Hybrid High-Order Methods for Anisotropic Diffusion Problems in 3D

Relatore: Prof. Luca FORMAGGIA
   Politecnico di Milano
Correlatori: Prof. Daniele A. DI PIETRO
   Institut Montpellierain A. Grothendieck
   Jérôme BONELLE
   EDF R&D

Tesi di Laurea di:
Riccardo MILANI
Matr. 849939
Politecnico di Milano,
École polytechnique

Anno Accademico 2015-2016
Sommario

Il metodo Hybrid High-Order (HHO) è stato recentemente sviluppato per la risoluzione di problemi alle derivate parziali ellittici su generiche griglie poliedriche. Vengono presi in considerazione gradi di libertà di spazi polinomiali definiti sia sulle facce, sia sulle cellule. Il grado di questi spazi è arbitrario e può essere scelto diverso per le facce e per le cellule. Gli strumenti principali del metodo sono un operatore di ricostruzione locale del potenziale, che assicura la consistenza, e un termine di stabilizzazione alla minimi-quadrati, per imporre la corrispondenza sulle facce tra i gradi di libertà faccia e quelli cellula. Questo lavoro si concentra in particolare sui problemi di diffusione anisotropa in 3D, i quali non hanno ricevuto una grande attenzione dal punto di vista numerico. In una prima parte discuteremo i principali risultati teorici del metodo e, in seguito, daremo maggiore spazio all’implementazione parallela di questo metodo nel software industriale di CFD Code_Saturne e a un’accurata analisi numerica e delle performances.

Parole Chiave: Hybrid High-Order, HHO, Diffusione Anisotropa, Griglie Generiche, 3D, Code_Saturne, Parallelizzazione.
Abstract

The Hybrid High-Order (HHO) method is a recently developed scheme for elliptic PDEs on general polytopal meshes. Its functional setting includes both face- and cell-based polynomial degrees of freedom. The order of the polynomial spaces is arbitrary and it may be different for the faces and the cells. The key-tools of the method are a local discrete potential reconstructor, ensuring consistency, and a least-square stabilization term, enforcing the matching between face- and cell-defined degrees of freedom. In this report, we will focus on the anisotropic diffusion primal problem in 3D, for which no in-depth numerical analysis is available. After a first part in which we will recall the major theoretic results, great emphasis will be given to the parallel implementation of such methods within the industrial CFD software Code_Saturne, and to a thorough numerical and performances analysis.

Keywords: Hybrid High-Order, HHO, Anisotropic Diffusion, General Meshes, 3D, Code_Saturne, Parallelization.
Ringraziamenti

Il progetto di questa tesi è stato realizzato durante uno stage presso l’EDF R&D Lab di Chatou, nel dipartimento Méchanique des Fluides, Energie et Environnement. Vorrei ringraziare in particolar modo Jérôme Bonelle, che mi ha sempre supportato e spinto a non tralasciare e migliorare ogni piccolo particolare. Il ringraziamento è naturalmente esteso a Martin Ferrand e, in generale, a tutta l’equipe di Code_Saturne: i mesi passati li sono stati davvero formativi e costituiscono un’esperienza molto positiva per la mia carriera.
## Contents

List of Figures ........................................... v
List of Tables ......................................... vii
List of Model-Problem- and Mesh-related Symbols .......... ix
List of Functional-Setting-related Symbols ............. x
List of HHO Operators and Matrices .................... xi

1 Introduction ............................................. 1
    1.1 Most Common Schemes for General Meshes .......... 1
        1.1.1 Lowest-Order Methods ..................... 1
        1.1.2 High-Order Methods ...................... 2
    1.2 The Hybrid High-Order Methods .................... 2
        1.2.1 Main Features .......................... 3
        1.2.2 Comparing HHO to Other Methods ........ 3
        1.2.3 Focus and Organization ................ 4

2 An Introduction to the HHO Methods .................. 7
    2.1 The Framework .................................... 7
        2.1.1 Model Problem and Notations ............ 7
        2.1.2 Admissible Mesh Sequences ............. 8
        2.1.3 Polynomial Spaces ..................... 9
        2.1.4 Diffusion Tensor ....................... 10
    2.2 The HHO Method ................................ 11
        2.2.1 Discrete Unknowns and the HHO(l)-family .. 11
        2.2.2 Essential HHO Operators ............... 12
            2.2.2.1 Potential Reconstructor .......... 12
            2.2.2.2 Stabilization .................. 15
        2.2.3 Discrete Problem and its Properties .... 17
            2.2.3.1 Discrete Formulation .......... 17
            2.2.3.2 Stability .................... 17
        2.2.4 Error Estimates .......................... 20
        2.2.5 Static Condensation ..................... 22
# A practical approach to HHO

## HHO Algebraic Framework

### Bases
- Monomial Bases
- Tuning the Bases up

### The Algebraic Framework
- Projection
- Reconstructor Operator
- Stabilization Operator
- The Local System
- Static Condensation
- The \((k+1)\)-th Order Reconstruction, \(p^{k+1}\)

## Implementation

### Choice of the Order of the Method

### Computing the Principal Axes of the Basis

### Dirichlet Boundary Conditions

### Integrating
- Matrix Filling

### Data Structures and Optimizations

### Parallelization

## Tests and results

### General Information

### Meshes

### Error Notation

### Elementary Tests

### Error Norms
- Norms Definition
- Results
- Discrete Max Principle

### Sensitivity

### Performances
- Parallelization and Profiling
- Total Times
- Choice of the Order

### Heterogeneous Anisotropy Tests

## Conclusions

### What has been achieved

### Final remarks and outlooks
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Face– and cell–defined polynomial DoFs of a cube</td>
<td>12</td>
</tr>
<tr>
<td>3.1</td>
<td>Face–basis setting</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Macro steps of the HHO method</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Subsimplex–subdivision for integrals computations</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Examples of the mesh families used for the tests</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>Relative energy errors</td>
<td>46</td>
</tr>
<tr>
<td>4.3</td>
<td>Relative project-related errors</td>
<td>47</td>
</tr>
<tr>
<td>4.4</td>
<td>Cell–DoFs of the solution of the FVCA6 test case</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Parallelization speed-up rates of the building times</td>
<td>53</td>
</tr>
<tr>
<td>4.6</td>
<td>Solving effort and discrete $L^2$ averaged errors</td>
<td>57</td>
</tr>
<tr>
<td>4.7</td>
<td>Solution of the heterogeneous anisotropy test</td>
<td>58</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Dimensions of several polynomial spaces \( P_d^k \) ........................................ 9
3.1 Dimensions of HHO local systems .................................................. 33
4.1 Maxima of the consistency errors ..................................................... 43
4.2 Orders of convergence of the approximation elementary tests .... 43
4.3 Summary of the error results ............................................................ 45
4.4 Discrete max principle results .......................................................... 49
4.5 Solver iterations with standard monomial bases ......................... 50
4.6 Comparison between different monomial basis types .................. 51
4.7 Solving times .................................................................................. 55
4.8 Errors of the heterogeneous test case ............................................ 59
# List of Model-Problem- and Mesh-related Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Omega)</td>
<td>Open, connected bounded polytopal domain, (\subset \mathbb{R}^d)</td>
</tr>
<tr>
<td>(f)</td>
<td>Source term of the model problem</td>
</tr>
<tr>
<td>(\psi_\partial)</td>
<td>Dirichlet boundary condition</td>
</tr>
<tr>
<td>(\phi_\partial)</td>
<td>Neumann boundary condition</td>
</tr>
<tr>
<td>(h)</td>
<td>Mesh size</td>
</tr>
<tr>
<td>(T_h)</td>
<td>Mesh</td>
</tr>
<tr>
<td>(T)</td>
<td>Generic cell, (\in T_h)</td>
</tr>
<tr>
<td>(F_h)</td>
<td>Set of the faces of (T_h)</td>
</tr>
<tr>
<td>(\mathcal{F}_T)</td>
<td>Set of the faces of a cell (T)</td>
</tr>
<tr>
<td>(F)</td>
<td>Generic face, (\in F_h)</td>
</tr>
<tr>
<td>(E)</td>
<td>Generic element of the mesh: cell (T) or face (F)</td>
</tr>
<tr>
<td>(h_F)</td>
<td>Diameter of a face (F)</td>
</tr>
<tr>
<td>(h_T)</td>
<td>Diameter of a cell (T)</td>
</tr>
<tr>
<td>(\mathbf{n}_{T,F})</td>
<td>Unit normal vector to (F) pointing out of (T)</td>
</tr>
<tr>
<td>(\mathbb{M})</td>
<td>Anisotropy tensor (piece-wise constant)</td>
</tr>
<tr>
<td>(\mu_{#T})</td>
<td>Greatest eigenvalue of (\mathbb{M}</td>
</tr>
<tr>
<td>(\mu_{\flat T})</td>
<td>Smallest eigenvalue of (\mathbb{M}</td>
</tr>
<tr>
<td>(\rho_T)</td>
<td>(\mu_{#T}/\mu_{\flat T})</td>
</tr>
<tr>
<td>(\mu_{T,F})</td>
<td>(\mathbb{M}\mathbf{n}<em>{T,F} \cdot \mathbf{n}</em>{T,F})</td>
</tr>
<tr>
<td>(n_F^T)</td>
<td>Number of faces of the cell (T), (= \text{card} \mathcal{F}_T)</td>
</tr>
</tbody>
</table>
List of Functional-Setting-related Symbols

$\mathbb{P}_m^k(\Omega)$  Space collecting the $m$-variate polynomial functions defined on $\Omega$ and of total degree up to $k$

$k$  Order of the face-based polynomial spaces

$l$  $\in \{k - 1, k, k + 1\}$

$l$  Order of the cell-based polynomial spaces, $= \max\{0, l\}$

$U_T^k$  Functional setting of the cell-defined DoFs of the cell $T$, $= \mathbb{P}_d^k(T)$

$\nu_T$  Generic function in $U_T^k$

$\Omega_F^k$  Functional setting of the DoFs of the face $F$, $= \mathbb{P}_{d-1}^k(F)$

$\nu_F$  Generic function in $\Omega_F^k$

$U_T^{k,F}$  Hybrid functional setting of the cell $T$, $= \mathbb{P}_{d}^l(T) \times \bigoplus_{F \in T} \Omega_F^k$

$\varphi_T$  Generic function in $U_T^{k,F}$

$B_T^k$  Set of basis functions of $\mathbb{P}_d^k(T)$

$\phi_T^k$  Generic, cell-defined basis function, $\in B_T^k$

$B_F^k$  Set of basis functions of $\mathbb{P}_{d-1}^k(F)$

$\varphi_F^k$  Generic, face-defined basis function, $\in B_F^k$

$B_{G}^{k+1}$  Set of basis functions used for the gradients, $= B_T^{k+1} \setminus \{\phi_T^0\}$

$c$  Dimension of the cell-defined polynomial space, $= \text{card } B_T^k$

$f$  Dimension of the face-defined polynomial space, $= \text{card } B_F^k$

$g$  Dimension of the gradient basis, $= \text{card } B_{G}^{k+1}$

$t$  Dimension of the hybrid local space, $= c + n_F f = \text{dim } U_T^k$
List of HHO Operators and Matrices

\[ \pi_T^k \quad L^2 \text{ orthogonal projection onto } \mathbb{P}_q^k(T) \]
\[ \pi_F^k \quad L^2 \text{ orthogonal projection onto } \mathbb{P}_d^k(F) \]
\[ I_T^k \quad \text{Reduction onto the local hybrid functional setting } U_T^k \]
\[ P_T^{k+1} \quad \text{HHO local potential reconstructor} \]
\[ q_T^{k+1} \quad \text{Operator giving a high order correction of the cell-DoFs} \]
\[ j_T \quad \text{Stabilization bilinear form} \]
\[ a_T \quad \text{Local bilinear form of the HHO weak formulation} \]
\[ a_h \quad \text{Global bilinear form of the HHO weak formulation} \]
\[ M_{EE} \quad \text{Mass matrix of a generic element } E \]
\[ G_T \quad \text{Matricial equivalent of } \nabla p_T^{k+1} \]
\[ J_T \quad \text{Matricial equivalent of } j_T(\cdot, \cdot) \]
\[ A_T \quad \text{Stiffness matrix of the HHO local system} \]
Chapter 1

Introduction

The interest in methods for elliptic PDEs on general meshes, possibly with non-conforming and polytopal cells, has constantly grown over the last few years. Two applications mainly lead the developments of such methods: geosciences and agglomeration procedures. In the first case, given the complexity of some soil layers, the meshes cannot be adapted to the needs of the standard methods; in the latter, error-driven fusion of two or more elements might result in polyhedral cells.

1.1 Most Common Schemes for General Meshes

1.1.1 Lowest-Order Methods

The first investigations around the possibility of using polytopal meshes concerned lowest-order schemes and several different approaches were devised.

A first family of methods is the Mimetic Finite Difference (MFD) schemes [12]. They were designed in order to have discrete formulations of differential operators and inner products which mimic the properties of their continuous counterparts and satisfy conservation laws and fundamental identities and theorems, for instance, the Green’s theorem. More recently, the MFD methods were extended also to general polyhedral meshes by [16,17,38].

Another direction of works intended to expand the classic finite volume schemes, hence the Mixed [30] and Hybrid [35,36] Finite Volume (MHFV) methods were devised. As the names suggest, the former includes vector-valued functions to approximate not only the solution, but its gradient too, and the latter includes scalar, edge-defined degrees of freedom to take into account the flux at the interfaces.

An original approach is the Compatible Discrete Operators (CDO) [14,15] framework. Three versions of this schemes are available, each hinged on the
discretisation of a Hodge operator combined with differential operator and operating on a primal and a dual mesh: a vertex-, a face- and a cell-based formulation. They have also been extended to the Stokes problem [13] and to the advection-diffusion one [18].

Generally, these families of methods lead to similar formulations: a bilinear form composed by two terms; one, usually based on a discrete operator acting as a gradient, ensure the consistency, and a stabilization needed to make the method robust. Moreover, most of the schemes here above actually share a common background and, in fact, they have been recently reorganized into macro families. The works in [31] demonstrates an algebraic equivalence between the MHFV and the MFD schemes. A larger group, the Gradient Schemes, was identified in [32] and it gathers some of the above-mentioned schemes, as well as other classic schemes such as the Galerkin and the nonconforming P1 Finite Elements methods. The CDO framework is another unifying viewpoint: its vertex-based version can be linked to the nodal MFD, and the cell-based one to MHFV and MFD.

1.1.2 High-Order Methods

More recently, more emphasis has been given to high-order polytopal methods, which usually lead to significantly convergence speed-ups whenever the solution is sufficiently regular.

An approach is given by the Virtual Elements Methods (VEM) [1, 8–11], derived by trying to extend the MFD framework to high orders. The approximation of the solution is sought in some particular polynomial spaces which, for a mesh element, have degrees of freedom attached to the vertices, the edges and the moments calculated on the element itself. A peculiar characteristic of the method is that there is no need of neither an explicit expression of the basis functions of these polynomial spaces, hence the name virtual, nor, more importantly, the use of quadratures. A method very close to VEM is the nonconforming VEM [4], sometimes called also High-Order Mimetic (HOM) [39]. It shares with the classic VEM the background on which it is based, in particular, it was devised as a high-order extension of MFD, too, but, the its original framework allows one to analyse simultaneously for any space dimension and any approximation order.

The Discontinuous Galerkin (DG) methods [2, 3, 24] are among the most used schemes and also their hybridizable version (HDG) [20] recently drew a lot of attention. The solution is characterized via the solutions of local problems which are globally coupled thanks to transmission conditions: jumps are allowed at the faces but they are penalized via a stabilization term included in the bilinear form used for the weak formulation of the problem.

1.2 The Hybrid High-Order Methods
1.2.1 Main Features

The focus of this work is on the Hybrid High-Order (HHO) methods, which, as the name suggests, can be included in the latest group of methods seen here above. Originally derived for the elasticity problem [26] and the isotropic diffusion [27], the HHO methods have witnessed a great development in the recent years and, so far, the framework has been extended, among others, to the advection [23] and even to the non-linear Leray–Lions problem [22]. It is defined hybrid because both cell- and face-based discrete unknowns are used. The related degrees of freedom are polynomial functions: their order \( k \geq 0 \) is arbitrary, hence the name high-order. The configuration allows arbitrary orders of convergence of the error estimates. Taking different values of the orders of the face-defined polynomial space, \( k \), and of cell-defined one, \( l \), leads to three different HHO-methods, gathered in the so-called HHO(\( l \))-family [19]. In fact, the cell-based polynomial order can be chosen among \( l \in \{k-1, k, k+1\} \), all ensuring the same properties.

In addition to the projection on polynomial spaces, two are the other key-tools of the HHO methods. The first is a local, high-order reconstruction operator: hinged on an simple integration by parts, it shows optimal approximation and consistency properties. This operator is also used to build a stabilization term, a least-squares penalization of the high-order terms, designed to try to avoid the possible multivaluedness at faces given by the hybridization and to match the properties of the reconstructor.

1.2.2 Comparing HHO to Other Methods

Some comparisons can be made to bridge HHO to some of the abovementioned methods. It has been shown [19] that, up to an equivalent choice of the stabilization, HHO, at its lowest order, \( k = 0 \), can be linked to the MHFV methods. Also, it can be viewed as a high-order extension of the face-based version of CDO.

A close relationship with the HDG framework, especially with the Local DG-hybridizable (LDG-H) [20] subclass, has been demonstrated [19]. In fact, HHO can be rewritten in a mixed formulation which takes into account the fluxes and which can be recast into an HDG-like form. The main difference between the two families of methods is the stabilization. They both penalize the residuals at the faces, in rough terms a difference between the face- and cell-defined function, but, differently from the naive version of HDG, HHO considers only high-order moments (of order \( k + 1 \)), precaution that will result in optimal approximation properties, in particular, a super convergence, order \( k + 2 \), for the \( L^2 \)-norm of the potential. Another difference between HHO and LDG-H is the functional setting in which the fluxes lie: the former uses \( \nabla P^k \) \( P^k \) \( \nabla \), the gradients of polynomial functions (dimension: \( \binom{k+1+d}{d} - 1 \) \( \binom{k+1+d}{d} - 1 \)), instead of the larger \( \nabla P^d \) \( P^d \) \( \nabla \) \( d \binom{k+d}{d} \) \( \binom{k+d}{d} \) considered in the HDG method. Moreover, the method HHO(\( l = k + 1 \)) is usually included among the HDG methods [19].
1.2. The Hybrid High-Order Methods

A first characteristic that distinguishes HHO from VEM is that, with VEM, boundary and internal degrees of freedom are combined to describe the same function, which is not polynomial; on the other hand, in HHO those degrees of freedom refer to different functions which are all polynomial. However, always [19] showed how the choice of the parameter $l = k - 1$ in HHO leads to the high-order MFD methods [4, 39]. Moreover, differently from VEM, the basis functions of the polynomial spaces have to be made explicit and integrals need to be computed, too. However, for the latter procedure, quadrature rules valid on generic elements can be used to ease the computation, for instance, a method as the one presented in [41] which explicitly addresses homogeneous functions and polynomial basis functions, or the procedure of reduction shown in [40], especially devised for the moments of inertia that can be adapted if monomial basis functions are considered.

