_{5,6}$, respectively. Finally, the least number of iterations to converge equals 70, which is 20 iterations less than in the previous case.



(a) Iteration number

(b) 20×20 level grids

Figure 4.12: *Left* : convergence rates for different polynomial approximations. *Right* : superimposition of grid levels lf (*fine gray*), lc_1 (*red*), lc_2 (*blue*)

The beneficial effect of polynomial approximation degree increases even more if macro-elements are made up by 20 elements of the finer level, see figure 4.12(b). The number of iterations to converge decreases of 17%, 16%, 16% and 13% when increasing the approximation degree from $\mathbb{P}_1$ to $\mathbb{P}_2$, from $\mathbb{P}_2$ to $\mathbb{P}_3$, from $\mathbb{P}_3$ to $\mathbb{P}_4$ and from $\mathbb{P}_4$ to $\mathbb{P}_6$, respectively. Even though the convergence rate is enhanced by the polynomial degree of the discretization, the least number of iterations equals 120, which is much higher than in the previous case.

This test points out some interesting properties of h-multigrid at constant polynomial approximation, which are listed below :

- Increasing the degree of the polynomial approximation has always a beneficial/negligible effect on the convergence rate of the V-cycle

- Enlarging coarse level elements enhances the beneficial effect of polynomial degree on the convergence rate. As a matter of fact, increasing the number of fine elements making up a coarse element is equivalent to broaden the error frequency band which has to be transferred and dumped on the coarse level. Then rising the polynomial degree in coarse elements enable to "capture" a larger error frequency band.

- The best convergence rates are showed by the V-cycle in which coarse level elements are constructed by agglomerating ten elements of the finer grid. Indeed, among the three considered coarse element "dimensions", the 10×10 configuration (figure 4.11(b)) distributes the error frequency band evenly on the three levels. To further justify the previous statement for the 10×10 configuration, we observe that in the finest level there are 1124 elements, in lc_1 level there are 121 macro-elements composed by 10 fine elements and in lc_2 level there are 13 macro-elements composed by 10 sub-elements

Thus we stated that distributing evenly the coarse level element dimensions gives the best convergence rates, provided that the polynomial degree of the approximation is high enough.

4.4.4 Number of "evenly distributed" multigrid levels

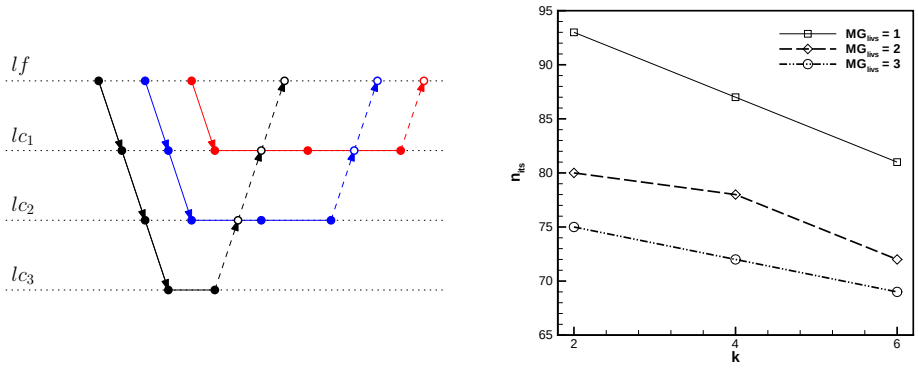
This last test deals with the effect of the number of levels of h-multigrid V-cycle scheme. In order to plan it as fair as possible, the global number of fine elements was evenly distributed among all coarse levels. That is, each macro-element is made up by n_{se} elements of the finer level, provided that

$$n_{se} \sim (n_e)^{1/n_{levs}}, \quad [12] \tag{4.28}$$

where n_e is the finest grid element number and n_{levs} is the global number of levels, which is the number of coarse levels plus one. Furthermore the

[12] Note that n_{se} is generally different from $(n_e)^{1/n_{levs}}$. Thus $\sim$ refers to the "closest integer" to the right hand expression in eq. (4.28)

number of smoothing iterations on the coarsest level was increased in V-cycles with less levels in order to make comparable the computational cost of the three considered cases, see figure 4.13(a).



(a) V-cycle schemes

(b) Iteration number

Figure 4.13: *Left* : representation of V-cycle scheme with 1 (*red*), 2 (*blue*) and 3 (*black*) coarse levels. *Right* : Number of iterations to converge for three V-cycle schemes and three polynomial degrees

The graph in figure 4.13(b) shows that the beneficial effect of polynomial degree increase is enhanced by the coarse element dimension, as observed in previous section. Moreover rising either the number of levels, which is equivalent to reduce macro-element dimensions, or the degree of polynomial approximations improves the convergence rate of the h-multigrid. This behaviour suggests that enabling coarse levels to "capture" larger and lager error frequency bands is more and more beneficial to convergence rates. As a matter of fact in h-multigrid the restriction operation transfers to the coarser level frequencies which are contained in the jumps of the state across finer element interfaces. Jumps include a frequency band which is theoretically infinite. Considering the Fourier transform of any one-

dimensional jump,

$$\mathcal{F}\{\, j_{a,\,b}(x)\,\} \;=\; \delta(f)\,\frac{b+a}{2} \;-\; i\,\frac{b-a}{2\pi f}\,, \tag{4.29}$$

we observe that, although signal energy decreases for high frequencies, a good approximation of the jump requires high degree polynomials.

4.5 Concluding remarks and improvements

We presented some preliminary yet significant tests. Although they should be followed by a more exact theoretical analysis, they point out some interesting properties of the developed h-multigrid method.