Several reasons may lead one to choose the HHO methods. As it was already mentioned, up to minor modification of the fundamental key-tools, their framework can be extended to a large range of physical problems: quasi-incompressible elasticity [26], anisotropic and heterogeneous diffusion [25], advection [23] and non-linear problems [22]. A priori error estimations can be used to devise an agglomeration/refinement procedure to speed up the computations while improving the approximation [29]. Moreover, the construction of HHO is dimension-independent and its compact stencil allows to treat independently and at the same time different elements. Concerning the size of the global problem, the number of global degrees of freedom grows more slowly than other methods (for instance, for HHO they are $O(\frac{1}{2} k^2 N_F)$ and for DG $O(\frac{1}{6} k^3 N_E)$, $N_F$ and $N_E$ being the number of faces and elements in the mesh, respectively) and even a static condensation can be locally performed to eliminate the cell-based degrees of freedom, resulting in a substantial reduction of the global size.

1.2.3 Focus and Organization

The main subject of our work is the three-dimensional anisotropic diffusion, with mixed Dirichlet-Neumann boundary conditions. The theoretic background was already set in [25, 28], but, except for [29], which, however, deals with the mixed form, no numerical analysis is available for the 3D case. Moreover, ours is the first and, so far, only implementation of the HHO methods in an industrial code: in fact, our developments are integrated in Code_Saturne [33], an open-source, multi-purpose CFD software developed by EDF R&D.

Given the industrial context and the three-dimensional setting, a great emphasis will be given to the sensibility and conditioning of the method, and, more in particular, to the performances. Concerning the former subject, different types of monomial basis functions (of the polynomial spaces) taking into account the diffusion properties and the inertia of the elements were tested in order to find the one ensuring the best results. As for the computational times, a great attention was paid to optimizing the method as much as possible, and we were led to parallelize the code, making it among the first parallel implementations
of HHO. The code will be finally tested on the FVCA6 [37] benchmark.

The organization of this report is as follows. After chapter 2 in which we recall the main theoretic results of HHO and the details of their proofs, we will provide in chapter 3 a more practical and computing-exploitable framework and some key-points around which our implementation is build. Finally, in chapter 4, the main results concerning both the theory-driven error estimates and the performances will be presented.
Chapter 2

An Introduction to the HHO Methods

In this chapter, first, we set some notations and recall some theoretic notions in order to provide a basis on which the HHO methods will be developed, presented in the second part of the chapter, following the original paper [27] and the more recent review [28]. We will also take the time to detail the proofs of the most relevant results of the method.

2.1 The Framework

2.1.1 Model Problem and Notations

Let \( \Omega \subset \mathbb{R}^d \), \( d \geq 2 \), be an open, connected, bounded polytopal domain, whose boundary and outward normal will be referred to with \( \Gamma \) and \( \mathbf{n} \). It is supposed that there exists a partition of \( \Gamma \), \( \Gamma_d \) and \( \Gamma_n \), such that \( \Gamma_d \cup \Gamma_n = \Gamma \), \( \Gamma_d \cap \Gamma_n = \emptyset \), and \( \Gamma_d \) has nonzero measure.

A pure-diffusion model problem is addressed: Find \( u : \Omega \to \mathbb{R} \) such that

\[
- \text{div}(\mathbb{M} \text{div} u) = f \quad \text{in} \; \Omega, \\
u = \psi_\partial \quad \text{on} \; \Gamma_d, \\
\mathbb{M} \nabla u \cdot \mathbf{n} = \phi_\partial \quad \text{on} \; \Gamma_n, 
\]

where \( f \in L^2(\Omega) \), \( \psi_\partial = (u_\partial)|_{\Gamma_d} \) with \( u_\partial \in H^1(\Omega) \), and \( \phi_\partial \in L^2(\Gamma_n) \) (provided that \( \Gamma_n \) has nonzero measure). Finally, \( \mathbb{M} \) is a Symmetric Definite-Positive (SPD) tensor, such that there exists a partition \( P_\Omega \) of \( \Omega \) in which \( \mathbb{M} \) is piecewise Lipschitz.

For a generic domain \( X \subset \overline{\Omega} \), the usual scalar product in \( L^2(X) \) and the related norm will be denoted as \( (\cdot, \cdot)_X \) and \( \| \cdot \|_X \), respectively; the subscript \( X \) will be usually omitted whenever \( X = \Omega \). Let us define also \( U := H^1(\Omega) \) and \( U_0 = \{ v \in U \mid v|_{\Gamma_d} = 0 \} \), then the classical weak formulation of problem (2.1)
reads: Find \( u_0 \in U_0 \) such that
\[
(M \nabla u_0, \nabla v) = (f, v) - (M \nabla u_0, \nabla v) + (\phi_\partial, v)_{\Gamma_n} \quad \forall v \in U_0
\] (2.2)
and then the solution \( u \in U \) is recovered: \( u = u_0 + u_\partial \).

### 2.1.2 Admissible Mesh Sequences

We take as reference the definition of admissible mesh sequence of [24, Chapter 1]. Let \( \mathcal{H} \subset \mathbb{R}^*_+ \) be a countable set of meshsizes having 0 as its unique accumulation point. Consider the set \( \mathcal{H} \). For all \( h \in \mathcal{H} \), \( \mathcal{T}_h = \{ T \} \) is a finite collection of nonempty, disjoint, open polygons/polyhedra, usually referred to as *cells*, such that \( \bigcup_{T \in \mathcal{T}_h} \overline{T} = \overline{\Omega} \) and \( h = \max_{T \in \mathcal{T}_h} h_T \), \( h_T \) being the diameter of the cell \( T \).

A hyperplanar, closed, connected subset \( F \) of \( \overline{\Omega} \) is called a *face*, if it has positive \((d - 1)\)-dimensional measure and such that either (i) there exists \( T_1, T_2 \in \mathcal{T}_h \) such that \( F = \partial T_1 \cap \partial T_2 \) or \( F \subset \partial T_1 \cap \partial T_2 \) and \( F \) is a side of both \( T_1 \) and \( T_2 \) (that is, \( F \) is an *interface*), or (ii) there exists \( T \in \mathcal{T}_h \) such that \( F = \partial T \cap \partial \Omega \) or \( F \subset \partial T \cap \partial \Omega \) and \( F \) is a side of \( T \) (that is, \( F \) is a *boundary face*). The faces of the first type, the *interfaces* or *internal faces*, are collected in the set \( \mathcal{F}^i_h \), and the *boundary faces* in \( \mathcal{F}^b_h \), and finally let \( \mathcal{F}_h \) be the set collecting all the faces of a mesh \( \mathcal{F}_h := \mathcal{F}^i_h \cup \mathcal{F}^b_h \). Moreover, let us define the set \( \mathcal{F}_T \) which, for all \( T \in \mathcal{T}_h \), gathers the faces composing the boundary of \( T \), \( \mathcal{F}_T := \{ F \in \mathcal{F}_h \mid F \subset \partial T \} \), and the set \( \mathcal{T}_F \), which, for all \( F \in \mathcal{F}_h \), collects the one or two cells sharing \( F \), \( \mathcal{T}_F := \{ T \in \mathcal{T}_h \mid F \subset \partial T \} \). For every face \( F \in \mathcal{F}_h \), the diameter of the face is denoted by \( h_F \), the unit normal vector to \( F \) pointing out of \( T \) will be referred to by \( n_{F,T} \) and, for every interface, a direction is chosen arbitrarily once for all defining the unit normal vector \( n_F \), so that, given a \( T \in \mathcal{T}_F \), \( n_F = \pm n_{T,F} \) accordingly to the chosen direction.

We are now ready to give the definition of admissible mesh sequences.

**Definition 2.1.1** (Admissible mesh sequences) The mesh sequence \( (\mathcal{T}_h)_{h \in \mathcal{H}} \) is admissible if, for all \( h \in \mathcal{H} \), \( \mathcal{T}_h \) admits a matching simplicial subsmesh \( \mathcal{S}_h \) such that there exists a real number \( \gamma > 0 \), independent of \( h \), such that, for all \( h \in \mathcal{H} \):

(i) for all simplex \( S \in \mathcal{S}_h \) of diameter \( h_S \) and inradius \( r_s \), \( \gamma h_S \leq r_s \);

(ii) for all \( T \in \mathcal{T}_h \), and \( S \in \mathcal{S}_T := \{ S \in \mathcal{S}_h \mid S \subseteq T \} \), \( \gamma h_T \leq h_S \)

In order to have a complete framework, we also need the mesh to be somehow compatible with the boundary conditions [28].

**Definition 2.1.2** (Compatible mesh sequences) The mesh sequence \( (\mathcal{T}_h)_{h \in \mathcal{H}} \) is compatible if, for all \( h \in \mathcal{H} \)

(i) for all \( T \in \mathcal{T}_h \), there exists a unique \( \Omega \subseteq \partial \Omega \) (the partition associated to the diffusion tensor) containing \( T \);
(ii) one can define two sets, $F^{d}_{h}$ and $F^{n}_{h}$, $F^{d}_{h} \cup F^{n}_{h} = F^{b}_{h}$, with

$$F^{d}_{h} := \{ F \in F^{b}_{h} \mid F \subseteq \Gamma^{d}_{h} \}$$

$$F^{n}_{h} := \{ F \in F^{b}_{h} \mid F \subseteq \Gamma^{n}_{h} \}$$

We recall two results about admissible mesh sequences. [24, Lemma 1.41]: There exists a positive integer $N_{\gamma}$ depending on the mesh regularity parameter $\gamma$ and the dimension $d$, such that

$$\max_{T \in \mathcal{T}_{h}} \text{card}(F_{T}) \leq N_{\gamma} \quad \forall \ h \in \mathcal{H}. \quad (2.3)$$

[24, Lemma 1.42]: For all $h \in \mathcal{H}$, all $T \in \mathcal{T}_{h}$ and all $F \in F_{T}$, the following holds:

$$\gamma^{2}h_{T} \leq h_{F} \leq h_{T}. \quad (2.4)$$

**Note 2.1.1** In what follows, $T$ will usually denotes a cell, $F$ a face and $E \in \{T,F\}$ refers to an unspecified element of the mesh $\mathcal{T}_{h}$

### 2.1.3 Polynomial Spaces

$P^{k}_{m}(\Omega)$ Polynomial spaces play a central role in the HHO method, thus, let us introduce $P^{k}_{m}(\Omega)$, for integers $k \geq 0$ and $1 \leq m \leq d$, which is the space collecting the $m$-variate polynomial functions defined on $\Omega$, of total degree up to $k$; its dimension is $N_{k,m} := \binom{k+m}{k} = \binom{k+m}{m}$. Then, for all $T \in \mathcal{T}_{h}$, $P^{k}_{m}(T)$ denotes the restriction to $T$ of the functions in $P^{k}_{m}(\Omega)$. Let us introduce also the broken polynomial spaces

$$P^{k}_{m}(\mathcal{T}_{h}) := \left\{ v \in L^{2}(\Omega) \mid v|_{T} \in P^{k}_{m}(T) \ \forall T \in \mathcal{T}_{h} \right\},$$

which are special instances of the broken Sobolev spaces

$$H^{p}(\mathcal{T}_{h}) := \left\{ v \in L^{2}(\Omega) \mid v|_{T} \in H^{m}(T) \ \forall T \in \mathcal{T}_{h} \right\},$$

for $p \geq 1$.

**Table 2.1**: Dimensions of several polynomial spaces $P^{k}_{d}$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$d=1$</th>
<th>$d=2$</th>
<th>$d=3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

Generally, $\pi_{U}$ denotes the $L^{2}$-orthogonal projector onto the functional space $U$. Common usages will be: (i) $\pi^{k}_{F}$, the projector onto $P^{k}_{F}(T)$, (ii) $\pi^{k}_{h}$, the projector onto $P^{k}_{d}(\mathcal{T}_{h})$ such that for all $T \in \mathcal{T}_{h}$ and all $v \in L^{2}(\Omega)$, $\left(\pi^{k}_{h}\right)|_{T} := \pi^{k}_{F}v|_{T}$, and
(iii) $\pi^k_T$, the projector onto $P^k_{d-1}(F)$, the restriction to $F$ of $P^k_{d-1} \circ \Xi^{-1}$, $\Xi$ being an affine bijective mapping from $\mathbb{R}^{d-1}$ to the affine hyperplane supporting $F$.

Finally, let us introduce some results that will be useful thereafter. We recall the Poincaré-Wirtinger inequality:

There exists a real number $C_P \geq 0$ such that, for all $T \in \mathcal{T}_h$

$$
\left\|v - \pi^0_T v\right\|_T \leq C_P h_T \|\nabla v\|_T \quad \forall v \in H^1(T),
$$

and the discrete and continuous trace inequalities [24, Lemmata 1.46 and 1.49]:

There exist two real numbers $C_{tr}$ and $C_{tr,c}$ depending on $\gamma$ such that, for all $T \in \mathcal{T}_h$ and $F \in \mathcal{F}_T$

$$
\left\|v\right\|_F \leq C_{tr} h_T^{-\frac{3}{2}} \left\|v\right\|_T \quad \forall v \in P^k_d(T),
$$

$$
\left\|v\right\|_{\partial T} \leq C_{tr,c} (h_T^{-1} \left\|v\right\|^2_T + h_T \left\|\nabla v\right\|^2_T)^{-\frac{1}{2}} \quad \forall v \in H^1(T).
$$

Moreover, [24, Lemma 1.40], there exists a real number $C_{app}$ which depends on $\gamma$ and $k$ but not on $h$, such that what follows holds for all $T \in \mathcal{T}_h$: For all $s \in \{0, \ldots, k + 1\}$, and all $v \in H^s(T)$,

$$
\left\|v - \pi^k_T v\right\|_{H^m(T)} + h_T^{\frac{1}{2}} \left\|v - \pi^k_T v\right\|_{H^m(\partial T)} \leq C_{app} h_T^{-m} \left\|v\right\|_{H^s(T)} \quad \forall m \in \{0, \ldots, s - 1\}. \quad (2.8)
$$

### 2.1.4 Diffusion Tensor

For the sake of exposition, we will consider the diffusion tensor $\mathbb{M}$ to be piecewise constant on the partition $P_\Omega$, and thus for every mesh $\mathcal{T}_h$, too. We set $\mathbb{M}_T := \mathbb{M}|_T$ and we denote with $\mu_{\#,T}$ the smallest eigenvalue of $\mathbb{M}_T$ and with $\mu_0,T$ the greatest one. Their ratio will be referred to by $\rho_T := \frac{\mu_{\#,T}}{\mu_0,T} \geq 1$ (recall that the tensor is SPD). Extend the definitions to the whole mesh $\mathcal{T}_h$: $\rho_{\#} := \min_{T \in \mathcal{T}_h} \mu_{\#,T}$, $\rho_{\#} := \max_{T \in \mathcal{T}_h} \mu_{\#,T}$, $\rho_{\#} := \max_{T \in \mathcal{T}_h} \rho_T$. Finally, define $\mu_{T,F} := \mathbb{M}_T \mathbf{n}_{T,F} \cdot \mathbf{n}_{T,F}$; notice that, $\mathbf{n}_{T,F}$ being a unitary vector, $\mu_{0,T} \leq \mu_{T,F} \leq \mu_{\#,T}$.

Since the diffusion tensor is SPD, one could define its square root, $\mathbb{M}_T^{1/2}$ (which is still SPD), computable, for instance, by diagonalization:

$$
\mathbb{M}_T = Q D Q^{-1} = (Q D Q^{-1}) \left(\underbrace{Q D Q^{-1}}_{\mathbb{M}_T^{1/2}}\right)
$$

where $Q$ is a unitary matrix and $D$ is diagonal with positive elements.

For the sake of brevity, we will use $a \lesssim b$ meaning that there exists a constant $C$ independent of $h_T$ (hence, of $h$ neither) and of the diffusion tensor $\mathbb{M}$ but possibly on the mesh parameter $\gamma$ and on the polynomial degree $k$, such that $a \leq C b$. 


2.2 The HHO Method

2.2.1 Discrete Unknowns and the HHO\((l)\)-family

We are now approaching the core of the HHO method. In what follows, we consider an admissible mesh sequence \((\mathcal{T}_h)_{h \in H}\) and fix an integer \(k \geq 0\), which, together with \(l\), is one of the few independent parameters of the method: \(k\) and \(l\) denote, respectively, the degree of the face and cell discrete polynomial space in which the approximation of the solution of the problem will be sought.

**Remark 2.2.1** (The HHO\((l)\)-family) The HHO method was originally developed only for \(l = k\). More recently [19, 28] explored the possibility of having \(l\) taken in a wider range, \(l \in \{k - 1, k, k + 1\}\), thus defining three different methods, gathered into the so-called HHO\((l)\)-family. These choices allow one to make some comparisons between HHO and other methods: for instance, as it was pointed out in [19], up to minor modification in the choice of the stabilization, the method \(l = k - 1\) leads to the High-Order Mimetic method (HOM) introduced by [4, 39].

For \(T \in \mathcal{T}_h\), let
\[
U_T^k := \mathbb{P}_d^l(T), \quad \mathcal{U}_k^l := \mathbb{P}_{d-1}(F) \quad \forall F \in \mathcal{F}_T,
\]
where
\[
\tilde{l} := \max\{0, l\},
\]
then, the set of local hybrid unknowns is defined as
\[
\mathcal{U}_T^k := U_T^k \times \mathcal{U}_{\partial T}^k, \quad \mathcal{U}_{\partial T}^k := \bigtimes_{F \in \mathcal{F}_T} \mathcal{U}_F^k.
\]

Hence, a generic element \(\mathbf{v}_T\) of \(\mathcal{U}_T^k\) can be written also as
\[
\mathbf{v}_T := \left( v_T \in U_T^k, \; v_{\partial T} \in \mathcal{U}_{\partial T}^k \right),
\]
with \(v_{\partial T} := \left( v_F \in \mathcal{U}_F^k \right)_{F \in \mathcal{F}_T}\).

Let us also introduce the local reduction operator \(I_T^k : H^1(T) \rightarrow \mathcal{U}_T^k\) which maps a function into its projection onto the local discrete space:
\[
I_T^k v := \left( \pi_T v, \left( \pi_F^k v \right)_{F \in \mathcal{F}_T} \right) \quad \forall v \in H^1(T). \quad (2.9)
\]

Once that a local framework is available, working spaces can be defined on the whole mesh. In particular, let us set
\[
\mathcal{U}_h^k := U_h^k \times \mathcal{U}_h^k,
\]
where
\[
U_h^k := \bigtimes_{T \in \mathcal{T}_h} U_T^k, \quad \mathcal{U}_h^k := \bigtimes_{F \in \mathcal{F}_h} \mathcal{U}_F^k.
\]
Notice that the potential is single-valued on the faces, even the interfaces. Moreover, thanks to the definitions, $U_h^k = \mathbb{P}_d^k(T)$. In what follows, for a given $v_h \in U_h^k$, $v_h$ will denote its restriction to $U_h^k$, $v_h$ the restriction to $\Omega_h^k$ and $v_T = (v_T, v_{\partial T}) \in U_T^k$ its restriction to the cell $T \in \mathcal{T}_h$.

In the global framework, the boundary conditions can be taken into account in the definition on the working spaces:

$$U_{h,0}^k := U_h^k \times \Omega_{h,0}^k, \quad \Omega_{h,0}^k := \{v_h \in \Omega_h^k \mid v_F \equiv 0 \ \forall F \in \mathcal{F}_h\}.$$  

Analogously to what has been done in the local setting, the global reduction operator is defined: $I_h^k : U \to U_h^k$ such that

$$(I_h^kv)_{|T} := I_h^kv_{|T} \quad \forall v \in U, \ \forall T \in \mathcal{T}_h.$$  

Note 2.2.1 (Notation) For the sake of clarity, hybrid variables and sets will be denoted in underlined roman font (e.g $v_T$ and $U_T^k$), cell-related ones in standard roman font (e.g $v_T$ and $U_T^k$) and, finally, face-related ones in fraktur font (e.g $v_{F}$ and $U_F^k$).