First let us summarize two main differences between p-multigrid and h-multigrid methods :

- The number of iterations to converge of the locally-implicit multistage scheme is independent of the polynomial degree, see section 2.10. This behaviour suggests that error dumping speed depends on some grid characteristic dimension, which is independent of the polynomial approximation. In order to "overcome this limit" p-multigrid schemes require to increase the number of smoothing iterations on coarse levels, where they are less expensive. Conversely, in h-multigrid schemes, the grid characteristic dimension widens out in coarse levels, then rising the number of smoothing iterations is not necessary, see section 4.4.1

- The second main difference between p- and h-multigrid methods concerns the meaning of the restriction operation. In p-multigrid solution restriction is equivalent to "cutting" the highest degree modes of the polynomial approximation. Conversely in h-multigrid it is not possible to evaluate which part of error frequency band is discarded by the restriction operation. As a matter of fact the restriction operation transfers to the coarser level a part of the frequency band contained

in the jumps across finer element interfaces. Then, in order to "capture" the optimum amount of information from fine levels, it could be interesting to construct coarse spaces which are not nested in the fine ones.[13] Although transfer operators could be defined for non-nested spaces [12], the implications of their use will be analysed further

Then it is interesting to list the main observations which can be derived by the previous tests :

- Increasing the number of smoothing iterations does not improve convergence rates of h-multigrid, refer to section 4.4.1
- Among tested locally-implicit multistage schemes, the best performances in terms of computational efficiency are showed by the three step scheme. By the way that smoother needs further investigations concerning α_k coefficients, see section 2.10
- Distributing evenly the coarse level element dimensions gives the best convergence rates, provided that the polynomial degree is high enough. Presented in section 4.4.3
- Increasing both the number of multigrid evenly distributed levels and the polynomial degree of the discretization is beneficial to convergence rates. More details and observations are reported in sections 4.4.3 and 4.4.4

Finally we want to spend some words on two possible improvements of the h-multigrid presented so far.

As noticed in section 4.3, the full multigrid approach which uses h-multigrid coarse grids to initialize the fine level solution is less efficient than the "classic" V-cycle. Indeed, the discretization error on any h-multigrid coarse grid is generally much higher than that on the finer one. This feature is related to the reduced number of state jumps of any level discretization in respect

[13] It is important to remark that using non-nested spaces does make sense just in h-multigrid

of the finer level one. Then any h-multigrid coarse level solution is not a proper initial state for the finer level.
By the way, an efficient scheme needs some progressive initialization mechanism, in order to minimize computational cost of initial iterations. The most natural solution lies on progressively increasing the polynomial degree of the discretization on the finest level by means of adaptive methods and accelerating convergence by means of the h-multigrid V-cycle. The whole process does not need any re-computation of transfer operators, see section 4.1.4, and the polynomial degree of coarse approximations is chosen with regard of finest level discretization. Furthermore, in section 4.3, it was sketched a residual based adaptive criterion useful for the described process. Anyway the use of the adjoint problem would be preferable in terms of computational efficiency.

As remarked in section 4.4.3, rising the polynomial degree of the discretization is always beneficial to h-multigrid convergence rate. Indeed the jumps across element interfaces on any level approximate state changes which need high polynomial degree representation on the coarser level. If we used coarse spaces which are nested in fine ones, the polynomial degree in any coarse element would be limited to the minimum polynomial degree among all fine elements which compose it. Conversely, if coarse spaces were not required to be nested in the fine ones, the polynomial degree could be risen freely in any element on any level. Transfer operators can be defined for non-nested spaces by means of the Galerkin projection, sections 4.1.1 and 4.1.2, thus the two-level correction scheme does not need any specific treatment in this case, see section 4.2.

By the way, an important remark arises. When using nested spaces some information is lost during the restriction process, but the coarse grid error is exactly represented in the fine space. Conversely the use of non-nested spaces implies leakage of information after the prolongation process too, because the coarse grid error can not be exactly represented in the fine space. This observation suggests that any coarse element polynomial degree can be risen until a prolongation error criterion is met.

4.5.1 Prolongation error in non-nested spaces

Aim of this section is to recall some of the concepts presented in section 4.1 in order to outline a simple way to evaluate the leakage of information occuring during the prolongation process in non-nested spaces. Once the prolongation error is defined for each coarse shape function, it could be bounded by a user-defined tolerance, as done for integration rules in section 2.4.

Let us define two fine approximation spaces V_h^α and V_h^β, where $\alpha > \beta$ and

$$V_h^k \stackrel{\text{def}}{=} \left\{ v_h \in L^2(K^H) : \; v_h|_{K_i^h} \in \mathbb{P}_k(K_i^h) \; \forall K_i^h \in K^H \right\},$$

and one coarse space

$$V_H^\alpha \stackrel{\text{def}}{=} \left\{ v_H \in L^2(K^H) : \; v_H \in \mathbb{P}_\alpha(K^H) \right\}.$$

Hence, V_H^α is nested in V_h^α, but it is not nested in V_h^β.

Then, consider the following hierarchic orthonormal shape sets, $\{ \varphi_i^h \}_{i=1}^{f_\alpha}$, basis of V_h^α, its sub-set $\{ \varphi_i^h \}_{i=1}^{f_\beta}$, basis of V_h^β, and $\{ \varphi_i^H \}_{i=1}^{c_\alpha}$, basis of V_H^α.[14]

Any φ_i^H can be exactly represented on V_h^α as :

$$\varphi_i^H = \sum_{j=0}^{f_\alpha} \left(\widetilde{I}_H^h \right)_{j,\,i} \varphi_j^h \quad \forall i \in \mathbb{N}_{(1,\,c_\alpha)}, \tag{4.30}$$

where $\widetilde{I}_H^h$ is the prolongation operator which transfers the solution from V_H^α to V_h^α. While, the approximated representation $\widetilde{\varphi}_i^H$ of φ_i^H on V_h^β reads :

$$\varphi_i^H \sim \widetilde{\varphi}_i^H = \sum_{j=0}^{f_\beta} \left(\widetilde{I}_H^h \right)_{j,\,i} \varphi_j^h \quad \forall i \in \mathbb{N}_{(1,\,c_\alpha)}, \tag{4.31}$$

[14] The integers f_α, f_β and c_α are the global numbers of degrees of freedom in V_h^α, V_h^β and V_H^α, respectively.

because, owing to hierarchic orthonormal "nature" of bases, the prolongation operator which transfers the state from V_H^α to V_h^β, is a sub-set of $\widetilde{I}_H^h$, see section 4.1.4. Then the L_2 norm of the prolongation error for φ_i^H reads :

$$E(\varphi_i^H)_{V_h^\beta} \;=\; \int_{K^H} (\varphi_i^H - \widetilde{\varphi}_i^H)^2 \;\mathrm{d}\mathbf{x} \;=\; \sum_{j=f_\beta+1}^{f_\alpha} \left(\widetilde{I}_H^h\right)_{j,\,i}^2 , \tag{4.32}$$

where the orthonormality property of bases was used.

Notice that $E(\varphi_i^H)_{V_h^\beta}$ depends on the sole $\left\{\left(\widetilde{I}_H^h\right)_{j,\,i}\right\}_{j=f_\beta+1}^{f_\alpha}$ coefficients. This result points out, once more, the crucial role of transfer operators in the study of *hp*-multigrid methods.

Appendix A

Gaussian integration rules

This appendix aims to present concepts and implementation details behind a C++ code, which computes optimal and slightly optimal quadrature formulas for classic reference domains.

A.1 Orthogonal polynomials

Orthogonal polynomials are classes of polynomials $\{p_n(x)\}$ defined over a range $[a, b]$ that obey the orthogonality relation :

$$\int_a^b w(x)\, p_n(x)\, p_m(x)\; dx = \delta_{m,\,n}\, c_n \tag{A.1}$$

where $w(x)$ is a weighting function and $\delta_{m,\,n}$ is the Kronecker delta. Further we will consider orthonormal sets of polynomials over $[-1, 1]$ range, thus c_n will equal one for all n.

Jacobi polynomials belong to a very wide class of orthogonal polynomials, indeed they form a complete orthogonal system in the interval $[-1, 1]$

with respect to the following weighting function :

$$w_{\alpha,\,\beta}(x) \;=\; (1-x)^{\alpha}\;(1+x)^{\beta}\,, \tag{A.2}$$

where α and β are real scalars greater than -1. Jacobi polynomials satisfy, see [24], the following relation :

$$\begin{aligned}\int_{-1}^{1} P_m^{(\alpha,\,\beta)}\,P_n^{(\alpha,\,\beta)}\,(1-x)^{\alpha}\,(1+x)^{\beta}\;dx \;=\\ =\;\; \frac{2^{\alpha+\beta+1}}{2n+\alpha+\beta+1}\,\frac{\Gamma(n+\alpha+1)\,\Gamma(n+\beta+1)}{n!\,\Gamma(n+\alpha+\beta+1)}\,\delta_{m,\,n}\,,\end{aligned} \tag{A.3}$$

where $\Gamma(z)$ is the gamma function :

$$\Gamma(z) \;=\; \int_0^{\infty} t^{z-1}\,e^{-t}\;dt\,.$$

Any polynomial, $P_n^{(\alpha,\,\beta)}(x)$, can be associated to an ordered list of coefficients which are computed by means of the following recurrence relation :

$$P_k(x) \;=\; \psi_1(x)\,P_{k-1}(x) \;-\; \psi_2(x)\,P_{k-2}(x)\,,$$

where k is an integer which runs over all degrees of freedom of $P_n^{(\alpha,\,\beta)}(x)$ and $\psi_{1,\,2}(x)$ are

$$\begin{aligned}\psi_1(x) &= \frac{(a-1)(\alpha^2-\beta^2)+(a-4)(a-3)(a-2)x}{2k(a-k)(a-2)}\\ \psi_2(x) &= \frac{2a(k+\alpha-1)(k+\beta-1)}{2k(a-k)(a-2)}\,,\end{aligned}$$

with $a = 2k+\alpha+\beta$. In order to complete the numerical procedure it is necessary to choose the initial conditions,

$$\begin{aligned}P_0(x) &= 1\\ P_1(x) &= \frac{\alpha-\beta}{2}+\frac{\alpha+\beta+2}{2}\,x\,,\end{aligned}$$

and to normalize at each k-step $P_k(x)$ with its evaluation in 1, $P_k(1)$. Finally, aiming to obtain an orthonormal set, we proceed as follows

$$p_n^{(\alpha,\,\beta)}(x) \;=\; \frac{P_n^{(\alpha,\,\beta)}(x)}{\|P_n^{(\alpha,\,\beta)(x)}\|_{w_{\alpha,\,\beta}}}\,,$$

where

$$\|P_n^{(\alpha,\,\beta)}(x)\|_{w_{\alpha,\,\beta}} \;=\; \sqrt{\int_{-1}^{1} \left(P_n^{(\alpha,\,\beta)}(x)\right)^2 \, w_{\alpha,\,\beta}(x)\; dx}\,.$$

Note that, knowing all coefficients of the polynomial expansion $P_n^{(\alpha,\,\beta)}(x)$, the integral in the expression above can be computed exactly.

A.2 Gaussian integration

In order to solve an integral numerically, quadrature formulas are used. A quadrature formula is basically a set of distinct abscissas $\{\, x_0,\, \ldots\,,\, x_n \,\}$ over a range $[-1, 1]$ and a set of corresponding scalars, called weights, $\{\, \alpha_0,\, \ldots\,,\, \alpha_n \,\}$, which verify :

$$\int_{-1}^{1} p_k(x)\, w(x)\; dx \;=\; \sum_{i=0}^{n} \alpha_i\, p_k(x_i)\,, \tag{A.4}$$

where $p_k(x)$ is a polynomial of order k and $w(x)$ is the weighting function. The maximum k for which eq (A.4) stands is called degree of precision of the quadrature. Gaussian quadratures aim to obtain the best numerical estimate of an integral for a given number of integration nodes x_i.

The fundamental theorem of gaussian quadrature, see theorem 10.1 in [25], states that the optimal abscissas of a quadrature formula are the roots of the orthogonal polynomial for the same interval and weighting function.