![Figure 2.1: Degrees of Freedom of the polynomial spaces $\mathbb{P}_2^k(F)$ for the faces, circles, and of $\mathbb{P}_3^k$ for the cell, pentagons. $k \in \{0, 1\}$.](image)

### 2.2.2 Essential HHO Operators

#### 2.2.2.1 Potential Reconstructor

The main tool of the HHO method is the local potential reconstructor operator $p_T^{k+1}$. It is defined as follows:

**Definition 2.2.1** (Local potential reconstructor) Given a cell $T \in \mathcal{T}_h$, for all $v_T = (v_T, v_{\partial T}) \in U_T^k$, $p_T^{k+1} : U_T^k \to \mathbb{P}_d^{k+1}(T)$ is the solution of the following Neumann well-posed problem: For all $w \in \mathbb{P}_d^{k+1}(T)$

$$(\mathbb{M}_T \nabla p_T^{k+1} v_T, \nabla w)_T = -(v_T, \text{div}(\mathbb{M}_T \nabla w))_T + \sum_{F \in \mathcal{F}_T} (v_F, \mathbb{M}_T \nabla w \cdot n_{T,F})_F \quad (2.10)$$
or, equivalently, performing an integration by parts (IBP), for all \( w \in \mathbb{P}_d^{k+1}(T) \)

\[
(\mathbb{M}_T \nabla p_T^{k+1} \mathbb{I}_T, \nabla w)_T = (\nabla v_T, \mathbb{M}_T \nabla w)_T + \sum_{F \in \mathcal{F}_T} (v_F - v_T, \mathbb{M}_T \nabla w \cdot n_{T,F})_F. \quad (2.11)
\]

It is obvious that \( p_T^{k+1} \) is a solution of a Neumann problem and thus that it is defined up to a constant. In order to have a unique solution, a constraint needs to be added to the system; the equality of the space averages is then imposed: In addition to (2.10), \( p_T^{k+1} \) also satisfies

\[
\int_T p_T^{k+1} \mathbb{I}_T = \int_T v_T \quad \forall \mathbb{I}_T \in \mathbb{U}_T^k. \quad \tag{2.12}
\]

Remark 2.2.2 (Independence from \( l \)) It is stressed that the definition of the potential reconstructor does not depend on the value of \( l \): even if \( l \neq k \), the arrival space of \( p_T^{k+1} \) will always be \( \mathbb{P}_d^{k+1}(T) \), and analogously the test functions \( w \) in (2.10) will always lie in \( \mathbb{P}_d^{k+1}(T) \).

The combination of \( p_T^{k+1} \) and \( \mathbb{I}_T^k \) defines an elliptic projector onto \( \mathbb{P}_d^{k+1}(T) \), in the sense that \( p_T^{k+1} \mathbb{I}_T^k v \) is the only function in \( \mathbb{P}_d^{k+1}(T) \) minimizing a certain quantity. In fact,

Lemma 2.2.1 (Elliptic projector and polynomial consistency) For all \( v \in H^1(T) \)

\[
(\mathbb{M}_T \nabla (v - p_T^{k+1} \mathbb{I}_T^k v), \nabla w)_T = 0 \quad \forall w \in \mathbb{P}_d^{k+1}(T). \quad (2.13)
\]

Moreover, \( p_T^{k+1} \mathbb{I}_T^k \) satisfies

\[
p_T^{k+1} \mathbb{I}_T^k v = v \quad \forall v \in \mathbb{P}_d^{k+1}(T). \quad (2.14)
\]

Proof. Following [27], for \( v \in H^1(T) \), we plug the definition of \( \mathbb{I}_T^k \) (2.9) into (2.10), for all \( w \in \mathbb{P}_d^{k+1} \)

\[
(\mathbb{M}_T \nabla p_T^{k+1} \mathbb{I}_T^k v, \nabla w)_T = -(\pi_T^I v, \text{div}(\mathbb{M}_T \nabla w))_T + \sum_{F \in \mathcal{F}_T} (\pi_T^k v, \mathbb{M}_T \nabla w \cdot n_{T,F})_F
\]

\[
= -(v - (v - \pi_T^I v), \text{div}(\mathbb{M}_T \nabla w))_T
\]

\[
+ \sum_{F \in \mathcal{F}_T} (v - (v - \pi_T^k v), \mathbb{M}_T \nabla w \cdot n_{T,F})_F
\]

\[
= -(v, \text{div}(\mathbb{M}_T \nabla w))_T + \sum_{F \in \mathcal{F}_T} (v, \mathbb{M}_T \nabla w \cdot n_{T,F})_F
\]

where we could pass from the second to the third line since \( \text{div}(\mathbb{M}_T \nabla w) \in \mathbb{P}_d^I(T) \subset \mathbb{P}_d^{k+1}(T) \) and the term inside the brackets are orthogonal to \( \mathbb{P}_d^{k+1}(T) \) owing to the definition of \( \pi_T^I \). The same applies to the second part of the right-hand side since \( \mathbb{M}_T \nabla w |_F \cdot n_{T,F} \in \mathbb{P}_d^k(F) \). Now, integrating by parts and recalling that \( \mathbb{M}_T \) is symmetric,

\[
(\mathbb{M}_T \nabla p_T^{k+1} \mathbb{I}_T^k v, \nabla w)_T = (\mathbb{M}_T \nabla v, \nabla w)_T
\]
which proves (2.13).

The orthogonality condition which has just been proven, applied to any \( v \in \mathbb{P}_{d+1}^k(T) \), implies that \( (v - p_T^{k+1} L_T^k v) \) is constant (it belongs to \( \mathbb{P}_d^0(T) \)); moreover, considering the constraint (2.12), we infer \( \int_T p_T^{k+1} (L_T^k v) = \int_T \pi_T v = \int_T v \) which concludes the proof.

\[ \square \]

**Note 2.2.2 (Minor properties of \( p_T^{k+1} L_T^k \))** It is straightforward to conclude from (2.14) that

\[ \nabla p_T^{k+1} L_T^k v = \nabla v \quad \forall \ v \in \mathbb{P}_{d+1}^k(T). \] (2.15)

Notice also that \( p_T^{k+1} L_T^k \) acts like the identity operator when it is applied to a function in \( \mathbb{P}_{d+1}^k(T) \).

**Remark 2.2.3 (General case)** The results (2.13) and (2.14) are not valid in the more general case of \( M \) being only Lipschitz.

The results in Lemma 2.2.1 allow one to prove a first approximation property.

**Lemma 2.2.2 (Approximation properties)** For all \( v \in H^{k+2}(T) \), the following holds:

\[
\left\| v - p_T^{k+1} L_T^k v \right\|_T + h_T^\alpha \left\| v - p_T^{k+1} L_T^k v \right\|_{\partial T} \]

\[
\quad + h_T \left\| \nabla (v - p_T^{k+1} L_T^k v) \right\|_T + h_T^{\frac{\alpha}{2}} \left\| \nabla (v - p_T^{k+1} L_T^k v) \right\|_{\partial T} \lesssim p_T^{2\alpha} h_T^{k+2} \| v \|_{H^{k+2}(T)}. \] (2.16)

with \( \alpha = \frac{1}{2} \).

**Proof.** We follow [27] again. \( p_T^{k+1} L_T^k \) being a projector (2.13) means

\[
M_T^{1/2} \nabla (v - p_T^{k+1} L_T^k v) = \inf_{z \in \mathbb{P}_{d+1}^k(T)} \left\| M_T^{1/2} \nabla z - \nabla v \right\|_T \lesssim \mu_{\#_T} h_T^{k+1} \| v \|_{H^{k+2}(T)}
\] (2.17)

where the inequality is obtained via (2.8) with \( s = k + 1 \) and \( m = 1 \). Now, using the Young inequality,

\[
\left\| \nabla (v - p_T^{k+1} L_T^k v) \right\|_T \lesssim \mu_{\#_T}^{1/2} \left\| M_T^{1/2} \nabla (v - p_T^{k+1} L_T^k v) \right\|_T \lesssim \rho_T^\alpha h_T^{k+1} \| v \|_{H^{k+2}(T)}
\]

From this, using the Poincaré–Wirtinger inequality (2.5) (notice that \( v - p_T^{k+1} L_T^k v \) has zero average), we get

\[
\left\| v - p_T^{k+1} L_T^k v \right\|_T \lesssim \rho_T^2 h_T^{k+2} \| v \|_{H^{k+2}(T)}. \] (2.18)

We can now use, firstly, the continuous trace inequality (2.7) and then equations (2.17) and (2.18) to obtain

\[
h_T \left\| v - p_T^{k+1} L_T^k v \right\|_{\partial T}^2 \lesssim \left\| v - p_T^{k+1} L_T^k v \right\|_T^2 + h_T^2 \left\| \nabla (v - p_T^{k+1} L_T^k v) \right\|_T^2 \lesssim \rho_T^{2\alpha} h_T^{2(k+2)} \| v \|_{H^{k+2}(T)}^2.
\]
Concerning the bound on $\|\nabla(v - \rho^{k+1}_T L^k_T v)\|_{\partial T}$, add and subtract $\pi_T^f \nabla v$ and use the triangular inequality to get
\[
\|\nabla(v - \rho^{k+1}_T L^k_T v)\|_{\partial T} \leq \|\nabla v - \pi_T^f \nabla v\|_{\partial T} + \|\pi_T^f \nabla v - \nabla(\rho^{k+1}_T L^k_T v)\|_{\partial T}.
\]

Owing to (2.8) with $s = k + 1$ and $m = 0$, we get
\[
h_T^{\frac{1}{2}} N_1 \lesssim h_T^{k+1} \|\nabla v\|_{H^{k+1}(T)} \lesssim h_T^{k+1} \|v\|_{H^{k+2}(T)}.
\]

Considering now $N_2$,
\[
N_2 \lesssim \max_{F \in \mathcal{F}_T} \left\|\pi^f_T \nabla v - \nabla \rho^{k+1}_T L^k_T v\right\|_{F} \lesssim h_T^{-\frac{1}{2}} \left\|\pi^f_T \nabla v - \nabla \rho^{k+1}_T L^k_T v\right\|_{T}
\]
\[
\lesssim h_T^{-\frac{1}{2}} \left\|\pi^f_T \nabla v - \nabla \rho^{k+1}_T L^k_T v\right\|_{T} \lesssim h_T^{-\frac{1}{2}} \left\|\nabla(\rho^f_T v - \rho^{k+1}_T L^k_T v)\right\|_{T} \lesssim h_T^{-\frac{1}{2}} \|\nabla v - \nabla \rho^{k+1}_T L^k_T v\|_{T}.
\]

\[
\lesssim \mu_T h_T^{-2} h_T^{k+1} \|v\|_{H^{k+2}(T)}
\]

where, in the second-to-last line, we take advantage of $\nabla \rho^{k+1}_T L^k_T v \in [\mathbb{P}^k_d(T)]^d$ to conclude that $\left\|\pi^f_T \nabla v - \nabla \rho^{k+1}_T L^k_T v\right\|_{T} \leq \left\|\nabla v - \nabla \rho^{k+1}_T L^k_T v\right\|_{T}$. Finally, combine (2.19) and (2.20) to obtain
\[
h_T^{\frac{3}{2}} \|\nabla(v - \rho^{k+1}_T L^k_T v)\|_{\partial T} \lesssim \mu_T h_T^{k+2} \|v\|_{H^{k+2}(T)}.
\]

\[ \square \]

**Remark 2.2.4** If $M$ is only Lipschitz, the inequality is valid with $\alpha = 1$ [25, Lemma 2.1].

### 2.2.2.2 Stabilization

If we now tried to rewrite the weak formulation (2.2) in terms of the newly introduced local potential reconstructor, we would observe that the obtained bilinear form is not coercive, meaning that the problem is not well-posed. It is then necessary to add another part, a stabilization, to ensure the sought property. Before introducing this, we need another operator:
\[ q^{k+1}_T : U^k_T \rightarrow \mathbb{P}^{k+1}_d(T) \]

is defined so that, for all $w_T \in U^k_T$,
\[
q^{k+1}_T w_T := w_T + \left(\rho^{k+1}_T w_T - \pi^f_T \rho^{k+1}_T w_T\right).
\]

Basically, the operator $q^{k+1}_T$ enriches a function in $U^k_T$ by adding to it a higher order correction (observe that the term into the brackets lies in $\mathbb{P}^{k+1}_d$) computed thanks to the local potential reconstructor.

**Remark 2.2.5** One also may notice that this correction does not depend on the relation (2.12) involving the averages of the reconstruction and of the original function.
Now, we are able to introduce the stabilization term:

**Definition 2.2.2 (Stabilization bilinear form)** For all $T \in \mathcal{T}_h$, the stabilization bilinear form $j_T : \mathcal{U}_T^k \times \mathcal{U}_T^k \rightarrow \mathbb{R}$ is defined as follows:

$$j_T(u_T, v_T) := \sum_{F \in \mathcal{F}_T} \frac{\mu_{T,F}}{h_F} \left( \pi_F^k(q_T^{k+1}u_T - u_F, q_T^{k+1}v_T - v_F) \right)_F$$  

(2.22)

**Remark 2.2.6** In the case $\mu$ is only Lipschitz, in the definition here above, $\mu_{T,F}$ is replaced by

$$\mathcal{M}_F = \|\mu_{T,F} \cdot n_{T,F}\|_{L^\infty(F)}.$$

**Note 2.2.3** Notice that, if $l = k + 1$, $\pi_T^l p_T^{k+1} v_T = p_T^{k+1} v_T$, hence $q_T^{k+1} v_T = v_T$ and finally

$$j_T(u_T, v_T) = \sum_{F \in \mathcal{F}_T} \frac{\mu_{T,F}}{h_F} \left( \pi_F^k(u_T - v_T), \pi_F^k(v_T - v_T) \right)_F$$

which is a stabilization very similar to those used in the HDG methods, among which the method HHO($k + 1$) is basically considered [19].

The stabilization is a least-squares penalization of the of the residuals at faces coming from the $L^2$-projection of $q_T^{k+1}$ and the face-defined function; it should lead to the single-valuedness at the faces. Moreover, $j_T$ and, in particular, the role that $q_T^{k+1}$ plays in it, have been specifically designed in order to have the same approximation and consistency properties of $p_T^{k+1} l_T^k$. In fact,

**Lemma 2.2.3 (Approximation properties)** For all $v \in P_{d+1}^{k+1}(T)$:

$$j_T(l_T^k v, w_T) = 0 \quad \forall \ w_T \in \mathcal{U}_T^k$$  

(2.23)

Moreover, for all $v \in H^{k+2}(T)$

$$j_T(l_T^k v, l_T^k v)^{\frac{1}{2}} \lesssim \mu_{\#,T}^{\frac{1}{2}} p_T^{\frac{1}{2}} h_T^{k+1} \|v\|_{H^{k+2}(T)}$$  

(2.24)

**Proof.** The first property is pretty straight-forward. Following [27], consider the part of $j_T$ related to $v$, which, we recall, is in $P_{d+1}^{k+1}(T)$. Owing to the definition of $l_T^k$ and to (2.14), we get:

$$q_T^{k+1} v = \pi_T^l v - p_T^{k+1} l_T^k v + \pi_T^l p_T^{k+1} l_T^k v = \pi_T^l v - v - \pi_T^l v = v$$

Hence,

$$\pi_F^k(q_T^{k+1} l_T^k v - (l_T^k v)_F) = \pi_F^k(v - \pi_F^k v) = 0$$

which proves (2.23).

Consider now $v \in H^{k+2}(T)$.

$$j_T(l_T^k v, l_T^k v) = \sum_{F \in \mathcal{F}_T} \frac{\mu_{T,F}}{h_F} \left\| \pi_F^k((l_T^k v)_F - \pi_T^l p_T^{k+1} l_T^k v - (l_T^k v - \pi_T^l p_T^{k+1} l_T^k v)) \right\|^2_F$$

$$\lesssim \mu_{\#,T} \max_{F \in \mathcal{F}_T} h_F^{-1} \left\| \pi_F^k((l_T^k v)_F - \pi_T^l p_T^{k+1} l_T^k v - (l_T^k v - \pi_T^l p_T^{k+1} l_T^k v)) \right\|^2_F$$
For any \( F \), owing to the definitions of \( L_T^k \) and to the fact that, on \( F \), \( \pi_T^k \circ \pi_T^f = \pi_T^f \), the argument of the maximum in the inequality here above can be rewritten as follows:

\[
\begin{align*}
\|h_T^{-1} \| &\leq h_T^{-1} \| \pi_T^k (v - p_T^{k+1} L_T^k v) - \pi_T^f (v - p_T^{k+1} L_T^k v) \|^2_F \\
&\leq h_T^{-1} \| \pi_T^f (v - p_T^{k+1} L_T^k v) \|^2_F + h_T^{-1} \| \pi_T^f (v - p_T^{k+1} L_T^k v) \|^2_F \\
&\leq h_T^{-1} \| v - p_T^{k+1} L_T^k v \|^2_F + C_T^2 h_T^{-2} \| \pi_T^f (v - p_T^{k+1} L_T^k v) \|^2_F \\
&\lesssim \rho_T h_T^{2(k+1)} \| v \|^2_{H^{k+2}}
\end{align*}
\]

thanks to, in order, the trivial inequality \( 2(a^2 + b^2) \geq (a + b)^2 \), the definition of \( \pi_T^k \), equations (2.6), (2.4) and finally the approximation property of \( p_T^{k+1} L_T^k \) (2.16). Now, (2.24) is easily recovered. \( \square \)

### 2.2.3 Discrete Problem and its Properties

#### 2.2.3.1 Discrete Formulation

For all \( T \in \mathcal{T}_h \), let us introduce the local bilinear form \( a_T \):

\[
a_T : U_T^k \times U_T^k \to \mathbb{R} \quad a_T(u_T, v_T) := (M_T \nabla p_T^{k+1} u_T, \nabla p_T^{k+1} v_T)_T + j_T(u_T, v_T) \quad (2.25)
\]

with \( p_T^{k+1} \) and \( j_T \) defined in (2.10) and (2.22) respectively. We are now ready to define the global bilinear form

\[
a_h : U_h^k \times U_h^k \to \mathbb{R} ; \quad (u_h, v_h) \mapsto \sum_{T \in \mathcal{T}_h} a_T(u_T, v_T) \quad (2.26)
\]

Hence, the HHO version of (2.2) reads:

Find \( u_{h,0} \in U_{h,0}^k \) such that

\[
a_h(u_{h,0}, u_h) = (f, v_h) - a_h(u_{h,0}, v_h) + \sum_{F \in \mathcal{F}_h^e} (\phi_F, v_F)_F \quad \forall u_h \in U_{h,0}^k \quad (2.27)
\]

where \( u_{h,0} := L_h^k u_0 \in U_h^k \). At last, the discrete solution \( u_h \in U_h^k \) is computed:

\[
u_h = u_{h,0} + u_{h,0}.
\]

Moreover, let us define the local and global (semi-)norm deriving from the bilinear forms \( a_T \) and \( a_h \):

\[
\|u_T\|_{a,T} := a_T(u_T, u_T)^{\frac{1}{2}}, \\
\|u_h\|_{a,h} := a_h(u_h, u_h)^{\frac{1}{2}}.
\]

#### 2.2.3.2 Stability

We now claim that the method is stable with respect to the following \( H_0^1(\Omega) \)-like discrete (semi-)norm [28]:

\[
\|u_h\|_{l,h}^2 := \sum_{T \in \mathcal{T}_h} \rho_T^{-1} \|u_T\|^2_{u,T} \quad \forall u_h \in U_{h,0}^k \quad (2.28)
\]
where
\[
\|u_T\|^2_{U,T} := \|\nabla u_T\|^2_T + \|u_T\|^2_{U,T} \quad \forall u_T \in U_h^k.
\] (2.29)
\[
\|u_T\|^2_{U,T,\partial T} := \sum_{F \in \mathcal{T}_T} \frac{\mu_{T,F}}{h_F} \|u_T - \pi_F u_T\|^2_F
\]

Notice that (2.28) is a genuine norm since \(U_{h,0}^k\) embeds the homogeneous Dirichlet boundary condition, instead (2.29) is only a semi-norm on \(U_h^k\).