Proposition A. *Being* $\{\, x_0,\, \ldots\,,\, x_n \,\}$ $n+1$ *distinct abscissas over a range* $[-1, 1]$*, the quadrature formula (A.4) has a degree of precision equal to*

$n + m$ (where m is a positive integer) if and only if the nodal polynomial $\omega_{n+1}(x)$, interpolating all the x_i abscissas, verifies :

$$\int_{-1}^{1} \omega_{n+1}(x)\, p(x)\, w(x)\; dx \;=\; 0 \quad \forall p(x) \in \mathbb{P}_{m-1} \tag{A.5}$$

According to (Proposition A), obtaining optimum quadratures is equivalent to find the nodal polynomial $\omega_{n+1}(x)$ which verifies eq. (A.5) for the biggest integer m. Then, the roots of $\omega_{n+1}(x)$ will be the optimum abscissas of the quadrature formula for the same interval and weighing function.

A.2.1 Gauss quadrature

The very optimum quadrature is obtained by avoiding to impose any kind of constraint on the abscissas of the quadrature formula. Thus, supposing that $w(x)$ equals $w_{\alpha,\beta}(x)$, we have that :

$$\int_{-1}^{1} P_{n+1}^{(\alpha,\beta)}(x)\, p(x)\, w_{\alpha,\beta}(x)\; dx \;=\; 0 \quad \forall p(x) \in \mathbb{P}_n\,. \tag{A.6}$$

Then, according to (Proposition A), the N-roots of $P_N^{(\alpha,\beta)}(x)$ define the abscissas of a quadrature which is exact for polynomials of order $2N-1$. This renowned quadrature is called gauss quadrature.

Before going further it is interesting to point out the reason why eq. (A.6) stands.

Observation A. *A set of Jacobi polynomials of order less or equal than m, $\{\, P_i^{(\alpha,\beta)}(x)\,\}_{i=0}^{m}$, is a basis for the space of polynomials of order less or equal than m, $\mathbb{P}_m$. Therefore, (A.6) can be rewritten as :*

$$\int_{-1}^{1} P_{n+1}^{(\alpha,\beta)}(x) \left(\sum_{j=0}^{n} c_j\, P_j^{(\alpha,\beta)}(x) \right) w_{\alpha,\beta}(x)\; dx \tag{A.7}$$

where $p(x)$ was expressed as its linear expansion on Jacobi polynomials. Finally, the orthogonality of $P_{n+1}^{(\alpha,\beta)}(x)$ with respect of all $\{\, P_j^{(\alpha,\beta)}(x)\,\}_{i=0}^{n}$ states that the eq. (A.7) is null.

A.2.2 Radau and Lobatto quadratures

Gauss quadrature is the best one, nevertheless it is possible to obtain slightly less optimal quadratures, imposing a constraint on one or two quadrature nodes :

- When immobilizing just one quadrature abscissa in the element boundary, it will be referred to radau quadratures
- When the both endpoints of the interval $[-1, 1]$ are included in the quadrature abscissas, it will be referred to lobatto quadratures

As done in section A.2.1, it is necessary to find a nodal polynomial $\omega_{n+1}(x)$ which verifies eq. (A.5) for the greatest possible m-value. Being $w(x)$ equal to $w_{\alpha\,\beta}(x)$, we choose :

$$\omega_{n+1}(x) = P_n^{(\alpha+1,\,\beta)}(x)(1-x) \qquad \{radau_1\ \} \tag{A.8}$$

$$\omega_{n+1}(x) = P_n^{(\alpha,\,\beta+1)}(x)(1+x) \qquad \{radau_{-1}\} \tag{A.9}$$

$$\omega_{n+1}(x) = P_{n-1}^{(\alpha+1,\,\beta+1)}(x)(1-x)(1+x) \qquad \{lobatto\ \} \tag{A.10}$$

which, replaced in eq.(A.5), state the following relations

$$\int_{-1}^{1} P_n^{(\alpha+1,\,\beta)}(x)\,p(x)\,w_{\alpha+1,\,\beta}(x)\;dx = 0 \qquad \forall p(x) \in \mathbb{P}_{n-1}$$

$$\int_{-1}^{1} P_n^{(\alpha,\,\beta+1)}(x)\,p(x)\,w_{\alpha,\,\beta+1}(x)\;dx = 0 \qquad \forall p(x) \in \mathbb{P}_{n-1}$$

$$\int_{-1}^{1} P_{n-1}^{(\alpha+1,\,\beta+1)}(x)\,p(x)\,w_{\alpha+1,\,\beta+1}(x)dx = 0 \qquad \forall p(x) \in \mathbb{P}_{n-2}$$

Thus, according to (Proposition A), the N-roots of (A.8) and (A.9) are radau quadrature nodes, exact for polynomial degrees at most equal to $2N-2$, whereas the N-roots of (A.10) are lobatto quadrature nodes, exact for polynomial degrees at most equal to $2N-3$.

A.2.3 Polynomial roots and quadrature weights

In section A.1 it was shown how to calculate coefficients of a Jacobi polynomial and, in sections A.2.1 and A.2.2, that their roots are abscissas of optimum quadratures. Then it remains to derive an algorithm to compute roots of mono-dimensional polynomials and a method to obtain quadrature weights from a set of quadrature abscissas.

Polynomial roots were calculated with the *Newton-Horner method*, which makes use of the deflation method to eliminate successively found roots. Any mono-dimensional polynomial

$$p_n(x) \;=\; \sum_{k=0}^{n} a_k\, x^k$$

can be written as :

$$p_n(x) \;=\; b_0 \;+\; (x - z)\, q_{n-1}(x\,;\, z)\,, \tag{A.11}$$

where z is any position on the x-axis and $q_{n-1}(x\,;\, z)$ is the associated polynomial to $p_n(x)$, which is

$$q_{n-1}(x\,;\, z) \;=\; \sum_{k=1}^{n} b_k\, x^{k-1}\,,$$

where coefficients $b_k(z)$ are computed by means of the following recurrence relation :

$$b_n \;=\; a_n \quad \ldots \quad b_k \;=\; a_k \;+\; b_{k+1}\, z \quad \ldots \quad b_0 \;=\; p_n(z)\,.$$

Using eq. (A.11), the polynomial derivative can be written as :

$$p'_n(x) \;=\; q_{n-1}(x\,;\, z) \;+\; (x - z)\, q'_{n-1}(x\,;\, z)\,,$$

thus $p'_n(z) \;=\; q_{n-1}(z\,;\, z)$ and Newton method becomes :

$$r_j^{(k+1)} \;=\; r_j^{(k)} \;-\; \frac{p_n(r_j^{(k)})}{p'_n(r_j^{(k)})} \;=\; r_j^{(k)} \;-\; \frac{p_n(r_j^{(k)})}{q_{n-1}(r_j^{(k)}\,;\, r_j^{(k)})}\,, \tag{A.12}$$

where $r_j^{(k)}$ is the j^{th} root at iteration (k).
Finally, the implemented process can be sketched as follows

- Iterate eq. (A.12) up to convergence
- Eliminate $(x - r_j^{(k)})$ root from $p_n(x)$, thus obtaining : $p^*_{n-1}(x) \;=\; q_{n-1}(r_j^{(k)}\,;\, r_j^{(k)})$, according to eq. (A.11).
- Repeat the procedure with $p^*_{n-1}(x)$. [1]

In order to determine quadrature weights, we, first, find the lagrangian basis $\{\phi_i(x)\}_{i=0}^n$ corresponding to the quadrature node set $\{x_i\}_{i=0}^n$,

$$\phi_i(x) \;=\; \frac{\pi(x)}{(x - x_i)\pi'(x_i)} \qquad \forall i \in \mathbb{N}_{(0,\,n)}\,,$$

where

$$\pi(x) = \prod_{i=0}^{n} (x - x_i) \quad \text{and} \quad \pi'(x_i) = \prod_{j=1,\,j\neq i}^{n} (x_i - x_j)$$

and, then, the integral of each lagrangian shape function $\phi_i(x)$ equals the integration weight α_i associated to quadrature node x_i

$$\alpha_i \;=\; \frac{1}{\pi'(x_i)} \int_{-1}^{1} \frac{\pi(x)\, w_{\alpha,\,\beta}(x)}{x - x_i}\; dx\,.$$

A.3 Tensor-product domains

Previous sections show how to derive $1D$ optimal quadrature formulas. Actually we want to introduce those $2D$ and $3D$ domains for which optimal cubatures are available. Cubatures for squares and cubes are obtained by means of tensor products of corresponding $1D$ quadratures.

[1] In case of $\alpha = \beta$ the Jacobi polynomial is symmetric, then at each iteration it is possible to eliminate two roots.

Hereinafter this renowned result is shown for the $\Omega = [-1,1]^3$ domain. Consider the integration of any Q_k, which is any linear combination of $\{x^i y^j z^h\}_{i,j,h=0}^{k}$ in Ω,

$$\int_{\Omega} Q_k \, \mathrm{d}\mathbf{x} \;=\; \sum_{i,\,j,\,h=0}^{k} c_{ijh} \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} x^i \, y^j \, z^h \; dx\, dy\, dz \,,$$

where c_{ijk} are coefficients of the linear expansion, then, using tensor product domain property, we write

$$\int_{\Omega} Q_k \, \mathrm{d}\mathbf{x} \;=\; \sum_{i,\,j,\,h=0}^{k} c_{ijh} \int_{-1}^{1} x^i \; dx \int_{-1}^{1} y^j \; dy \int_{-1}^{1} z^h \; dz \,.$$

Finally, using eq. (A.4) and the linearity of integration, the cubature rule for Ω reads :

$$\int_{\Omega} Q_k \, \mathrm{d}\mathbf{x} \;=\; \sum_{o,\,m,\,n=1}^{N} \alpha_o \, \alpha_m \, \alpha_n \, Q_k(x_o,\, y_m,\, z_n) \,,$$

where N is the number of $1D$ quadrature nodes needed to integrate a polynomial of degree at most equal to k.

A.4 Non-tensor-product domains

Non-tensor-product domains do not benefit of Fubini's theorem, nevertheless it is possible to derive non-optimal cubatures for this kind of domains by mapping their coordinates in associated tensor-product domain cubatures. This method is equivalent to that used to transfer cubatures in the real space as done in section 2.3. However, in this case, the mapping $\widehat{T}$ is known, then its jacobian will be included in the weighting function $w_{\alpha,\,\beta}(x)$.

In the following tables are presented all the "ingredients" which allow to construct cubatures which integrate polynomials on all "classic" $2D$ and

K_{type}	Ω
Triangle	$\{(x,y) : -1 \leq x,y \,;\, x+y \leq 0\}$
Tetrahedron	$\{(x,y,z) : -1 \leq x,y,z \,;\, x+y+z \leq -1\}$
Pyramid	$\{(x,y,z) : (z-1)/2 \leq x,y \leq (1-z)/2 \,;\, -1 \leq z \leq 1\}$
Prism	$\{(x,y,z) : -1 \leq x,y,z \leq 1 \,;\, x+y \leq 0\}$

Table A.1: "Classic" element shapes are listed with the definition of their domain, Ω

K_{type}	$\vec{x} = \widehat{T}(\vec{X})$	$\lvert J_{\widehat{T}} \rvert$
Triangle	$x = (1+X)(1-Y)/2 - 1$ $y = Y$	$\frac{1-Y}{2}$
Tetrahedron	$x = (1+X)(1-Y)(1-Z)/4 - 1$ $y = (1+Y)(1-Z)/2 - 1$ $z = Z$	$\frac{(1-Y)(1-Z)^2}{8}$
Pyramid	$x = X(1-Z)/2$ $y = Y(1-Z)/2$ $z = Z$	$\frac{(1-Z)^2}{4}$
Prism	$x = (1+X)(1-Y)/2 - 1$ $y = Y$ $z = Z$	$\frac{1-Y}{2}$

Table A.2: Mapping of tensor-product domain in the non-tensor-product domain, $\vec{x} = \widehat{T}(\vec{X})$, and its jacobian, $\lvert J_{\widehat{T}} \rvert$

$3D$ non-tensor-product domains.