**Lemma 2.2.4 (Norm equivalence)** For all \(T \in \mathcal{T}_h\) and all \(u_T \in U_h^k\),
\[
\|u_T\|^2_{U,T} \lesssim a_T(u_T, u_T) \lesssim \rho_T \|u_T\|^2_{U,T}
\] (2.30)
Hence, for all \(u_h \in U_h^k\),
\[
\|u_h\|^2_{U,T} \lesssim a_h(u_h, u_h) \lesssim \rho_h \|u_h\|^2_{U,h}
\] (2.31)

**Proof.** First, let us state that, once that (2.30) is proven, a summation over \(T \in \mathcal{T}_h\) would yield directly (2.31).

We follow [25, Lemma 3] and its reference [27]. Let us address first the face-related terms. Consider the one in the definition of (2.29), add and subtract \(\pi_F^k q^{k+1}_T \tilde{u}_T\) inside \(\|\cdot\|_{U,T,\partial T}\), recall that, on \(F\), \(\pi_F^k \circ \pi_T^l = \pi_T^l\)
\[
\sum_{F \in \mathcal{T}_T} \frac{\mu_{T,F}}{h_F} \|v_F - v_T\|^2_F \lesssim \sum_{F \in \mathcal{T}_T} \frac{\mu_{T,F}}{h_F} \left( \|v_F - \pi_F^k q^{k+1}_T \tilde{u}_T\|^2_F + \|q^{k+1}_T \tilde{u}_T - \pi_T^k q^{k+1}_T \tilde{u}_T\|^2_F \right)
\]
\[
\lesssim j_T(u_T, u_T) + \mu_{\#T} \|\nabla q^{k+1}_T \tilde{u}_T\|^2_T
\]
\[
\lesssim j_T(u_T, u_T) + \rho_T \|M_T^{1/2} \nabla q^{k+1}_T \tilde{u}_T\|^2_T
\] (2.32)
where, in order to pass to the second line, we used (2.6) and (2.8) with \(s = 1\) and \(m = 0\). In order to obtain the inverse inequality, similarly to what we just did, in the stabilization term, add and subtract \(\pi_T^k v_T\) and use again (2.6) and (2.8), so that
\[
j_T(u_T, u_T) \lesssim \sum_{F \in \mathcal{T}_T} \frac{\mu_{T,F}}{h_F} \|v_F - v_T\|^2_F + \rho_T \|M_T^{1/2} \nabla q^{k+1}_T \tilde{u}_T\|^2_T
\] (2.33)

Now, we just need to compare \(\|M_T^{1/2} \nabla q^{k+1}_T \tilde{u}_T\|^2_T\) and \(\|M_T^{1/2} \nabla v_T\|^2_T\). Observe that, taking \(w = v_T\) in (2.11),
\[
\|M_T^{1/2} \nabla v_T\|^2_T = (M_T \nabla q^{k+1}_T \tilde{u}_T, \nabla v_T) - \sum_{F \in \mathcal{T}_T} (v_F - v_T, M_T \nabla v_T \cdot n_{T,F})_F
\]
\[
\leq (M_T \nabla q^{k+1}_T \tilde{u}_T, \nabla v_T) + \sum_{F \in \mathcal{T}_T} |(v_F - v_T, M_T \nabla v_T \cdot n_{T,F})_F| \] (2.34)
Consider the sum,
\[
A \leq \sum_{F \in \mathcal{F}_T} \left( \sqrt{\frac{h_T}{h_F}} \| v_F - v_T \|_F \right) \left( \frac{h_F}{\mu_{T,F}} \| M_T \nabla v_T \cdot n_{T,F} \|_F \right)
\]
apply Young’s inequality and notice that
\[
\sum_{F \in \mathcal{F}_T} h_F \left\| \frac{M_T}{\mu_{T,F}} \nabla v_T \cdot n_{T,F} \right\|_F \leq \sum_{F \in \mathcal{F}_T} h_F \frac{\left( M_T n_{T,F} \cdot n_{T,F} \right) \left| M_T \nabla v_T \cdot n_{T,F} \right|}{M_T n_{T,F} \cdot n_{T,F}} \lesssim \left\| M_{1/2} \nabla v_T \right\|_T^2
\]
where we have concluded thanks to the discrete trace inequality (2.6).

Now, in (2.34), use Cauchy-Schwarz and Young’s inequalities, as well as $\rho_T \geq 1$ and (2.32) to obtain
\[
\left\| M_{1/2} \nabla p_T \right\|_T^2 \lesssim \left\| p_{1/2} \nabla p_T^{k+1} \right\|_T^2 + \| U_T \|_{U,T} \lesssim \rho_T \left\| p_{1/2} \nabla p_T^{k+1} \right\|_T^2 + j_T(v_T, v_T)
\]
and combine again with (2.32) in order to have the left bound in (2.30).

Let us address now the second inequality. Take the equivalent definition of $\left\| \nabla p_T^{k+1} \right\|_T$, plug the definition of $\nabla p_T^{k+1}$ (2.11) in, and proceed as in (2.34) to obtain
\[
\left\| M_{1/2} \nabla p_T^{k+1} \right\|_T = \sup_{w \in \mathcal{P}_{k+1}(T)} \frac{(M_T \nabla p_T^{k+1} \cdot \nabla w)_T}{\left\| M_{1/2} \nabla w \right\|_T}
\]
\[
\lesssim \left\| M_{1/2} \nabla U_T \right\|_T + \sum_{F \in \mathcal{F}_T} \sqrt{\frac{h_T}{h_F}} \| v_F - v_T \|_F \quad (2.35)
\]
Now, combine (2.33) and (2.35) to get the second bound in (2.30), which concludes the proof.

**Remark 2.2.7 (General case)** If $M$ is only Lipschitz, the definition of the norm in (2.29) should be modified [25]:
\[
\| u \|_{U,T, *}^2 := \sum_{F \in \mathcal{F}_T} \frac{M_F}{h_F} \| v_F - v_T \|_F^2
\]
Nevertheless, bounds similar to those of Lemma 2.2.4 can be shown for this norm, too.

The results in Lemma 2.2.4 implies directly

**Corollary 2.2.1 (Well-posedness)** The HHO problem (2.27) is well-posed.
2.2.4 Error Estimates

Two are the main results in error analysis for the HHO method, one with respect to the norm $\| \cdot \|_{a,h}$ and the other with respect to the standard $L^2$-norm $\| \cdot \|_{L^2(\Omega)}$.

**Theorem 2.2.1** (Analysis-norm error estimate) Let $u \in U_0$ and $u_h \in U_{k,h,0}$ be the solutions to (2.1) and (2.27) respectively; suppose also that $u \in H^{k+2}(\Omega)$. Then, the following holds:

$$\|I_k u - u\|_{a,h}^2 \leq \|I_k u - u_h\|_{a,h}^2 \leq \sum_{T \in T_h} \mu_{\# T} r_T^{1+2\alpha} h_T^{2(k+1)} \|u\|^2_{H^{k+2}(\Omega)}$$

(2.36)

with $\alpha = \frac{1}{2}$.

**Remark 2.2.8** (General case) As in Lemma 2.2.2, $\alpha = 1$ if $\mathcal{M}$ is only Lipschitz.

**Proof.** We follow once again [27] and [25]. The first inequality has been already proved in (2.31) from which we can also derive (notice that $I_k u - u_h \in U_{k,h}$):

$$\|I_k u - u_h\|_{a,h} \leq a_h(I_k u - u_h, I_k u - u_h) \leq \sup_{v_h \in U_{k,h}} \|v_h\|_{a,h} = 1 a_h(I_k u - u_h, v_h).$$

(2.37)

$\mathcal{E}_h(v_h)$ is the consistency error and it can be rewritten as follows

$\mathcal{E}_h(v_h) = a_h(I_k u, v_h) - a_h(u_h, v_h)$

(2.27)

$\mathcal{E}_h(v_h) = a_h(I_k u, v_h) - a_h(u_h, v_h) - a_h(u_h, v_h) + \sum_{F \in F_h} (\phi_F, v_F)$

(2.38)

since, by definition, $u_h = u_{h,0} + u_{h,\partial}$. Moreover

$$\sum_{T \in T_h} (f, v_T) = \sum_{T \in T_h} \left(- \text{div}(\mathcal{M}_T \nabla u), v_T\right)$$

$$= \sum_{T \in T_h} \left(\mathcal{M}_T \nabla u, \nabla v_T\right) + \sum_{F \in F_T} (v_F - v_T, \mathcal{M}_T \nabla u \cdot n_T, F)$$

$$- \sum_{T \in T_h} \sum_{F \in F_T} (v_F, \mathcal{M}_T \nabla u \cdot n_{T,F}, F)$$

$$\mathcal{M} = \sum_{F \in F_h} \sum_{T \in T_f} (v_F, \mathcal{M}_T \nabla u \cdot n_F) =$$

$$= \sum_{F \in F_h} (v_F, \mathcal{M}_T \nabla u, n_{T,F}, F) + \sum_{F \in F^d} (v_F, \mathcal{M}_T \nabla u \cdot n_{T,F}, F) + \sum_{F \in F^r} (\phi_F, v_F)$$
where \( \nabla u \) is the jump at the interfaces which is null since the flux is continuous [24, Lemma 4.3] and where we have taken into account the boundary conditions of \( v_h \in U_{h,0}^k \). Recombining, we get:

\[
\sum_{T \in \mathcal{T}_h} (f, v_T)_T + \sum_{F \in \mathcal{F}_h^k} (\phi_\partial, v_F)_F \\
= \sum_{T \in \mathcal{T}_h} \left( (\mathcal{M}_T \nabla u, \nabla v_T)_T + \sum_{F \in \mathcal{F}_T} (v_F - v_T, \mathcal{M}_T \nabla u \cdot n_{T,F})_F \right)
\]

where, at the left-hand side, we have exactly the term in the square brackets in (2.38). Thus, using (2.10), (2.25) and (2.26) we obtain:

\[
E_h(v_h) = \sum_{T \in \mathcal{T}_h} (\mathcal{M}_T \nabla (p_{k+1}^T I_T^k u - u), \nabla v_T)_T \\
+ \sum_{T \in \mathcal{T}_h} \sum_{F \in \mathcal{F}_T} (v_F - v_T, \mathcal{M}_T \nabla (p_{k+1}^T I_T^k u - u) \cdot n_{T,F})_F + \sum_{T \in \mathcal{T}_h} j_T(I_T^k u, v_h)
=: E_1 + E_2 + E_3
\]

In order to bound \( E_1 \) and \( E_2 \) we use (2.16) and (2.31) to get

\[
|E_1 + E_2|^2 \lesssim \sum_{T \in \mathcal{T}_h} \mu_{\#} T \rho_T^{1+2\alpha} h_T^{2(k+1)} \|u\|^2_{H^{k+2}(\Omega)}.
\]

Concerning the bound on \( E_3 \)

\[
|E_3|^2 \lesssim \sum_{T \in \mathcal{T}_h} j_T(I_T^k u, I_T^k u)_T + j_T(v_T, v_T) \\
\lesssim \sum_{T \in \mathcal{T}_h} \mu_{\#} T \rho_T^{1+2\alpha} h_T^{2(k+1)} \|u\|^2_{H^{k+2}(\Omega)} \|v_T\|^2_{a,T}.
\]

The sought result (2.36) is obtained by combining (2.37), (2.39), (2.40) and (2.33).

Supposing a little more regularity on the solution \( u \) of the model problem (2.1), a bound on the \( L^2 \)-norm of the error can be demonstrated. Let us introduce the elliptic regularity: For all \( f \in L^2(\Omega) \), \( z \), the unique solution of (2.2) with homogeneous boundary conditions (both for Dirichlet and Neumann types), satisfies the following a priori estimate

\[
\|z\|_{H^2(\Omega)} \leq C_{\text{ell}} \rho^{-1} \|f\|_{L^2(\Omega)}
\]

for all \( f \in L^2(\Omega) \) with \( C_{\text{ell}} \) only depending on \( \Omega \). Then, it can be shown [28, Theorem 3.2]

**Theorem 2.2.2** (L^2-norm error estimate) Assume the elliptic regularity and suppose also that \( f \in H^{k+\epsilon}(\Omega) \) and \( \phi_\partial \in W^{k+\epsilon,\infty}(\Gamma_n) \), with \( \epsilon = 1 \) if \( k = 0 \) and
\( \epsilon = 0 \) otherwise. Then, under the assumptions of Theorem 2.2.1, the following holds

\[
\rho \| I_h u - \psi_h \|_{L^2(\Omega)} \lesssim \rho_{\#}^{1/2} \rho_h^{1/2+\alpha} h^{1/2} \left( \sum_{T \in T_h} \mu_{\#_T}^{1/2} \rho_T^{1+2\alpha} h_T^{2(k+1)} \| u \|_{H^k(\Omega)}^2 \right)^{1/2} \\
+ h^{k+2} \left( \| f \|_{H^{k+\epsilon}(\Omega)} + \| \phi \|_{W^{k+\epsilon,\infty}(\Gamma_n)} \right)
\]

### 2.2.5 Static Condensation

If one had a close look at the global problem (2.27), he would notice that the cell-related Degrees of Freedom (DoFs) of two different elements are actually uncoupled. One can then devise a static condensation technique that allows to locally eliminate the cell-DoFs, leading to a global system with only face-defined unknowns.

In fact, the HHO global problem (2.27) can be reformulated as follows [19, Section 2.5]:

\[
a_T ((u_T, 0), (v_T, 0)) = (f|_T, v_T)_T - a_T ((0, u_{\partial T}), (v_T, 0)) \quad \forall v_T \in U^h_T, \forall T \in T_h,
\]

\[
a_h(\psi_h, (0, v_F)) = \sum_{F \in F_h} (\phi, v_F)_F \quad \forall \psi_h \in U^h_{\partial,0}. \quad (2.41a)
\]

Notice that the problem has been split into \( \mathcal{T}_h \) local problems, which, in each cell \( T \in \mathcal{T}_h \), expresses \( u_T \) in terms of \( u_{\partial T} \) and \( f|_T \), and one global problem which will give the face DoFs.

**Note 2.2.4** A more thorough theoretic analysis of the static condensation for HHO can be found in [19] or in [28].
Chapter 3

A practical approach to HHO

In this second chapter, we are going to reformulate the main ideas of the HHO method in order to have a more exploitable framework. In the first part, we will show how the operators and bilinear forms are translated into an algebraic point of view, then we will outline the main concepts behind the actual implementation of the method.

3.1 HHO Algebraic Framework

3.1.1 Bases

3.1.1.1 Monomial Bases

As we have seen in the previous chapter, the HHO method hinges on the polynomial functional spaces $\mathbb{P}_{d}^{k}$, which, clearly have the nice property of being finite-dimensional, hence making them suitable for computer handling. However, they still leave some room for choices to the user, especially in the selection of their bases. The method itself is completely independent of the choice of the generating functions, in fact, one could notice that no mention to the subject has been made in the development of the theory. Nonetheless, picking a smart basis, with some peculiar properties, can save some work, especially when dealing with integration and resolution of linear system, even if of small dimension.

A naive approach would be to choose monomial basis functions. Take for instance $\mathbb{P}_{d}^{k}(T)$ (recall that $T \subset \mathbb{R}^{d}$), then its monomial basis is:

$$\mathcal{B}_{T}^{k} := \left\{ \phi_{\beta_{j}}^{T}(x) = \prod_{i=1}^{d} x_{i}^{\beta_{i}} \mid \forall \beta^{j} = (\beta_{i}^{j})_{i=1,...,d} \subset \mathbb{N}^{d} \text{ s.t. } \sum_{i=1}^{d} \beta_{i}^{j} \leq k \right\}.$$ 

Note 3.1.1 (Advantages and disadvantages) A first advantage of this kind of bases is that they are hierarchical, meaning that a basis of $\mathbb{P}_{d}^{k}$ can be easily
extended to one of $\mathbb{P}_d^{k+1}$ by simply adding the monomials of degree $k+1$: they are basically embedded one in the other, like Russian-dolls. Moreover, this type of bases does not need a lot of preparation or computation since they are pretty straight-forward, and they come really in handy when dealing with high degrees. In fact, once the lowest order functions are computed, the higher order ones are recovered without much effort by just multiplying the ones which have been already evaluated.

However, accordingly to the position of $T$ with respect to the origin of the axes, those functions can take very different values (compare the cases $T_1 = [0,1]^d$ and $T_2 = [10,11]^d$) which, eventually, could make the global linear system ill conditioned. With a little more effort, this drawback can be overcome by two means: (i) centering the monomial onto the barycenter of the element, $x_C$, and (ii) scaling it by dividing by a characteristic length of the element, $r_T$. For instance, starting from $\tilde{\varphi}_T^0(x) = x_1^2 x_2$, we obtain

$$x_1^2 x_2 \xrightarrow{\text{centering}} (x_1 - x_{C,1})^2 (x_2 - x_{C,2}) \xrightarrow{\text{scaling}} \left(\frac{x_1 - x_{C,1}}{r_T}\right)^2 \left(\frac{x_2 - x_{C,2}}{r_T}\right).$$

With these modifications, the image of every basis function of any element is included in a ball centered in 0 and with small radius, so that, two cells, which are far off one from the other, have comparable basis functions.

**Note 3.1.2** (Properties of the monomial basis) Notice that using the barycenter makes the first order monomial $L^2$-orthogonal to the constant basis function. Consider, for instance, $\tilde{\varphi}_0^T(x) \equiv 1$ and $\tilde{\varphi}_1^T(x) = x_1 - x_{C,1}$ ($x_C$ being the barycenter of the cell $T$), their scalar product reads

$$\int_T \tilde{\varphi}_0^T \tilde{\varphi}_1^T = \int_T (x_1 - x_{C,1}) = \int_T \left(x_1 - \frac{\int_T x_1}{|T|}\right) = 0.$$ 

Moreover, notice that, by scaling, we make the functions 'dimensionless' (we have divided a vector, $x$, by a length, $r_T$) preventing the basis to be dependent on the mesh size, $h$, thus making the method more solid against ill conditioning when refining.

So far, we have dealt only with cell basis functions; concerning the face ones, we will use the same ideas (hierarchical, scaled and centered monomials), but first, we need to set up a local, face-defined coordinate system. Consider a face $F$ and, for the sake of simplicity, let us take the dimension, $d = 3$. Choose two non-parallel unitary vectors lying on the face $F$, $x_{1,F}$ and $x_{2,F}$: those will be the two axes of the above-mentioned face coordinate system. Hence, the monomials of the basis function will be defined with respect to $x_{1,F}$ and $x_{2,F}$, meaning that the first coordinate of the face local system is computed by projecting a vector onto $x_{1,F}$ and analogously for the second coordinate and $x_{2,F}$. Thus, if we transpose into the face framework the function used in the example (3.1), we obtain

$$\tilde{\varphi}_F^F(x) = \left(\frac{(x - x_F) \cdot x_{1,F}}{r_F}\right)^2 \left(\frac{(x - x_F) \cdot x_{2,F}}{r_F}\right).$$
where \( x_F \) is the barycenter of the face and \( r_F \) the characteristic dimension of the face.