As presented in the next section, all concepts above were implemented

K_{type}	$\omega_{k+1}(\vec{X})$	$p_k(\vec{x})$	$\alpha \in \vec{x}$
Triangle	$P_k^{(0,\,0)}(X)$ $P_k^{(1,\,0)}(Y)$	$\{x^i\, y^j : i{+}j \leq k\}$	$\frac{\alpha_o\, \alpha_m}{2}$
Tetrahedron	$P_k^{(0,\,0)}(X)$ $P_k^{(1,\,0)}(Y)$ $P_k^{(2,\,0)}(Z)$	$\{x^i\, y^j\, z^h : i{+}j{+}h \leq k\}$	$\frac{\alpha_o\, \alpha_m\, \alpha_n}{8}$
Pyramid	$P_k^{(0,\,0)}(X)$ $P_k^{(0,\,0)}(Y)$ $P_k^{(2,\,0)}(Z)$	$\{x^i\, y^j\, z^h : i{+}j{+}h \leq k\}$	$\frac{\alpha_o\, \alpha_m\, \alpha_n}{4}$
Prism	$P_k^{(0,\,0)}(X)$ $P_k^{(1,\,0)}(Y)$ $P_k^{(0,\,0)}(Z)$	$\{x^i\, y^j\, z^h : i{+}j, h \leq k\}$	$\frac{\alpha_o\, \alpha_m\, \alpha_n}{2}$

Table A.3: Tensor-product domain polynomials, $\omega_{k+1}(\vec{X})$, whose root tensor product gives cubature nodes. Definition of polynomials which are exactly integrated in non-tensor-product domain, $p_k(\vec{x})$. Relation between non-tensor-product weights, $\alpha \in \vec{x}$, and tensor-product weights.

in C++ class. Tests show a perfect agreement with results obtained by means of integration rules which can be found in [30], [31], [32] and [33]. As already remarked, the quadratures derived so far are optimum just for tensor-product domains, then, in the main code, integration rules for triangles and tetrahedrons were copied from [33]. However, little programs, like those producing data in sections 2.2, 2.4 and 2.5 or plotting shape functions in section 2.7, made use of this C++ class.

A.5 C++ class structure

In figure A.1 is depicted the hierarchic structure which provides integration rules for all "classic" reference elements.
All modules are shortly described below :

(a) Computes cubatures for "classic" reference elements. Its templates parameters are listed form left to right :

- Polynomial type (P_k, Q_k, ...)
- Reference frame (edge, quadrilateral, triangle, ...)
- Including only inner nodes (VI) or boundary nodes too (VB)

(b) Returns all geometric information about reference elements. See tables A.1 and A.2

(c) Provides any kind of quadrature in the range $[-1, 1]$:

- Gauss, inner nodes only. See section A.2.1
- Lobatto, both the endpoints included. See section A.2.2
- Radau, one enpoint (-1 or 1) included. See section A.2.2

Those classes get α and β scalars, as non-template parameters, too

(d) Computes lagrangian bases and derives cubature weights. See section A.2.3

(e) Computes Jacobi polynomials and derives cubature nodes. See section A.2.3

(f) Any $1D$ polynomial is associated to its expansion coefficients. Then, all basic operations among polynomials are defined acting on their expansion monomials (summation, difference, product, integration, derivation, ...). Furthermore this module makes use of MAPM class, [34], which allows the use arbitrary precision, in order to obtain accurate results even for very high order quadratures.

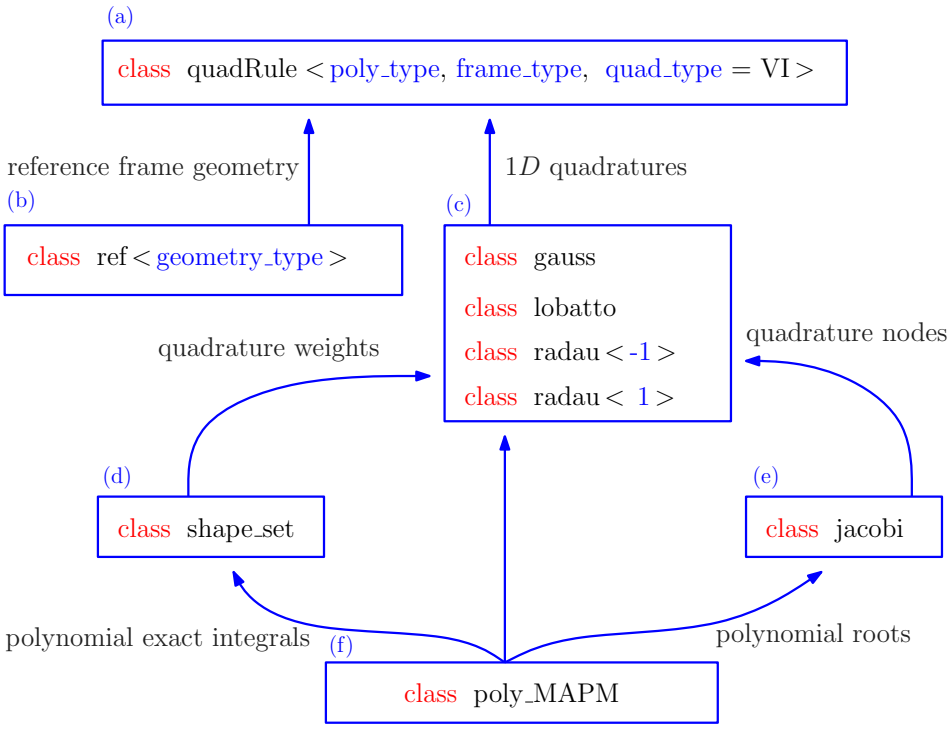


Figure A.1: Graph of the cubature class structure

Appendix B

Geometries and mappings

Discontinuous Galerking methods are well-suited for high-order representation of both solution and geometry. DGFEM can achieve higher accuracy on coarser meshes than low-order methods. However, in order to realize this advantage, it is critical that solid wall boundaries be represented with high-order polynomials compatible with the order of the interpolation for the state variables. Otherwise, numerical errors generated by the low-order boundary representation may overwhelm any potential accuracy gains offered by high-order methods. The importance of this high-order boundary representation is demonstrated with several well-know inviscid, [35] [36], viscous, [37], and turbulent flow test cases, [38].

Let us consider "classic" first order geometries, triangles and quadrilaterals with straight faces are defined by three and four vertices, respectively. Thus, their mappings in the real space are polynomials of three degrees of freedom for triangles and of four degrees of freedom for quadrilaterals. Then, the choice of polynomial spaces for those element mappings is straightforward. Triangle reference coordinates are mapped in the real space by means of

$$\mathbb{P}_k \stackrel{\text{def}}{=} \{\, X^i\, Y^j \,:\, i+j \,\leq\, k \,\}$$

polynomials, whereas quadrangle mappings use

$$\mathbb{Q}_k \stackrel{\text{def}}{=} \{\, X^i\, Y^j \;:\; i,\, j \;\leq\; k \,\}$$

polynomials. As a matter of fact, $\mathbb{P}_1$ and $\mathbb{Q}_1$ polynomials have respectively three and four degrees of freedom. Moreover this choice is "compatible" with integration rules. In square reference domains, tensor-product cubatures integrate exactly $\mathbb{Q}_k$ polynomials, see section A.3, whereas in triangular reference regions, cubatures integrate exactly $\mathbb{P}_k$ polynomials, see table A.3.

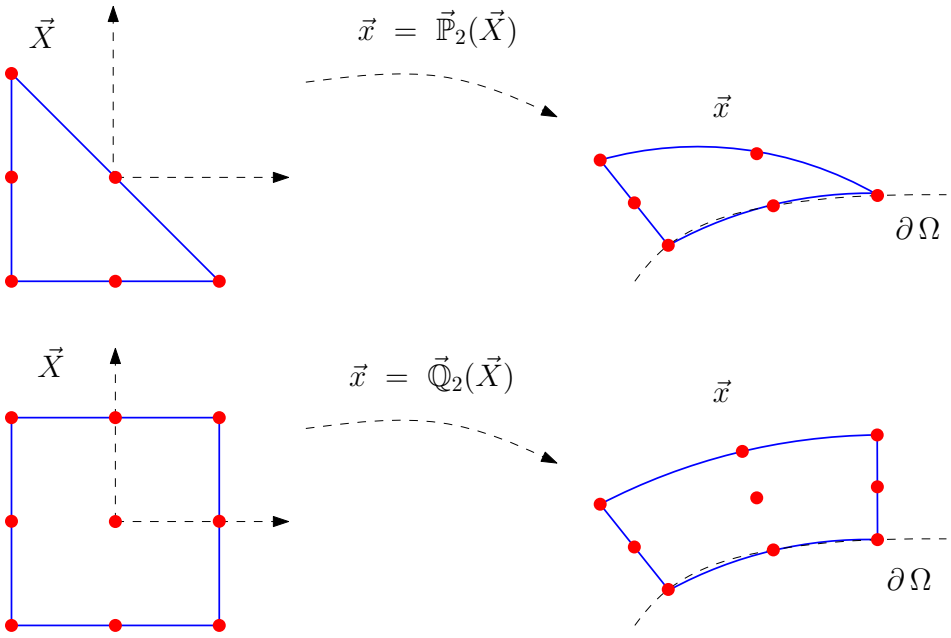


Figure B.1: Second order triangles and quadrilaterals

Arguing from analogy with first order geometries, we state that tringles

and quadrangles, whose curved faces are approximated by a p_k polynomial, are mapped in the real space by means of $\mathbb{P}_k$ and $\mathbb{Q}_k$ polynomials, respectively. As shown in figure B.1, once the polynomial degree of boundary representation, k, is chosen, so it is for the number of degrees of freedom which define the geometry in the real space. Hence, the number of nodes, defining k-order elements, is $(k+1)(k+2)/2$ for triangles and $(k+1)(k+1)$ for quadrilaterals. Some nodes are located in the element boundary, as required by face discretizations, and the rest of them are placed in the interior region.

Bibliography

[1] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini. Unified analysys of discontinuous galerkin methods for elliptic problems. *SIAM J.Numer.Anal.*, 39:1749–1779, 2001.

[2] F. Bassi, A. Crivellini, and D.A. Di Pietro. An artificial compressibility flux for the discontinuous galerkin solution of the incompressible navier stokes equations. *submitted to Elsevier Science*, 2005.

[3] F. Bassi, A. Crivellini, D.A. Di Pietro, and S. Rebay. An implicit high-order discontinuous galerkin method for steady and unsteady incompressible flows. *submitted to Computer and Fluids*, 2006.

[4] F. Bassi and S. Rebay. Accurate 2d euler computations by means of a high order discontinuous finite element method. *Lecture Notes in Ph.*, XIV ICNMFD, Bangalore, 1994.

[5] F. Bassi and S. Rebay. Numerical solution of euler equations with a multiorder dicontinuous finite element method. *K.S.S. Armfield,P.Morgan (Eds.), Proceedings of the Second international Conference on Computational Fluid Dynamics*, Springer, Berlin:199–204, 2002.

[6] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini. A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows. *2nd Europian Conference on Turbomachinery Fluid Dynamics and Thermodynamics (Antwerpen, Belgium), Teconologish Instituut,*, pages 99–108, 1997.

[7] B. Cockburn, S. Hou, and C.W. Shu. The runge-kutta local projection discontinuous galerkin finite element method for conservations laws iv: The multidimensional case. *Math Comput.*, 54:545, 1990.

[8] K.J. Fidkowski. A high-order discontinuous galerkin multigrid solver for aerodynamic applications. *Master Thesis in Aerospace Engineering*, Massachusetts institute of technology, 2004.