The properties described here above for the cell-monomials, are valid for the face bases, too.

**Remark 3.1.1 (Cell axes)** The same reasoning involving projections on some axes used for the face bases can be applied to the cell ones, too. Consider three linearly independent unitary vectors of \( \mathbb{R}^3 \), \( a_1, a_2, a_3 \). Then, the cell monomials generating the whole basis are

\[
\tilde{\phi}^T_i = \frac{(x - x_C) \cdot a_i}{r_T} \quad i = 1, 2, 3.
\]

One could easily see that the above-mentioned naive basis is obtained by choosing the \( a_i = e_i \), that is the canonical basis of \( \mathbb{R}^3 \).

**Note 3.1.3 (Notation)** From here on out, \( \mathcal{B}_E^k \) will denote a basis of \( \mathbb{P}_d(E) \) and \( \phi^E \) or \( \varphi^E \) a basis function belonging to \( \mathcal{B}_E^k \).

### 3.1.1.2 Tuning the Bases up

Even in this simple basis structure, there is still some freedom in the setup, especially when choosing the axes and the scaling factor.

As for the axes, since we are dealing with projections, it is better to choose vectors that are as far from parallel as possible, otherwise we would end up with a distorted local system, which, numerically, could have difficulties in spanning all the element, cell or face. For instance, for a face \( F \), one could consider all the vectors connecting the face barycenter and the mid-points of the edges, then an educated choice would be to select as axes the two most-orthogonal vectors, that is the two with the smallest scalar product, as in Figure 3.1.

Another choice could be to compute the **principal axes of inertia** of the element. Consider an element \( E \) with uniform density, compute the frame inertia matrix relative to the barycenter of the element, \( x_e \),

\[
[\tilde{M}_{EE}]_{ij} = \int_E (x_i - x_{e,i})(x_j - x_{e,j}) \quad i, j = 1, \ldots, \dim E
\]
then, by definition, the principal axes are the directions of the rotation that makes $M_{EE}$ diagonal, in simple terms, its eigenvectors. In fact, since $M_{EE}$ is a real symmetric matrix, there exists a unitary matrix, $Q$, and a diagonal one, $\Lambda$ such that $M_{EE} = Q \Lambda Q^T$, then, the principal axes are the columns of $Q$. Notice that, $Q$ being unitary, the principal axes can be chosen to be orthonormal. This kind of axes could be a smart choice because the first-order basis monomials generated with them would be $L^2$-orthogonal to each other. In fact, the mass matrix of those functions is proportional to the inertia matrix

$$\int_E \phi_i^E \phi_j^E = \frac{a_i \cdot a_j}{r_E^2} \int_E (x_i - x_{e,i})(x_j - x_{e,j}) = \delta_{ij} \frac{1}{r_E^2} \int_E (x_i - x_{e,i})(x_j - x_{e,j}).$$

This kind of basis will often be referred to as inertial.

One could also imagine that, given a set of initial vectors $\{\tilde{a}_0^i\}$, the results of their multiplication with the diffusion tensor, $\{\tilde{a} := M\tilde{a}_0^i\}$, could be beneficial because we are taking into account the parameters of the problems even in the functional setting, hence, making the resolution more easy. Of course, this procedure can be applied to the principal axes, but, in the case we are lead to perform an eigendecomposition, it would seem more considerable to compute the eigenvectors of the diffusion tensor and use them as axes of the bases. In fact, somehow, this could be equivalent to making the problem isotropic or, at least, orthotropic.

Different choices are possible also for the scaling factor, $r_E$. As it was mentioned, its aim is to make the basis functions dimensionless and to make their images fit into a 0-centered, small-radius ball, thus it should be a characteristic dimension of the element. An easy way could be to choose a power of the measure of $E$, for instance, the square root of the surface for a face $F$, or the cubic root of the volume for a cell $T$. This information is usually known, so it would be quite quick to recover $r_E$. Another non-expensive choice would be to take the longest edge of the element. Finally, one could consider the diameter of the element, defined as the largest distance between two vertices of the element. Provided that we use a monomial basis, this latter choice would lead to

$$\phi_j^E(E) \subset [0, 1] \quad \forall \phi_j^E \in B_k^E, \forall k \in \mathbb{N}, E \in (T_h \cup F_h),$$

but it would be quite expensive to compute.

### 3.1.2 The Algebraic Framework

In the view of the implementation, we associate to every HHO operator the algebraic equivalent, that is, a matrix. We are now going to show how these matrices are built.

Let us fix $k$ and $l$ (and, consequently, $\tilde{l}$) with the same meaning as above. Let us also choose: (i) for all $T \in T_h$, $B_{k+1}^T = \{\phi_i^T\}$, a set of basis functions of $P_d^{k+1}(T)$ and suppose that it contains the constant function $\phi_0^T \equiv 1$, and (ii) for all $F \in F_h$, $B_k^F = \{\varphi_i^F\}$, a set of basis functions of $P_{d-1}^k(F)$. Suppose also
Chapter 3. A practical approach to HHO

that, from $\mathcal{B}_G^{k+1}$, we can extract a subset of $\mathcal{B}_T^{k+1}$, basis of $\tilde{P}_l(T)$; moreover define $\mathcal{B}_G^{k+1} := \mathcal{B}_T^{k+1} \setminus \{\phi_0^T\}$ where we have taken off the constant function. In fact, as we will see in what follows, $\mathcal{B}_G^{k+1}$ will be used to recover the gradient of a function, thus one can drop the constant functions whose gradients are null. Finally, set: (i) $c := \text{card } \mathcal{B}_T$, (ii) $g := \text{card } \mathcal{B}_G^{k+1} = \text{card } \mathcal{B}_T^{k+1} - 1$, and (iii) $f := \text{card } \mathcal{B}_F^k$. Notice that they do not depend neither on $T$ nor on $F$, but only on $k$ and $l$.

Moreover, given a cell $T$, define $n_T := \text{card } \mathcal{F}_T$, the number of the faces of $T$ and set $t := c + n_T f$. Notice that $t = \dim U_T^k$.

3.1.2.1 Projection

The first operator we investigate is the $L^2$-orthogonal projector onto a polynomial space, say, for instance, $\tilde{P}_l(T)$. Computing the projection of a generic function $v \in H_1(T)$ onto this space means solving the following linear system

$$M_T \tilde{v} = \Phi_v$$

where

- $M_T \in \mathbb{R}^{c \times c}$ and $[M_T]_{ij} = \int_T \phi_i^T \phi_j^T$ for any $\phi_i^T, \phi_j^T \in \mathcal{B}_T^k$. $M_T$ is a mass matrix and thus it is SPD;
- $\Phi_v := \{\int_T v \phi_i^T\}_{i=1,...,c}$.

Up to minor modifications, the same applies for a face $F$ and $\tilde{P}_{d-1}(F)$; in particular, for a generic element $E$ in $T_h$ or in $F_h$, $M_{EE}$ refers to the mass matrix of the basis of $E$ computed as follows: $[M_{EE}]_{ij} = \int_T \phi_i^E \phi_j^E$ for any $\phi_i^E, \phi_j^E \in \mathcal{B}_E^k$.

3.1.2.2 Reconstructor Operator

Let us recall the definition of the potential reconstructor, (2.11) and (2.12):

$$(M_T \nabla p_T^{k+1} \mathcal{U}_T, \nabla w)_T = (\nabla v_T, M_T \nabla w)_T + \sum_{F \in \mathcal{F}_T} (v_F - v_T, M_T \nabla w \cdot n_{T,F})_F
$$

$$\int_T p_T^{k+1} \mathcal{U}_T = \int_T v_T \quad \forall \mathcal{U}_T \in U_T^k.$$ 

Notice that, their gradients being null, the constant basis functions are used only in the second equality. Given that in the HHO weak formulation $a_T(\cdot, \cdot)$ the consistency term only involves gradients, and that the stabilization term does not depend on (2.12), Remark 2.2.5, at this stage of the method, the constant basis functions of the cell polynomial space can be temporarily dropped. Therefore, the appropriate basis used in order to recover the reconstructor is $\mathcal{B}_G^{k+1} = \mathcal{B}_G^{k+1} \setminus \{\phi_0^T\}$. Hence, the equation imposing the equivalence of the averages can be postponed to a later step.

Now, from an algebraic standpoint, solving the problem (2.11) means inverting a linear system of the form:

$$M_G G_T = R_G := [R_{F_0} | \cdots | R_{F_n}^T | R_T]$$

(3.3)
where the symbol $|$ denotes a juxtaposition of two matrices.

Let us have a look at the matrices more in detail.

- $M_G \in \mathbb{R}^{g \times g}$ is a sort of stiffness matrix of $\mathcal{B}_G^{k+1}$. In fact, $[M_G]_{ij} = \int_T \nabla \phi_i^T \cdot \nabla \phi_j^T \forall \phi_i^T, \phi_j^T \in \mathcal{B}_G^{k+1};$

- $R_T \in \mathbb{R}^{g \times c}$ and $[R_T]_{ij} = \int_T \nabla \phi_i^T \cdot \nabla \phi_j^T - \sum_{F \in \mathcal{F}_T} \int_F \nabla \phi_i^T \cdot n_{T,F} \phi_j^T$ for all $\phi_i^T \in \mathcal{B}_G^{k+1}$, and all $\phi_j^T \in \mathcal{B}_T^j$.

Finally, for all $R_{F_i} \in \mathcal{F}_T$, the elements of $R_{F_i} \in \mathbb{R}^{g \times l}$ are: $[R_{F_i}]_{ij} = \int_{F_i} \nabla \phi_i^T \cdot n_{C,F_i} \varphi_{F_i}^j$, for all $\phi_i^T \in \mathcal{B}_G^{k+1}$ and all $\varphi_{F_i}^j \in \mathcal{B}_{F_i}^j$.

**Remark 3.1.2** $(G_T)$ Notice that even the right-hand side of (3.3) is a matrix so that the solution $G_T$ is a matrix too, in particular $G_T \in \mathbb{R}^{g \times l}$. This is legit since $G_T$ is the algebraic translation of the operator $\nabla p_{d+1}^{k+1}$ whose image is $\nabla p_{d+1}^{k+1} (T) \subset [\mathcal{P}_d^k(T)]^d$. In particular, for $u \in H^1(T)$

$$\nabla u(x) \approx \sum_{\phi_i^T \in \mathcal{B}_G^{k+1}} [G_T I_T^k u], \nabla \phi_i^T.$$ 

**Remark 3.1.3** (Cost of the inversion) If one chooses not to take into account the constant functions, the matrix $M_G$ is SPD (recall that it is a stiffness matrix), which means that it could be inverted with the Cholesky factorization, costing only $O(\frac{4}{3}g^3)$ operations. In the other case, the equality of the averages is usually enforced with a Lagrange multiplier and the matrix is only Symmetric Positive-Semidefinite, and thus the solution can cost more.

Now, let us see how this operator is used in the local system. Consider $u, v \in \mathcal{P}_d^k(T)$ and their discrete polynomial projection $\tilde{u}, \tilde{v} \in \mathbb{R}^c$. Then, the consistency part in $a_T(u, v)$ (2.25) can be rewritten as follows

$$(\mathcal{M}_T^{\frac{1}{2}} \nabla u, \mathcal{M}_T^{\frac{1}{2}} \nabla v)_T = (M_G^{-1} R_G \tilde{u})^T M_G (M_G^{-1} R_G \tilde{v})$$

$$= \tilde{u}^T R_T (M_G^{-T} M_G) (M_G^{-1} R_G) \tilde{v}$$

$$= \tilde{u}^T R_T G_T \tilde{v}$$

where we used the definition of $G_T$. Hence, the matrix that will appear in the local system is $R_T^T G_T$.

### 3.1.2.3 Stabilization Operator

Only the stabilization is left to build. Let us first reorganize the terms in the face scalar product by changing the sign and by expanding $q_T^{k+1}$, so that

$$j_T (\tilde{u}_T, \tilde{v}_T) = \sum_{F \in \mathcal{F}_T} \frac{h_F}{\mu_F} (\pi_F(u_T + p_T^{k+1} u_T - p_T^{k+1} u_T) - u_F, \pi_F(v_T + q_T^{k+1} v_T - q_T^{k+1} v_T) - v_F)_F.$$
We can now read better the algebraic equivalent, which is governed by

\[ J_T = \sum_{F \in \mathcal{F}_T} \frac{\mu_{T,F}}{h_F} B^T_F \, M_{FF} \, B_F \]  

(3.4)

with

\[ B_F = M^{-1}_{FF}(M_{FT}(1_c - M^{-1}_{TT}M_{TG}G_T) + M_{FG}G_T) - 1_c. \]

Let us have a look at the matrix.

- \( M_{FT} \in \mathbb{R}^{f \times c} \) is the mass matrix collecting the scalar products of the face- and cell-DoFs: \([M_{FT}]_{ij} = \int_F \varphi_F^i \phi_T^j \) for all \( \varphi_F^i \in \mathcal{B}_F^k \) and \( \phi_T^j \in \mathcal{B}_T^l \);

- \( M_{FG} \in \mathbb{R}^{f \times g} \) is defined analogously to \( M_{FT} \) with the only difference that \( \phi_T^j \in \mathcal{B}_G^{k+1} \);

- \( M_{TG} \in \mathbb{R}^{c \times q} \) is a sort of mass matrix between \( \mathcal{B}_T^l \) and \( \mathcal{B}_G^{k+1} \), so that \([M_{TG}]_{ij} = \int_T \phi_T^i \phi_G^j \) for all \( \phi_T^i \in \mathcal{B}_T^l \) and \( \phi_G^j \in \mathcal{B}_G^{k+1} \);

- \( 1_c, 1_f \) are the identity matrices of dimension \( c = \dim \mathbb{P}_d(T) \) and \( f = \dim \mathbb{P}_{d-1}(F) \), respectively. Notice that the matrices which they are added to are rectangular, so, when summing, we actually mean adding those identity matrices to the sub-matrices related to the DoFs associated to a cell \( T \) or to a specific face \( F \).

### 3.1.2.4 The Local System

Let us now define the right-hand side of the local system,

\[ b_{loc,T} := [b_T^T | b_{T,F}^T]^T \]

(3.5)

where we have distinguished between cell- and face-related DoFs. Referencing (2.27), the procedures to take into account the load \( f \) and the Neumann boundary condition \( \phi_\partial \) are basically the same, that is the \( L^2 \)-scalar product with respect to the basis functions. Thus, for a cell \( T \)

\[ [b_T]_i = \int_T f \phi_T^i \phi_T^i \in \mathcal{B}_T^f \]

and analogously for a face \( F \in \mathcal{F}_h^n \)

\[ [b_F]_i = \int_F \phi_\partial \varphi_F^i \varphi_F^i \in \mathcal{B}_G^k. \]

Ignoring temporarily the Dirichlet boundary conditions, which will be dealt with later in this chapter, the local system is now ready. In fact, its stiffness matrix, translation of the bilinear form \( a_T(\cdot, \cdot) \), reads:

\[ \mathbb{R}^{t \times t} \ni A_T = R_T^G M_G + J_T. \]

hence, the whole system is

\[ A_T u_{loc,T} = b_{loc,T}, \]  

(3.6)

where \( u_{loc,T} \) is the local solution, which can be subdivided in cell and face DoFs just like \( b_{loc,T} \) in (3.5).
3.1.2.5 Static Condensation

As we have seen in the previous chapter, not the whole matrix $A_T$ is assembled in the global system: in fact, a static condensation can be performed locally. More specifically, in (3.6), we can distinguish between cell- and face-related DoFs as follows:

$$
\begin{array}{c}
c \\
\{ \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\} \\n\end{array}
$$

where the braces show the dimensions of the submatrices. Also, notice that $A_{TF} = A_{FT}^T$. Now, rewriting the two sub-systems for $u_T$ and $u_{T,F}$, we obtain

$$
\begin{align*}
\mathbf{u}_T &= \tilde{u}_T - A_{TF}^{-1} A_{FT} \mathbf{u}_{T,F} \\
(A_{FF} - A_{TF} A_{FT}^{-1} A_{FT}) \mathbf{u}_{T,F} &= \mathbf{b}_{T,F} - A_{FT} A_{FT}^{-1} \mathbf{b}_T .
\end{align*}
$$

Comparing now equation (2.41) and (3.8), we can notice how (3.8a) is exactly the algebraic translation of (2.41a).

As it was already pointed out, only $\tilde{A}_{FF}$ and $\tilde{b}_{T,F}$ will be assembled in the global system. Then, once $\mathbf{u}_{T,F}$ is extracted from the global solution $\mathbf{u}_{h,F}$, $\mathbf{u}_T$ is recovered using (3.8a).

The main advantage of the static condensation is that it allows to reduce the dimension of the global system by (temporarily) eliminating the cell-DoFs. However, there is a cost to pay, that is, the inversion of $A_{TT}$ (3.8b). Nevertheless, this overhead is certainly smaller than the delta which we would have had without the static condensation: in fact, consider that the dimension of $A_{TT}$ is $c = \dim P^k_d(T) \leq 10$ even when $k = 2$ and $l \leq k$, hence, the inversion can be performed without too much effort.

3.1.2.6 The $(k + 1)$–th Order Reconstruction, $p_{T}^{k+1}$

Suppose that, after the resolution of the global system, also the inverse of the static condensation (3.8a) has been performed for every cell, then, one has access to

- for each cell $T$, an approximation of the solution which lies in $P^k_d(T)$: $\tilde{u}_T^k(x) := \sum_{\phi_i \in B_T^k} [u_T]^i \phi_i^T (x)$;
- for each face $F$, an approximation of the solution in $P^k_{d-1}(F)$: $\tilde{u}_F^k(x) := \sum_{\phi_i \in B_F^k} [u_{T,F}]^i \phi_i^F (x)$.
However, with a little more effort one can reconstruct $p_T^{k+1} \mathbb{1}_T u_T \in \mathbb{P}_d^{k+1}(T)$.

Let us recall what we have seen in subsubsection 3.1.2.2, define
\[ g := G_T u_{\text{loc},T} \quad \text{then} \quad \sum_{\phi^T \in \mathbb{B}_G^{k+1}} [g]_i \nabla \phi^T_i(x) \approx \nabla u_T(x). \]

We could imagine that the coefficients $[g]_i$ are enough to recover the approximation of the solution in $\mathbb{P}_d^{k+1}(T)$, but that is not the case. In fact, $g$, since it is computed from $G_T$, describes the gradient and it does not carry any information about the average of the function in the cell (recall that we have excluded the constant functions). However, we are only left to find the right coefficient $\tilde{u}_T^{k+1}$ related to the constant basis function $\phi^T_0$, then
\[ \tilde{u}_T^{k+1}(x) = \tilde{u}_{T,0} + \sum_{\phi^T \in \mathbb{B}_G^{k+1}} [g]_i \phi^T_i(x) \]
is the $(k + 1)$-th order polynomial approximation of the solution that we are looking for. The idea is to use $\tilde{u}_T^T$ because it is described by $\mathbb{B}_T^T$ which does include the constant functions. Thus, let $\bar{f}^T$ denote the average of $f$ over the cell $T$, then, of course,
\[ \bar{w}_T^T = \bar{u}_T^T = \bar{u}_T^{k+1} \]

hence, expanding the second equality and solving for $\bar{w}_{T,0}$
\[ \bar{w}_{T,0}^{k+1} = [w_T]_0 + \frac{1}{\phi_0^T} \left( \sum_{i=1}^{c-1} ([w_T]_i - [g]_i) \phi^T_i - \sum_{i=c}^g [g]_i \phi^T_i \right). \]

Notice that the averages $\bar{\phi}_i^T$ are easily recovered from the first row of $M_{TG}$.

Of course, this operation would have not been necessary if in the computation of $p_T^{k+1}$ we had kept the constant functions.
3.2 Implementation

In this section, we will discuss some numerical and algorithmic choices made during the implementation. The code was developed within Code_Saturne \[33\], an open-source, C-based multi-purpose CFD software developed by EDF R&D. Working within this wide, well-established environment has its advantages, of course: many useful routines were already coded, especially those dealing with the mesh reading and the linear solvers, which eased our work considerably.

It is also important to say that only the 3D case has been considered, so that, for instance, the local HHO functional space setting for a cell $T$ is

$$U_T^k = P_3^l(T) \times \bigotimes_{F \in F_T} P_2^k(F)$$

### 3.2.1 Choice of the Order of the Method

We would like to stress here that, in view of an industrial, quick-and-dirty use, we put from the beginning a bound to the degree,

$$0 \leq k \leq 2,$$

which, however, ensures orders of convergence up to the fourth. The reasons behind this choice are multiple.

First, we tried to keep the total computational time comparable to those of lowest-order methods, such as CDO or the MHFV. Consider the integrals that one has to compute; when the order gets higher, one should use quadrature rules more and more precise, hence, more and more time-consuming. Moreover, as one could infer from Table 3.1, the dimension of the local system grows fast, a fact that, taking into account all the matrix operations and inversions which are needed, leads to slower computation times.

High orders might also be more difficult to handle from a numerical standpoint. One should be careful with the rounding error which could impact the order of convergence: results in [27, Section 4] showed that round-off errors were responsible for the under-convergence measured in 2D even for $k = 4$ and for Cartesian meshes.

Finally, in order to have the theoretic orders of convergence, quite demanding assumptions about the smoothness of the solution have to be made, hypotheses which are often not attended.

Once that the order $k$ is fixed, the cell polynomial order $l$ is left to be chosen. With similar concerns about the performances and local dimensions, we have easily excluded the choice $l = k + 1$. We hesitated between the other two cases, but we finally decided that the choice $l = k - 1$ would have not brought in a significant gain performance-wise: it is true that the smaller dimension could have sped up some operations, but, recall that, in any case, the arrival space of $p_{T}^{k+1}$ is $P_{d+1}(T)$. Thus, we settled for $l = k$. 
Table 3.1: Dimensions of the gradient basis and of the HHO local system of a hexahedron and of a tetrahedron for different values of $k$ ($l = k$)

<table>
<thead>
<tr>
<th>$k$</th>
<th>$g = B_G^{k+1}$</th>
<th>$\dim U_{\pi}^{k,\text{hexa}}$</th>
<th>$\dim U_{\pi}^{k,\text{tetra}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>46</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>80</td>
<td>60</td>
</tr>
</tbody>
</table>

### 3.2.2 Computing the Principal Axes of the Basis

In the previous section, we discussed about the possibility of using principal axes as generators of the basis. We recall that the aim of this procedure was to end up with diagonal mass matrices, or, at least, quite sparse and with extra-diagonal terms of a couple of orders of magnitude smaller than the diagonal elements. It is unnecessary to say that this kind of matrices are easily handled and that they improve the results performance- and precision-wise. The computation of the principal axes is achieved by a diagonalization.

Since we are interested in the eigenvectors and not in the eigenvalues, even just an approximation of the mass matrix suffices. For instance, in 2D, there is no need to compute the Jacobian of the transformation which allows the change of the integrating variables from the local/face system to the global/volumetric system. In fact, for a regular face, the Jacobian would be constant and it would impact only the eigenvalues and not the eigenvectors.

Actually, even an approximation of the eigenvectors would be enough for our needs. What one would obtain in this case would be an almost diagonal matrix, as described here above. Thus, there is no need for our diagonalization algorithms to be very precise.

### 3.2.3 Dirichlet Boundary Conditions

In order to deal with Dirichlet boundary conditions, homogeneous or not, it was decided to set up a penalization procedure. In particular, it will be performed locally, even before the static condensation.

Fix the penalty parameter $\eta \gg 1$, typically $\eta \propto 10^{10-12}$. Referencing (3.7), if $F_\partial \in \mathcal{F}_h^d$, it suffices to add $\eta \mathbf{1}_f$ (the identity matrix of dimension the cardinality of $P_k^d(F_\partial)$) to the sub-matrix of $A_{FF}$ related to the concerned face and to add $\eta \tilde{\psi}$ to $b_{F_\partial}$, where $\tilde{\psi}$ is the projection of the Dirichlet boundary condition $\psi_\partial$ onto $P_k^d(F_\partial)$ computed as in (3.2).

$$
\tilde{\psi} = M_{F_\partial F_\partial}^{-1} \left[ \int_F \varphi_i F_\partial \psi_\partial \right]
$$

$$
b_{F_\partial} + = \eta \tilde{\psi}
$$

$$
A_{F_\partial F_\partial} + = \eta \mathbf{1}_f
$$

**Note 3.2.1** (The homogeneous case) For homogeneous Dirichlet boundary con-
ditions, the righ-hand side vector is forced to zero:
\[ \mathbf{b}_{F_{\partial}} = 0. \]

**Note 3.2.2 (Difference with the Neumann BC)** Notice that, differently from the Neumann conditions or the source term, here we have to perform an inversion after the scalar product: in fact, we want to impose the real solution for the \( F_{\partial} \)-related DoFs, thus the projection need to be computed.

### 3.2.4 Integrating

As it was presented in the previous section, the key tool for building the matrices is the Gaussian integration. *Code_Saturne* has several quadrature rules, exact for polynomials up to the fifth order in dimension 1, 2 and 3, but, unfortunately, they can be applied only to the simplexes [34]. Subdividing a general element into simplexes can easily overcome the problem. For instance, take a general face with more than three edges. Consider one of them and the triangle formed with the center of the face: now one can use the quadrature. Repeat for every edge and the sum of the results is the sought integral. The same procedure can be applied for a volumetric integral by considering the tetrahedra defined by the vertices of an edge, the center of the face to which the edge belongs and the center of the cell.

This procedure is quite costly, especially when high order quadrature rules with large stencils are applied. In fact, it requires to integrate over \( 2 \times \# \text{edges} \) sub-elements, for instance, 24 sub-tetrahedra for a cube. There are strategies to divide the elements in a smarter way: for instance, consider a square, one of its diagonals divides it into 2 triangles, which means a great gain compared to the 4 triangles produced with the naive procedure described here above. Analogously, for a general face, consider two consecutive edges and join the two non-common vertices to have a triangle.

Unfortunately, without a priori hypotheses about the regularity of the face, those procedures can fail: consider for example a concave face as in Figure 3.3. Thus, since we do know which type of element we are dealing with, the best that we could do was to implement a check on the type which switches to a one-shot computation of the integral if the considered element is a simplex. Whenever we encounter a triangular face, another little optimization is to consider only one sub-tetrahedron (the one formed by the three vertices and the cell center) instead of the typical three. However, the naive procedure allows one to handle well the unstructured meshes and to be independent of the numbering of the vertices/edges.

A final remark concerns the exactitude of the available quadrature rules which can integrate polynomials up to the fifth order. This is enough to correctly build the matrices for our framework: in fact, worst case scenario, \( k = 2 \) and the maximum order of the polynomials which we are lead to integrate is five (in \( M_{TG} \) and \( M_{FG} \)).

Thus, at the beginning of the computations, the most compact quadrature rule, which is also exact at least for all the integrals demanded by the order \( k \),
is chosen: this allows one to speed up the computations for the lowest orders, \( k = 0 \), for instance. As for the source term and the boundary conditions, since there is no a priori knowledge of the functions, the highest order (up to 5th order polynomials) quadrature rule is used.

### 3.2.4.1 Matrix Filling

Not every element of all the matrices (recall that they are integrals) needs to be computed. Take, for instance, \( M_{FF} \), it is symmetric, so once the upper triangular part is computed, you can just copy it into the lower triangular part. Or notice that \( M_{TT} \) is just a sub-matrix of \( M_{TG} \). In the end, not many matrices have to be computed entirely. This filling procedure can save a lot of computations and can go even further if we make some hypotheses about the basis. Let us discuss this matter in details.

Consider a hierarchical monomial cell basis with orthonormal axes, say, for the sake of simplicity, the canonical ones, \( \{e_1, e_2, e_3\} \). Then, up to multiplications and sums, even the elements of \( M_G \) are already computed in \( M_{TG} \) (or \( M_{TT} \)). For instance, consider the basis function \( \phi_6^T = \frac{x y}{r_T^2} \), where we have supposed, for the sake of simplicity, that the cell center is 0, then

\[
\int_T \nabla \phi_6^T \cdot \nabla \phi_6^T = \frac{1}{r_T^2} \int_T \left( \frac{x}{r_T} \frac{x}{r_T} + \frac{y}{r_T} \frac{y}{r_T} \right) = \frac{1}{r_T^2} \left( \int_T (\phi_1^T)^2 + \int_T (\phi_2^T)^2 \right) \tag{3.9}
\]

where \( \phi_1^T = \frac{x}{r_T} \) and \( \phi_2^T = \frac{y}{r_T} \): one can now see that the integrals here above are the same as in \( M_{TT} \). Hence, even \( M_G \), which is the gradient mass matrix, can be filled with values coming from the cell mass matrix. Unfortunately, this procedure is highly dependent on the numbering of the functions within the basis, thus, it can be formed only by manually hard-coding the equivalences between matrix elements as the one in (3.9).

However, with orthonormal axes, low orders and iso/orthotropic diffusion, the filling procedure could be achieved without too much effort and in a way that allows the code to be quite maintainable. Without those hypotheses, the readability of the code would be heavily altered and the debugging process hardly possible.
Those are the reasons why two versions of the code are have available:

- The **optimized** version works with a hierarchical monomial basis with orthonormal axes, at least for the cell basis. It takes advantage as much as possible of this matrix filling procedure and it presents routines specifically optimized for the chosen order: for instance, if \( k = 0 \), the cell part of the stabilization term reduces to a value only, and thus many matrix products can be avoided;

- The **general** version works with any type of bases (so far, however, only the monomial one have been implemented) and any order \( k \). Almost all the matrices are built individually: except for the symmetrix matrices, every single integral is explicitly computed.

The version that will be integrated in *Code_Saturne* and that we have used to compute the results shown in the next chapter is the general one, because it is valid in all the situations. However, the optimized version shows building times as much as two times smaller than the general one.

```plaintext
foreach Cell T do
    Compute volumetric integrals;
    Matrix filling;
    foreach Face F do
        Compute surface integrals;
        Matrix filling;
    endforeach
    Solve for reconstructor;
    Compute stabilization;
    Compute boundary conditions;
    Compute source term;
    Assemble local problem;
    Static condensation;
    Assemble into global problem;
end
Solve global problem;
Post processing;
```

**Algorithm 1**: Steps of the HHO algorithm.

### 3.2.5 Data Structures and Optimizations

Now that we are investigating the implementation aspects more in depth, it is useful to recall that the method was coded in C: sometimes this was a restriction and it forced us to make certain choices and to be always careful at the implementation and optimization.
We found ourselves in the situation of having to implement our own matricial library. Thus, we settled on a simple structure for describing a matrix: a row-major vector of values and two integers for the numbers of rows and columns. Conscious of the limits of such a basic structure, we have been careful when dealing with these matrices. For instance, instead of the classical row-column matrix product, we compute a row-row product which better exploits the row-majorness of the matrices and which gives the same result of the classical one if the second matrix is symmetric or if we use its transposed, instead. The transposed forms were also used in the inversions. In fact, since the right-hand side is often a matrix, too, we are lead to extract one column at time. Given the row-majorness, this is not a cache-friendly operation, thus, it was preferred passing the transposed of the right-hand side to avoid cache-misses. Moreover, the returned matrix is often the transposed of the actual solution, because, in what would have followed, it would have been used in matrix products or other inversions. Always concerning the resolution of linear systems, different methods have been implemented and the user can choose the one to use: in addition to the default and faster Cholesky-LDLT solver (recall that we always deal with SPD matrices), also the LU and its companion PLU are available.

The structure lodging the polynomial bases has been conceived to mimic both the public/private and the inheritance paradigms of C++. Both for the cell and face bases, the public-access members are the size of the basis and a pointer to the builder function and one to the routine which, given a point, evaluates all the basis functions. For the cell basis, also the size of the gradient and the function that returns the gradients of the basis functions are accessible. Those elements are always built. However, the core which contains the real definition of the basis, the so-called basis builder, is private: in fact, it is defined in the implementation file, hence, its members are accessible only to function defined in the same file. Not all its members are built every time: accordingly to the requested type of basis, some of them are allocated and others left unbuilt. For instance, the cell builder contains the pointers to the matrices used in the eigenvector algorithm but they are allocated only if the inertial type is required. Of course, also the pointers to the evaluation and the gradient functions are set accordingly to the basis type. Another reason behind a private basis builder is the need to store sensitive quantities, such as the scaling factor or the axes of the basis: of course, having access to something as the C++ lambda functions would have made all this more easy.

Giving large attention to performance concerns, it is a policy of Code_Saturne to avoid possibly time-consuming mallocs and frees as much as possible. Thus, consider an object which is cell-specific, as the basis can be. Instead of creating and destroying one for each iteration of algorithm 1, the members are allocated before the cell-loop, then they undergo a set-up which loads the information about the current element, and, finally, they are free’d once the loop is finished: this is the typical lifespan of that kind of objects. Often, following this idea about avoiding unnecessary de-allocations, those objects have members which only act
as buffers, such as vectors containing temporary matrices, for instance, those used in the computation of the stabilization term or in the inversions of local linear systems.

### 3.2.6 Parallelization

Let us recall the definition of the global HHO problem (2.27): Find $u_{h,0}$ such that, for all $v_h \in U_{k,0}$,

\[
\sum_{T \in T_h} a_T(u_T, v_T) = \sum_{T \in T_h} \left\{ (f_T, v_T)_T - a_T(u_{T,\beta}, v_T) \right\} + \sum_{F \in F_h} (\psi_{F,\beta}, v_F)_F
\]

The first thing which one may notice is that it is nothing but the sum of local problems which are independent one from another. This remark, which allowed us to perform the static condensation, suggests that a parallelization could be added to the code, in order to speed up the building of the global system. In particular, we used OpenMP to manage the cell-loop in algorithm 1.

The choice of the parallel interface was basically forced by the architecture Code_Saturne, which is, in fact, structured with two parallelization layers. A high-level one, by means of MPI, controls the mesh reading and its subdivision into multiple processors, as well as the resolution of the global linear system. A low-level layer, managed with the OpenMP framework, assists the local routines and speeds up the process by distributing on multiple threads easy operations, for instance, initialization of vectors or matrix products, or just large local loops, as in our case.

Given the memory management of OpenMP and the structure of the method itself, no particular problems were encountered during the process of parallelization: this is actually a huge advantage of HHO.
Chapter 4

Tests and results

Now that both the theoretic and practical approach to HHO have been given, this chapter will deal with the results of our implementation. We will take care to show that the previously discussed convergences are attended, both for the intermediate operators and the final solution. An analysis of the performances and of the sensibility of the method will be provided, too.

4.1 General Information

4.1.1 Meshes

For most of our tests, the domain is the unit cube
\[ \Omega = [0,1]^3. \]

Six families of meshes, presented in Figure 4.1, have been considered, basically those of the first FVCA6 benchmark [37]:

**Hexa** or H, for short, uniform regular Hexahedra (cubes); for the finest mesh:
\[ \text{card}(\mathcal{F}_h) \approx 800k. \]

**Ker** or simply K, Kershaw meshes, heterogeneous hexahedral elements. Most of them are quite deformed, very stretched, but, the regularity increases when refining. For the finest mesh: \( \text{card}(\mathcal{F}_h) \approx 800k; \)

**CB** CheckerBoard meshes, non-conforming regular hexahedra; for the finest mesh: \( \text{card}(\mathcal{F}_h) \approx 600k. \)

**PrT** meshes of Prisms with Triangular bases; for the finest mesh: \( \text{card}(\mathcal{F}_h) \approx 300k. \)

**PrG** meshes of Prisms with General bases; for the finest mesh: \( \text{card}(\mathcal{F}_h) \approx 300k. \)
Figure 4.1: Examples of the mesh families used for the tests
TH meshes of TetraHedra; for the finest mesh: \( \text{card}(\mathcal{F}_h) \approx 2.5\text{M} \).

**Note 4.1.1** Notice that the TH meshes are not the tetrahedral ones of the benchmark: they have been created at EDF and we decided to use them because they show a little more regularity in refining than the FVCA ones. In fact, when passing from a coarse to a finer mesh, every tetrahedron is subdivided into 8 sub-tetrahedra.

Usually (all but for the TH family), we denote a mesh by their type and a number which tells in how many cells an edge of the domain is subdivided into. For instance, H8 denotes a mesh of regular cubes with eight cells per edge.

As pointed out here above, the finest meshes of each family have around five-hundred thousands faces, making the dimension of the global system of about 4.8M with \( k = 2 \), which can still be handled on standard laptops. The TH series, however, is larger. Its finest mesh has 2.5M faces: in this case, for \( k = 2 \), the dimension of the matrix of the global system is about 15M \( \times \) 15M, which could demand even up to 15GB of memory. Hence, more powerful machines may be needed to perform the computations.

### 4.1.2 Error Notation

From here on out, when talking about the order of convergence, usually denoted by \( R \), it will be compute as follows:

\[
R := -3 \frac{\log_{10} \frac{\text{err}_{\text{fine}}}{\text{err}_{\text{coarse}}}}{\log_{10} \frac{\#\text{DoFs}_{\text{fine}}}{\#\text{DoFs}_{\text{coarse}}}}
\]

where the 3 is due to the space dimension which we have considered, and \( \#\text{DoFs} \) is the total size of the global system:

\[
\#\text{DoFs} := \dim(P^k_2) \text{ card}(\mathcal{F}_h).
\]

(recall that we have performed the static condensation).

Moreover, unless stated differently, the results will always be presented in the relative form with respect to the \( L^2 \)-norm of the analytic solution, \( u \), hence

\[
\|\text{err}\|_{\text{rel}} := \left( \frac{\sum_{T \in T_h} \|\text{err}\|_T^2}{\|u\|_{L^2(\Omega)}^2} \right)^{1/2}.
\]

### 4.2 Elementary Tests

A first series of test concerns the basic operators of the method: the reconstructor \( p_T^{k+1} \) and the stability bilinear form \( j_T(\cdot, \cdot) \). The aim is to prove the consistency, equations (2.15) and (2.23), and approximation properties, equations (2.16) and
(2.24), derived in chapter 2. For these tests, we used the diffusion tensor of the FVCA mild-anisotropy test case, which is uniform on the whole domain

\[
M = \begin{pmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix}
\] (4.1)

which gives \( \rho_\flat = 2 - \sqrt{2}/2 \), \( \rho_\# = 2 + \sqrt{2}/2 \).