[9] K.J. Fidkowsky, T.A. Oliver, and D.L. Darmofal. p-multigrid solution of high-order discontinuous galerkin discretizations of the compressible navier stokes equations. *Journal of Computational Physics*, 207:92–113, 2005.

[10] S.K. Godunov. A difference scheme for numerical computiations of discontinuous solutions of hydrodynamic equations. *Math Sb.*, 47:271, 1959. [Russian][Translated by U.S. Joint Publ. Res. Service (1969)].

[11] B.T. Helenbrook and D.J. Mavriplis. Analysis of p-multigrid for continuous and discontinuous finite element discretisations. *AIAA*, 2003.

[12] K. Hillewaert, J.F. Remacle, N. Cheveaugeon, P.E. Bernard, and P. Geuzaine. Analysis of hybrid p-multigrid method for the discontinuous galerkin discretization of the euler equations. *ECCOMAS CFD*, 2006.

[13] C.R. Nastase, D.J. Mavriplis. High-order discontinuous galerkin methods using an hp-multigrid approach. *Journal of Computational Physics*, 213:330–357, 2006.

[14] C.R. Nastase, D.J. Mavriplis. Discontinuous galerkin methods using an hp-multigrid solver for inviscid compressible on three dimentional unstructured meshes. *AIAA*, 107, 2006.

[15] M.M. Rashid and M. Selimotic. A three-dimensional finite element method with arbitrary polyhedral elements. *Int. J. Numer. Meth. Engng.*, 67:226–252, 2006.

[16] A.H. Shapiro. The dynamics and thermodynamics of compressible fluid flow. *Ronald Press Company, New York*, vol. II, 1953.

[17] I. Moulitsas and G. Karypis. Multilevel Algorithms for Generating Coarse Grids for Multigrid Methods. http://www-users.cs.umn.edu/ moulitsa/software.html. University of Minnesota, Departement of Computer Science / Army HPC Research Center Minneapolis

[18] A. Brandt. Guide to Multigrid Developement. *Springer-Verlag*, 1982.

[19] B.T. Helenbrook and H.L. Atkins. Coupling p-multigrid to geometric multigrid for discontinuous Galerkin formulations of the Poisson equation. *18th AIAA Computational Fluid Dynamics Conference*, Miami, FL, June 2007.

[20] S.J. Sherwin and G.E. Karniadakis A triangular spectral element method; applications to the incompressible Navier-Stokes equations. *Computer methods applied mechanics and engineering*, 123:189–229, 1995.

[21] S.X. Giraldo and T. Warburton A nodal triangle-base spectral element method for the shallow water equations on the sphere *Journal of computational physics*, 207:129–150, 2005

[22] S.X. Giraldo High order triangle-base discontinuous Galerkin methods for hyperbolic equations on a rotating sphere *Journal of computational physics*, 214:447–465, 2006

[23] C. Lomtev, I. Quillen and G.E. Karniadakis Spectral/hp methods for viscous compressible flows on unstructured 2D meshes *Journal of computational physics*, 144:325–347, 1998.

[24] E. W. Weisstein, Jacobi Polynomial, http://mathworld.wolfram.com/JacobiPolynomial.html.

[25] A. Quarteroni, R. Sacco and F. Saleri, *Numerical Mathematics* (Springer-Verlag, New-York, 2000).

[26] S. Sherwin and G.E. Karniadakis, Thetrahedral hp finite elements: algorithms and flow simulations *Journal of computational physics* **124**, 14 (1996).

[27] C. Lomtev, I. Quillen and G.E. Karniadakis Spectral/hp methods for viscous compressible flows on unstructured 2D meshes *Journal of computational physics* **144**, 325 (1998).

[28] M. Bos, L. Taylor and B. Wingate Tensor product Gauss-Lobatto points are Fekete points for the cube *Mathematics of computation* **70**, 1543 (2000).

[29] M. Vincent, R.C. Taylor and B. Wingate An algorithm for computing Fekete points in the triangle *Siam Journal of Numerical Analysis* **38**, 1707 (2000).

[30] R. Cools, Encyclopaedia of Cubature Formulas, http://www.cs.kuleuven.ac.be/ nines/research/ecf/ecf.html.

[31] R. Cools An Encyclopaedia of Cubature Formulas *J. Complexity* **19**, 445 (2003).

[32] R. Cools Monomial cubature rules since "Stroud": a compilation – part 2 *J. Comput. Appl. Math.* **112**, 21 (1999).

[33] P. Solin, K. Segeth and I. Dolezel High-Order Finite Elements Methods *Chapman & Hall/ CRC Press*, Boca Raton, FL (2003).

[34] M. C. Ring, MAPM, A Portable Arbitrary Precision Math Library in C, http://www.tc.umn.edu/ ringx004/mapm-main.html.

[35] F. Bassi and S. Rebay High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations *Journal of Computational Physics*, 138:251–285, 1997.

[36] Z.J. Wang and Yen Liu Extension of the spectral volume method to high-order boundary representation *Journal of Computational Physics*, 211:154–178, 2006.

[37] F. Bassi and S. Rebay A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations *Journal of Computational Physics*, 131:267–279, 1997.

[38] F. Bassi, A. Crivellini, S. Rebay and M. Savini Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k-ω turbulence model equations *Computers & Fluids* 34:507–540, 2005.

[39] F. Bassi, A. Ghidoni, S. Rebay and P. Tesini High-order accurate p-multigrid discontinuous Galerkin solution of the Euler equations *submitted to : Int. J. Numer. Meth. Fluids*

[40] L. Giraud, J. Langou and M. Rozloznik On the loss of orthogonality in the Gram-Schmidt orthogonalization process *CEFACS, Technical Report No. Tr/PA/03/25*

[41] F. Bassi, S. Rebay Numerical solution of the euler equations with a multiorder discontinuous finite element method. *Computational Fluid Dynamics 2002: Proceedings of the Second International Conference on Computational Fluid Dynamics*, S. Armfield, P. Morgan, K. Srinivas, Springer Verlag: Sydney, 2002; 207:92–113