Concerning the consistency, once the order \( k \) is fixed, a test function \( v \in P_{d+1}^k(\Omega) \) is chosen and its reduction \( I^j_T v \) is computed: its DoFs are gathered in \( v_h \) or in \( v_{loc,T} \) when restraint to a cell \( T \). Then, one evaluates:

- the error

\[
\text{err}_{g,\text{max}} := \max_{x \in T, T \in T_h} \left| \nabla v(x) - \sum_{\phi^T_i \in B^k_T} [G_T v_{loc,T}]_i \nabla \phi^T_i(x) \right|
\]

in order to prove (2.15), \( G_T \) being the set of the Gauss point of the selected quadrature rule on a cell \( T \).

- the absolute value of each element of \( J_T v_{loc,T} \)

\[
\text{err}_{s,\text{max}} := \max_{i=1,\ldots,k; T \in T_h} |[J_T v_{loc,T}]_i|
\]

for the consistency error of the stabilization.

Accordingly to the chosen order of the method \( k \), the test functions, \( v_k \) were

\[
\begin{array}{c|ccc}
  k & 0 & 1 & 2 \\
  v_k & x + y + z & x + y z & x^3 + y^2 z \\
\end{array}
\]

The results, given in Table 4.1, are satisfying. Most of the time, on the finest mesh, the error was smaller than the machine epsilon. As expected, the worst results were measured for the most challenging mesh family, the Kershaw one. The stability operator \( j_T \) does not show results as good as those of the reconstructor \( p_{k+1}^T \), but they are not alarming, they are still below an acceptable level of precision.

Choose now a test function \( v \in H^{k+2}(\Omega) \), then, in order to prove the approximation properties, we evaluate the order of convergence of

\[
\text{err}_{g,cv} := \sum_{T \in T_h} \left\| \nabla v(x) - \sum_{\phi^T_i \in B^k_T} [G_T v_{loc,T}]_i \nabla \phi^T_i(x) \right\|_{L^2(T)}
\]

and of

\[
\text{err}_{s,cv} := \sum_{T \in T_h} v_{loc,T}^T J_T v_{loc,T}.
\]
Table 4.1: Orders of magnitude of the maxima (finest meshes) of the consistency error for the gradient reconstructor and stabilization operators for different mesh families.

<table>
<thead>
<tr>
<th></th>
<th>( \text{err}_{g,\text{max}} )</th>
<th>( \text{err}_{s,\text{max}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>1E-26</td>
<td>1E-24</td>
</tr>
<tr>
<td>Ker</td>
<td>1E-22</td>
<td>1E-15</td>
</tr>
<tr>
<td>CB</td>
<td>1E-24</td>
<td>1E-19</td>
</tr>
<tr>
<td>PrT</td>
<td>1E-24</td>
<td>1E-17</td>
</tr>
<tr>
<td>PrG</td>
<td>1E-24</td>
<td>1E-18</td>
</tr>
<tr>
<td>TH</td>
<td>1E-22</td>
<td>1E-20</td>
</tr>
</tbody>
</table>

Table 4.2: Orders of convergence (finest over second-finest meshes) for the gradient reconstructor and stabilization operators

<table>
<thead>
<tr>
<th>R</th>
<th>( \text{err}_{g,\text{cv}} )</th>
<th>( \text{err}_{s,\text{cv}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>1.00</td>
<td>2.01</td>
</tr>
<tr>
<td>K</td>
<td>0.99</td>
<td>1.99</td>
</tr>
<tr>
<td>CB</td>
<td>1.00</td>
<td>2.00</td>
</tr>
<tr>
<td>PrT</td>
<td>1.00</td>
<td>2.01</td>
</tr>
<tr>
<td>PrG</td>
<td>1.01</td>
<td>2.02</td>
</tr>
<tr>
<td>TH</td>
<td>1.00</td>
<td>2.01</td>
</tr>
</tbody>
</table>

In both cases, the convergence rate should be \( R = k + 1 \). In fact, for the reconstructor \( p^{k+1}_T \), one can extract from (2.16)

\[
\| \nabla v - \nabla p^{k+1}_T \|_{L^2(T)} \lesssim \rho_T^{1/2} h_T^{k+1} \| v \|_{H^{k+2}(T)},
\]

instead, for the stability, the order is easily inferred from (2.24).

The test function for this last test was

\[
v(x) = \sin(\pi x) \sin(2\pi y) \sin(\pi z).
\]

The orders of convergence between the finest and second-finest mesh of each family are given in Table 4.2: they match perfectly those predicted by the theory, even for the deformed meshes. We remark some slight under-convergences only for the Kershaw and the CheckerBoard families.

### 4.3 Error Norms

The method was tested on the mild-anisotropy FVCA6 benchmark [37]. Let us recall the problem setting. The domain is the unit cube, \( \Omega = [0,1]^3 \) and the diffusion tensor is the one shown in (4.1). The analytical solution is known

\[
u(x) = 1 + \sin(\pi x) (\pi (y + 1/2)) (\pi (z + 1/3))
\]
which has the nice property of being bounded: $u(\mathbf{x}) \in [0, 2] \ \forall \mathbf{x} \in \mathbb{R}^3$. The right hand side of the problem, $f$, and the Dirichlet boundary conditions were set accordingly to (4.2). As for the HHO-related parameters, we recall that we have decided to use

$$k = 0, 1, 2 \quad l = k.$$  

Concerning the solver used for the resolution of the global system, we mostly used the sparse, conjugate gradient with algebraic-multigrid preconditioner provided by PETSc [5], an open-source suite of data structures and parallel routines for scientific computing. The conjugate gradient with Jacobi preconditioner implemented in Code_Saturne was often used, too. For both solvers, the residual threshold for the blocking criterion was usually set to $\text{eps} = 1E - 12$ and the maximum number of iterations to $\text{MAX} = 10000$.

### 4.3.1 Norms Definition

Let $u$ denote the analytical solution, and $u_h$ the HHO-issued discrete one. First, we have considered the errors from the theoretic results from subsection 2.2.4, which are recalled here:

- The energy norm:

$$\text{ene}^2 := \sum_{T \in T_h} a_T (L_T^k u - u_T, L_T^k u - u_T) \lesssim h^{2(k+1)} \|u\|_{H^{k+2}(\Omega)}^2$$

- The $L^2$ norm of the error with respect to the $P^k_0(\Omega)$-projection of the solution:

$$\text{proj}^2 := \sum_{T \in T_h} \|\pi^k_T u - u_T\|_{L^2(T)}^2 \lesssim h^{2(k+2)} \|u\|_{H^{k+2}(\Omega)}^2$$

We investigated with special interest also two other norms:

- The $L^2$ norm of the error of $p^{k+1}_T u_T$:

$$\text{reco}^2 := \sum_{T \in T_h} \|u - p^{k+1}_T u_T\|_{L^2(T)}^2$$

with an expected convergence rate of $R = k + 2$. This is the standard $L^2$ norm of the error given by the best possible continuous approximation obtainable with the method;

- The discrete-like $L^2$ norm of the error of the averages:

$$\text{avg}^2 := \sum_{T \in T_h} \|\overline{u}_T - \overline{u}_T\|_{L^2(T)}^2 \sum_{T \in T_h} (u_T - \overline{u}_T)^2 |T|$$

The expected order of convergence is $R = k + 2$. This norm may allow one to make some comparisons between HHO and low-order methods.
Table 4.3: Relative errors for the finest meshes and ratio with respect to the second-finest one of every family for different orders $k$.

<table>
<thead>
<tr>
<th></th>
<th>ene</th>
<th>proj</th>
<th>reco</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>Finest</td>
<td>R</td>
<td>Finest</td>
<td>R</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>7.89E-02 1.05</td>
<td>5.65E-04 2.01</td>
<td>5.79E-04 2.01</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.71E-03 2.00</td>
<td>5.20E-06 3.06</td>
<td>1.50E-06 3.09</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.30E-05 3.01</td>
<td>7.27E-08 4.04</td>
<td>1.28E-08 4.07</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
<td>2.50E-01 0.87</td>
<td>5.76E-03 1.67</td>
<td>5.84E-03 1.70</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.58E-02 1.91</td>
<td>1.00E-04 3.24</td>
<td>7.50E-05 3.54</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.33E-04 2.94</td>
<td>2.51E-06 4.19</td>
<td>1.69E-06 4.51</td>
</tr>
<tr>
<td>CB</td>
<td>0</td>
<td>9.43E-02 1.08</td>
<td>8.40E-04 2.04</td>
<td>9.24E-04 2.04</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.21E-03 2.00</td>
<td>2.17E-05 3.04</td>
<td>1.04E-05 2.99</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7.83E-05 3.01</td>
<td>5.24E-07 4.03</td>
<td>1.77E-07 3.98</td>
</tr>
<tr>
<td>PrT</td>
<td>0</td>
<td>1.23E-01 1.00</td>
<td>1.48E-03 1.98</td>
<td>1.76E-03 1.98</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.93E-03 1.99</td>
<td>2.24E-05 3.04</td>
<td>9.29E-06 3.02</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7.50E-05 3.01</td>
<td>4.50E-07 4.03</td>
<td>9.03E-08 4.06</td>
</tr>
<tr>
<td>PrG</td>
<td>0</td>
<td>1.41E-01 1.00</td>
<td>1.72E-03 2.01</td>
<td>1.77E-03 2.02</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.94E-03 2.02</td>
<td>3.12E-05 3.08</td>
<td>8.63E-06 3.15</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.08E-04 3.04</td>
<td>6.95E-07 4.08</td>
<td>1.36E-07 4.09</td>
</tr>
<tr>
<td>TH</td>
<td>0</td>
<td>6.03E-02 1.05</td>
<td>3.27E-04 2.01</td>
<td>3.66E-04 2.01</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>9.14E-04 1.99</td>
<td>3.70E-06 3.01</td>
<td>1.26E-06 3.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>8.62E-06 3.00</td>
<td>3.12E-08 4.01</td>
<td>7.14E-09 4.03</td>
</tr>
</tbody>
</table>

### 4.3.2 Results

We provide in Table 4.3 a summary of the results: for every mesh family and every order $k$, all the errors on the finest meshes are shown, as well as the ratio between the errors of the finest and the second-finest mesh.

**Note 4.3.1** The data about the averages errors is missing for $k = 0$ because, at that order, this error is exactly the same as the projection related one.

In order to give an idea of the convergence rates, we plotted in Figure 4.2 and Figure 4.3, respectively, the energy error and the projection-related one against the system size. Up to the due modifications, the behaviors are very similar for all the four errors considered here above. On the other hand, Figure 4.4 shows two examples of the recovered solutions on two peculiar meshes.

**Remark 4.3.1** Generally speaking, the results match the predictions. Orders of convergence higher than expected are observed for the averages errors for most of the mesh families. If it could have been somehow expected for the hexagonal case, because of its regularity, we are a little surprised to notice that this happens also for the prism families and, especially, the Kershaw one (although, this is partially due to the increase in the regularity at the finest Kershaw meshes). However, this behavior is not observed for the lowest order, $k = 0$, where the Kershaw under-performs for all the type of errors. That could be interpreted as
Figure 4.2: Relative energy errors plotted against the system size. Green: $k = 0$. Red: $k = 1$. Blue: $k = 2$. The rates of convergence R are colored accordingly to their related orders $k$. 
Figure 4.3: Relative project-related errors plotted against the system size. Green: $k = 0$. Red: $k = 1$. Blue: $k = 2$. The rates of convergence R are colored accordingly to their related orders $k$. 

(a) Hexaedral mesh types

(b) Prisms and tetrahedral mesh types
4.3. Error Norms

(a) Mesh: PrG20

(b) Mesh: Ker16

Figure 4.4: Costant piece–wise plots of the cell–DoFs of the FVCA solution for $k = 0$

an indication that HHO, given its construction, gives the best of its potential only at higher orders.

4.3.3 Discrete Max Principle

No theoretic result is available for HHO about the max principle, but we would like to know if it is satisfied in practice.

We have decided to test the solution given by the linear solver, so that, except for the inverse of the static condensation, no reconstruction could influence the results. Then, for every face, we have recovered its own DoFs and evaluated the solution at its center (given that the monomials are centered, this means that we have taken into account only the DoF related to the constants) and we have checked if the values lied between the bounds (notice that the solution of the FVCA6.1 test case satisfies $0 \leq u(x) \leq 2$). The cell DoFs have been tested analogously.

The principle was satisfied by the face DoFs except for some outliers in the meshes TH00, CB16, K8, K16 and K32 with $k = 0$. As for the cell ones, outliers have been registered for all the meshes, but only for $k = 0$: the data of every mesh can be found in Table 4.4.

As one can see, the number of cell which does not satisfies the principle is generally very small with respect to the total number. Also, the value of the error is small too, if compared to the range of the function. Moreover, the results usually improve when refining the mesh: both the percentage of the cells where the principle is not satisfied and the delta with respect to the real bound grow smaller. Only in the PrT and PrG mesh families, we observe a growth from the second-finest meshes (where no outliers have been found) to the finest one.

The results seem, however, quite satisfactory, and, once again, it seems that higher orders behave better than the zero-th one.
Table 4.4: Data about the cell-related discrete max principle for \( k = 0 \). The columns show: the number of outliers, their percentage with respect to the total number of cells, the max delta-error registered and its percentage with respect to the global range.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Over shoots</th>
<th>Under shoots</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>%</td>
</tr>
<tr>
<td>H</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>K</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>2</td>
</tr>
<tr>
<td>K</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>3</td>
</tr>
<tr>
<td>CB</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>PrT</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>PrG</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>TH</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
Table 4.5: PETSc MultiGrid-CG solver iterations for inverting the global system build with standard monomial basis functions. $k = 2$, MAX=10000, eps = $1E-12$.

<table>
<thead>
<tr>
<th>Mesh ID</th>
<th>#Iterations</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2533</td>
<td>$1E-13$</td>
</tr>
<tr>
<td>20</td>
<td>7065</td>
<td>$1E-13$</td>
</tr>
<tr>
<td>30</td>
<td>MAX</td>
<td>$1E-10$</td>
</tr>
<tr>
<td>40</td>
<td>MAX</td>
<td>$1E-09$</td>
</tr>
<tr>
<td>8</td>
<td>6185</td>
<td>$1E-13$</td>
</tr>
<tr>
<td>16</td>
<td>MAX</td>
<td>$1E-04$</td>
</tr>
<tr>
<td>32</td>
<td>MAX</td>
<td>$1E-04$</td>
</tr>
</tbody>
</table>

4.4 Sensitivity

The problem at hand turned out to be quite sensitive to the basis type chosen for the polynomial spaces. Preliminary tests would fail when the constant basis function was not set to $\phi_0^T \equiv 1$ or when, instead of the diameter, other choices of the scaling factor were considered, among those discussed in subsection 3.1.1.2: even the constant solution was hard to get for non-regular meshes.

Often, for non-regular meshes, standard monomial basis, and $k = 2$, the linear solver would not converge. As it could be inferred from Table 4.5: the linear system was obviously hard to invert, even for coarse meshes, and it looks like it was not invertible at all in the Kershaw case. Notice also that, even when the system could be solved, the number of iterations needed is ridiculously high if compared with the one needed with the inertial bases, which can be found in Table 4.7. Moreover, as one would expect, the condition number of the method seems to grow with order: in fact, with same basis setting (standard monomials) but lower orders ($k = 0, 1$), the only cases in which the solver did not reach convergence were the most irregular ones of Kershaw, and, again, in the other cases, the numbers of iterations were a lot smaller than those in Table 4.5, more precisely, less than a thousand.

Concerning the choice of the basis axes, the results for the mesh K16 and $k = 2$ solved with the Code_Saturne standard conjugate gradient solver are provided in Table 4.6 for the sake of giving an example. Referencing 3.1.1.2, one can find the number of iterations needed to invert the system for different configurations of the basis functions obtained by combining the procedures involving the inertia and/or the multiplication by the diffusion tensor $M$. For each operation the possible cases are: not performed, performed on the faces only, on cells only, or on faces and cells. First of all, we have chosen those mesh, order and solver because they showed a greater sensitivity to the basis type than the PETSc one, so that, even minimal differences between two configuration could be observed.

We notice that the naive monomial (first row) performs always worse than the inertial one: as we have already said, (almost) only in this latter case the
Table 4.6: Number of iterations needed by the Code_Saturne solver to invert the $k = 2$ problem on the mesh K16 for different choices of the basis axes based on the diffusion and/or inertial properties. Columns/rows legend: F=operation performed on the faces, C=operation performed on the cells, NO=no operation performed

<table>
<thead>
<tr>
<th>Principal axes</th>
<th>Anisotropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO MAX 8402 MAX MAX</td>
<td></td>
</tr>
<tr>
<td>C MAX 8397 MAX MAX</td>
<td></td>
</tr>
<tr>
<td>F 590 590 4360 4352</td>
<td></td>
</tr>
<tr>
<td>F&amp;C 590 590 4352 4352</td>
<td></td>
</tr>
</tbody>
</table>

iterative solver converged, at least for the problem setting at hand. We observe that the best choice is to use the principal axes for the face bases, last two rows: the obtained gain is outstanding. However, multiplying them by the anisotropy tensor deteriorates the results: compare the first and the third columns. Using or not the cell principal axes has no real effect (look at the first and second rows), but applying the anisotropy tensor to the cell axes seems to improve the results: even in the case where no principal axes were computed (first row), it allowed to solve the system (second column).

We also tried to use the eigenvectors of the anisotropy matrix as cell axes but it did not lead to any significant improvement: it behaved similarly to the cell-inertial case.

**Note 4.4.1** We would advise to always use a basis with principal axes computed only on the faces and without performing any other operations. The additional cost of this configuration is almost zero but the decrease in the number of iterations with respect to the naive type is really significant. It seems that there is no need to compute the cell principal axes; on the other hand, applying the anisotropy tensor to the cell bases could be useful, however, only in the case that no inertial operation was performed on the face bases, otherwise no significant improvements has been measured.

Some remarks may explain the behaviors described here above. The main reason is probably the static condensation. Recall that the DoFs of the global system are all face-defined, thus, it seems logic that the best results were obtained when considering the inertia of the faces. The same was not recorded with the cell inertial probably because the cell-DoFs have less influence on the global system. The reason that could explain why multiplying the face axes by the anisotropy tensor made the convergence harder is that, in the settings that we have considered, the diffusion is a 3D property, which has a larger scope than just a face. Consider also how, in the reconstructor (2.10) or in the stabilization term definition (2.22), the anisotropy is used with respect to the faces. The tensor is always multiplied by the normal of the face: in fact, what actually matters is the flux at the face, thus, since the axes lie on the face, they have no
influence on the flux itself. The same reasoning could explain the improvements obtained when the anisotropy tensor was applied to the cell axes.

4.5 Performances

One of the aim of this project was to implemented a high-order method which was also competitive with the low-order ones performance-wise, thus it is crucial to see how our code behaves. The results which will be shown in this section, were obtained on a HP Z820 Workstation, equipped with 2 Intel Xeon E5-2697 v2 processors, with 12 cores/24 threads each.

4.5.1 Parallelization and Profiling

We have already said that the HHO method has as structure naturally open to the parallelization and that we accomplished it with the \texttt{OMP} interface. Figure 4.5 shows the speed-up rates of the building times for the finest meshes of each family (except for Kershaw). It is important to say that they are not averages.

Let us address first the evident negative behavior of the tetrahedral mesh TH03. It may be useful to say that that mesh undergoes a building loop slightly different than the one used for the other meshes. In fact, in this case, there is no need for a division in sub-tetrahedra/sub-triangles when computing the integrals, hence the operation performed on a cell are less costly than the other cases: this might explain why the growth starts at a smaller number of threads than the other meshes. However, the behavior improves if the order grows: at $k = 2$ there no remarkable difference with respect to the other meshes. Naturally, this order is the most favorable for the parallelization because it demands a large computational effort and the recorded times are always greater than what could be an educated guess of the overhead induced by the parallelization itself.

On the other hand, we notice that the steepness generally reflects what one would expect. However, when passing from 12 to 16 threads the time always gets greater instead of smaller. We do not know the exact reason. We would have expected it when running on more than 24 threads: in that case, the machine having 24 physical cores, at least 2 threads would be run over the same core, situation that could undoubtedly deteriorate the performances. It is true that, when using more than 12 cores, both processors are used. We do not know exactly how the memory is shared, but, possibly, one processor might need to get some data which is stored in the local memory of the other one. This kind of cache-miss could realistically be responsible for the observed altered rate.

In the end, we might say that our algorithm, in perfect conditions, would lead to very positive parallelization speed-ups.

Some profiling analyses with cache simulation were run in order to assess the performances of our algorithm and to identify which routines needed improvements. Time-wise, the most costly call is the one computing the stability term: it does not surprise us since this operation needs a lot of inversions and
Figure 4.5: Building times [s] of the global system against the numbers of threads used for different meshes and orders.
matrix products, which, we know, are quite expensive. Among the most costly routines, there is also the function computing the integrals, as one could have expected, too. Moreover, the HHO-related routines were responsible for many first-level cache read/write data misses: it is somehow inevitable considering the great number of matrix operations. Finally, very few Last-Level cache misses were imputable to the HHO functions: given that that is the most costly type of miss, this is quite satisfactory and it proved the solid construction of our algorithm.

4.5.2 Total Times

A summary concerning the total times can be found in Table 4.7. The most interesting datum is the last column, which shows the ratio between the solving and the building times. Notice that the solving time is always at least almost five times greater than the building one. Hence, even if we had a perfectly parallelized building routine but a serial global solver, that would not change the state of things since the leading time will always be the solving one. However, more optimistic results have been obtained when using the Code_Saturne-standard conjugate gradient, instead of the PETSc one: the building times do not change of course, but the solving ones decrease relevantly, making the ratio leaning less towards the solving step. Moreover, it should be said that the times shown in the table are not an average, but just a single measure, and that the performances of PETSc may vary a lot: in fact, it happened that, comparing two runs over the same mesh, PETSc would perform the same number of flops but one in the half of the time of the other.

Note 4.5.1 The building time for K64 and $k = 2$ cannot be compared to the others building times since it demanded more efforts. In fact, the mass matrices of some very irregular cells needed to be normalized otherwise the pivots would have been smaller than the machine epsilon.

As one would expect, the highest building-/solving-time ratios were obtained for the Kershaw family, which is the most deformed one. It is interesting to notice that, for almost every mesh family, the building time gets two times bigger when passing from $k = 0$ to 1, and 4 times when passing from 1 to 2. We were not able to point out the exact reason explaining those values, but it surely is a mix of (i) the growth in the dimensions of $P_3^k(T)$, $P_2^k(F)$ and $I_2^k$, thus causing heavier matrix inversions and products, and (ii) the larger stencil used in the Gaussian quadrature rules, needed in order to computing exactly the HHO operators.

Note 4.5.2 Comparisons internal to Code_Saturne were made between HHO with $k = 0$ and CDO CDO [13]: our implementation HHO resulted up to 6 times slower. Some points should be considered in order to try to explain this. First, the CDO routines are by now fully optimized; on the contrary, our implementation of HHO is the first version in Code_Saturne and it has room for improvements, starting from the usage of an optimized matricial library. Secondly, as we have already said, our implementation is valid for any order $k$ but it
Table 4.7: Iterations, flops and times for inverting the problems on the finest meshes for $k = 0, 1, 2$ and inertial bases with the PETSc CG solver with algebraic multigrid preconditioner run serially.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Solver</th>
<th>#iter</th>
<th>#flops</th>
<th>Build [s]</th>
<th>Solve [s]</th>
<th>Solve Build</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Build</td>
<td>Solve</td>
<td>Build</td>
</tr>
<tr>
<td>0</td>
<td>H</td>
<td>11</td>
<td>2.87E+08</td>
<td>2.12E+01</td>
<td>1.12E+02</td>
<td>5.3</td>
</tr>
<tr>
<td>1</td>
<td>H</td>
<td>45</td>
<td>8.20E+09</td>
<td>4.00E+01</td>
<td>5.38E+02</td>
<td>13.4</td>
</tr>
<tr>
<td>2</td>
<td>H</td>
<td>66</td>
<td>4.46E+10</td>
<td>1.68E+02</td>
<td>7.65E+02</td>
<td>4.5</td>
</tr>
<tr>
<td>0</td>
<td>K</td>
<td>72</td>
<td>1.88E+09</td>
<td>2.23E+01</td>
<td>3.01E+02</td>
<td>13.5</td>
</tr>
<tr>
<td>1</td>
<td>K</td>
<td>171</td>
<td>3.11E+10</td>
<td>4.15E+01</td>
<td>8.23E+02</td>
<td>19.9</td>
</tr>
<tr>
<td>2</td>
<td>K</td>
<td>510</td>
<td>3.45E+11</td>
<td>1.70E+02</td>
<td>3.70E+03</td>
<td>21.7</td>
</tr>
<tr>
<td>0</td>
<td>CB</td>
<td>15</td>
<td>4.92E+08</td>
<td>1.77E+01</td>
<td>2.41E+02</td>
<td>13.6</td>
</tr>
<tr>
<td>1</td>
<td>CB</td>
<td>35</td>
<td>8.98E+09</td>
<td>4.02E+01</td>
<td>3.53E+02</td>
<td>8.8</td>
</tr>
<tr>
<td>2</td>
<td>CB</td>
<td>72</td>
<td>7.10E+10</td>
<td>1.78E+02</td>
<td>1.00E+03</td>
<td>5.6</td>
</tr>
<tr>
<td>0</td>
<td>PrT</td>
<td>35</td>
<td>3.27E+08</td>
<td>6.89E+00</td>
<td>3.77E+01</td>
<td>5.5</td>
</tr>
<tr>
<td>1</td>
<td>PrT</td>
<td>67</td>
<td>4.20E+09</td>
<td>1.13E+01</td>
<td>9.24E+01</td>
<td>7.0</td>
</tr>
<tr>
<td>2</td>
<td>PrT</td>
<td>123</td>
<td>2.82E+10</td>
<td>5.26E+01</td>
<td>3.45E+02</td>
<td>6.5</td>
</tr>
<tr>
<td>0</td>
<td>PrG</td>
<td>18</td>
<td>2.00E+08</td>
<td>8.57E+00</td>
<td>5.52E+01</td>
<td>6.4</td>
</tr>
<tr>
<td>1</td>
<td>PrG</td>
<td>30</td>
<td>2.45E+09</td>
<td>1.65E+01</td>
<td>1.05E+02</td>
<td>6.4</td>
</tr>
<tr>
<td>2</td>
<td>PrG</td>
<td>94</td>
<td>2.90E+10</td>
<td>6.83E+01</td>
<td>4.37E+02</td>
<td>6.4</td>
</tr>
<tr>
<td>0</td>
<td>TH</td>
<td>21</td>
<td>1.29E+09</td>
<td>1.14E+01</td>
<td>9.92E+01</td>
<td>8.7</td>
</tr>
<tr>
<td>1</td>
<td>TH</td>
<td>100</td>
<td>3.89E+10</td>
<td>4.18E+01</td>
<td>5.41E+02</td>
<td>13.0</td>
</tr>
<tr>
<td>2</td>
<td>TH</td>
<td>154</td>
<td>2.14E+11</td>
<td>1.73E+02</td>
<td>2.80E+03</td>
<td>16.2</td>
</tr>
</tbody>
</table>
could be easily specified to the case \( k = 0 \) (recall that, in that case, \( \dim P^0_d = 1 \) which, among others possible optimizations, makes the matrix inversions trivial) which would undoubtedly improve the performances. Moreover, HHO was explicitly designed to be valid for any order, thus it seems perfectly logic that, at the lowest order, it could underperform with respect to other lowest-order-specified methods.

### 4.5.3 Choice of the Order

An interesting analysis compares the effort needed for solving the problem and the error obtained with different values of the order \( k \). In Figure 4.6 we plotted the "cost" of the inversion against the discrete-like \( L^2 \) averages error. We have chosen this error since the solution does not undergo any post treatment, so that the solving cost is a good approximation of the total one (recall that the solving is always the most expensive task). Moreover, since the PETSc outcomes may vary a lot time-wise, it looked that the number of performed flops was more stable and thus we reckoned that it could be a good estimator of the effort demanded by the problem, vis-à-vis also the fact that that number should not have changed if we would have run the cases on a different machine.

Suppose that one wants to reach a given error threshold, one may wonder which is the best combination of mesh refining and order. From the plots in Figure 4.6, it could be inferred that, with an higher order \( k \), one could get the wished error more easily and quickly, in the sense that it is reached with a smaller computation cost and on coarser meshes than with a lower order. Analogously, if one is ready to pay a given cost, choosing a high order and a coarse mesh would give better errors than a low order and a fine mesh.

It is interesting to notice in Figure 4.6 that there exists a logarithmic relation between the number of flops in the solving stage and the error; this happens also for the other error types. Moreover, the steepness of the slopes depends on the mesh type, as one would expect. For instance, fix an order and consider the curves of the Kershaw and of the Hexa. The Kershaw one is steeper: being irregular and deformed, the inversion is harder and the errors are greater. Finally, it is interesting to compare the steepness of the slopes of different orders for a fixed mesh type. The slopes obtained with \( k = 2 \) are always less steep than those of the lower orders. It could be inferred that it is better to use high orders since the cost, as function of the error, grows more slowly: a little more effort would result in a greater gain in the error for high orders with respect to the low-order cases.

We were not able to explicitly write the law governing the relationship, but, if found, it could be very useful since one could recover which order, refinement level and polytopes type to choose in order to get a certain error within a given cost, and all this even before meshing the domain.
Figure 4.6: Solving effort plotted against $L^2$ averages errors for different meshes. Green: 0th order. Red: 1st order. Blue: 2nd order.
4.6 Heterogeneous Anisotropy Tests

A second test with a more challenging setup was performed. The domain is $\Omega = [0, 1] \times [0, 0.1] \times [0, 0.1]$. The anisotropy tensor is piece-wise constant

$$M = \begin{pmatrix} \kappa(x) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

with

$$\kappa(x) = \begin{cases} \kappa_1 = 10^5 & x \leq 0.5 \\ \kappa_2 = 1 & \text{elsewhere} \end{cases}$$

The source term is $f \equiv 0$, and Dirichlet boundary conditions are imposed at the in/outlet

$$u_\partial(x) = \begin{cases} 1 & x = 0 \\ 0 & x = 1 \end{cases}$$

and homogeneous Neumann conditions are set elsewhere. The solution is known:

$$u(x) = \begin{cases} u_1(x) = 1 - \frac{2}{\kappa_1 + \kappa_2} x & x \leq 0.5 \\ u_2(x) = \frac{2\kappa_1}{\kappa_1 + \kappa_2} (1 - x) & \text{elsewhere} \end{cases}$$

The mesh used for the test is built by joining together two submeshes of cubes, one for each subdomain with the different definitions of the anisotropy tensor, and with different mesh-sizes ($h_{T,1} = 4 h_{T,2}$): the result is non-conforming. One can have a look at Figure 4.7.

The solution being linear, we expect null errors even even for $k = 0$, and, in fact, one can see from Table 4.8 that the errors (they are not relative) are very close to zero. It is legit that they are not exactly zero nor smaller than the machine epsilon because the threshold of the residual for the blocking criterion of the iterative solver was $1E-12$. It does not surprise us, then, that the projection-related error varies a little with the order and neither that the energy error, which, we recall, takes into account the error of the gradient and the stabilization, is $O(10^{-11})$.
Table 4.8: Energy and project-related $L^2$ error norms and discrete max principle of the heterogeneous anisotropy test for different orders.

<table>
<thead>
<tr>
<th>$k$</th>
<th>ene</th>
<th>proj</th>
<th>Max principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.34E-11</td>
<td>3.74E-14</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>1.57E-11</td>
<td>7.13E-15</td>
<td>OK</td>
</tr>
<tr>
<td>2</td>
<td>3.22E-11</td>
<td>2.01E-14</td>
<td>OK</td>
</tr>
</tbody>
</table>
Chapter 5

Conclusions

In this last chapter, we will sum up what has been presented in this report and, finally, we will draw some conclusions and provide axes of analysis which could lead future developments and investigations around the considered HHO problem.

5.1 What has been achieved

In chapter 2, after setting the theoretic background needed for the development of the Hybrid High-Order schemes, we described the main features and properties of the key-operators on which the HHO method for the anisotropic diffusion is hinged and we recalled the principal results while re-demonstrating the most important of them.

In chapter 3, we provided an algebraic translation of the HHO theoretic framework, giving the details for every operator and step of the method. Special attention was given to the implementation, including the parallelization of the code, and, in particular, to the choice and the construction of the basis of the polynomial spaces, which, computational-wise, turned out to be a key point of the method.

In chapter 4, we tested our HHO operators and algorithm on the FVCA6 benchmark. The error estimates predicted by the theory were recovered and very satisfying speed-up parallelization rates were measured, too. A sensitivity to the choice of the bases was recorded. Finally, we made a comparison between the results obtained with different order of the method $k$.

5.2 Final remarks and outlooks

Even if the HHO framework allows a functional setting of polynomials of arbitrary order $k$ (and consequently, arbitrary rates of convergence), we restrained
ourselves to $0 \leq k \leq 2$. The main reason is the industrial context of \textit{Code_Saturne} in which our code has been integrated. In fact, on one hand, we want a method as general as possible and with the least strict hypotheses. Hence, for instance, it is useless to choose a high order if the solution is not very smooth (which is the case in most industrial applications) because the theoretic order of convergence will not be recovered. Moreover, the computation cost and the conditioning number, as one would expect, grow with the order. Thus, in order to get computational times which are comparable with those of lowest-order methods (for instance, Finite Volumes or CDO) and reasonable for an industrial application, we decided to put an upper-bound to the order, $k \leq 2$.

On the other hand, willing to have a method applicable to most of the cases, we had to leave aside some possible optimizations. The most evident example is the quadrature procedure. So far, our algorithm uses the standard \textit{Code_Saturne} way, which is dividing into sub-simplexes then using Gauss rules. This naturally implies that computing the integrals is one of the most limiting step of the method in terms of performances which has been confirmed by a profiling analysis.

A first improvement could allow us to avoid computing more than one time the data related to the interfaces whose basis functions and mass matrices are independent of the cell. Once we have encountered an internal face, we store the information which we will be used again when dealing with another cell which shares the same face. This could considerably cut the quadrature cost but it may impact the possibility of building the local systems perfectly in parallel and cell-wise.

Another idea could be to exploit the divergence theorem to transform the costly high-order volumetric integrals into 2D ones. Since we are using simple monomial basis functions, that could be applied without to much effort. Another solution could be to use the Euler’s homogeneous function theorem combined again with the divergence and/or the Green’s theorem as shown in [41]. The gain of this strategy involving 3D integrals rewritten as 2D ones might be increased by coupling it with the Gauss-Lobatto quadrature rules. Since the stencil always includes the end points of the integration interval, many function evaluations, those on the edges in particular, could be reused. Another smart procedure that could be exploited is the reduction of volumetric moments of inertia into 1D integrals presented by [40]: in addition to the computation of the principal axes used in the basis functions, it can be applied to the building of the mass matrices, which, if monomial bases are considered, are basically moments of inertia.

Being able to integrate over geometric shapes other than simplexes (for instance, quadrangles in 2D or pyramids and hexahedra in 3D) can definitely ease the computation. However, the available strategies usually need the integration domain to verify some hypotheses (convexity above all) and some implementation precautions have to be taken (for instance, a precise ordering of the vertices in an element): hence, the range of applicability would be reduced, situation that we would like to avoid.

The impossibility of using some standard optimization techniques led us
to find other expedients in order to improve the performances. First of all, taking advantage of the construction of HHO itself which is highly local, the algorithm was parallelized, making it among the first parallel implementations of HHO. The speed-up rates measured for the system-building times were very satisfying, and, for \( k = 2 \), the theoretic rate was basically recovered for all the mesh families that have been considered. Even though, not surprisingly, the solving time appears to be the leading one when compared to the building ones, the results concerning the parallelization are promising and suggest that further works in that direction may improve again the performances.

Another optimization-related axis of work that led to satisfying results was the analysis of the basis functions used for the polynomial spaces. The method proved itself to be sensitive to the choice of these functions, at least for the problem at hand: it has been shown how minimal differences in the set-up of the basis functions can lead to large variations of the conditioning number of the global system (and, consequently, the number of iterations needed by the solver, hence the solving time, too).

In this work, we started by using the naive monomial bases and we devised some modifications in order to take into account the diffusion properties of the problem and/or the inertia of the element at hand. Using the anisotropy tensor to compute the axes generating cell basis functions generally decreased the number of iterations. However, the most positive results were obtained when considering the principal axes of inertia of the faces: for some cases, the number of iterations needed to solve the global system was more than ten times smaller than the one issued from the naive monomial basis. In fact, using the principal axes makes the first-order monomials of the basis to be orthogonal, hence the local mass matrices used to build the system are diagonal or at least quite sparse, leading to an easier handling. In theory, the same applies for the principal axes of the cells but no significant improvement has been measured: this might be explained by the fact that a static condensation procedure is performed on the local system before being assembled into the global one, which is thus written only in terms of face-defined DoFs.

We are convinced, then, that further investigations concerning the bases should lead future developments. A starting point could be using orthonormal functions, which would be different from our principal-axes type because all the functions would be orthogonal. Moreover, different types of scalar products used in the orthogonalization or for building of the inertia-like matrix could be considered. In our test cases, the diffusion tensor was piece-wise constant and that could have been one of the reasons of the success of the principal axes, but, in the case of generic anisotropy, one could possibly use the tensor as a weight for the scalar product. Finally, other families of polynomials (Chebyshev, Legendre, Jacobi,\ldots) could be considered.

Most likely, performance-wise, other bases would not be better than the monomial one, at least for the construction and evaluation steps, but they could significantly improve the condition number of the matrix of the global system and consequently the solving time.
The results concerning the error estimates were satisfying, at least for the orders that we have considered: the expected rates of convergence were generally recovered, even on deformed meshes, and some super-convergences were recorded, too. If we made some comparisons among the tested orders, which, we recall, are \( k = 0, 1, 2 \), we would be led to think that it is better to choose a high-order scheme, at least for the problem at hand. In fact, on deformed meshes, rates of converge closer to the theoretic ones were measured with \( k \geq 1 \) than with \( k = 0 \), and, differently from the zero-th order, the discrete max principle was always satisfied for \( k \geq 1 \). Moreover, the building-over-solving-time ratio does not seem to grow significantly with the order. Also, the speed-up rates of the parallelization seem to improve when moving towards higher orders and, for \( k = 2 \), were very close to the theoretic ones. More importantly, the relations discovered between the solving effort and the error suggest that, for an equivalent increase in the effort, the obtained gain on the error is greater at higher orders. Analogously, a given error is achieved on coarser meshes and with less effort for high orders than for low ones.

For the time being, we think that \( k = 1 \) could be a good compromise. The computational times are still within a reasonable range and orders of convergence up to the third are ensured. Moreover, these rates could be easily matched also by the time scheme in a non-stationary problem, which, most likely, would be the next development of our work.
Bibliography


