UNIVERSITAT POLITÈCNICA DE VALÈNCIA

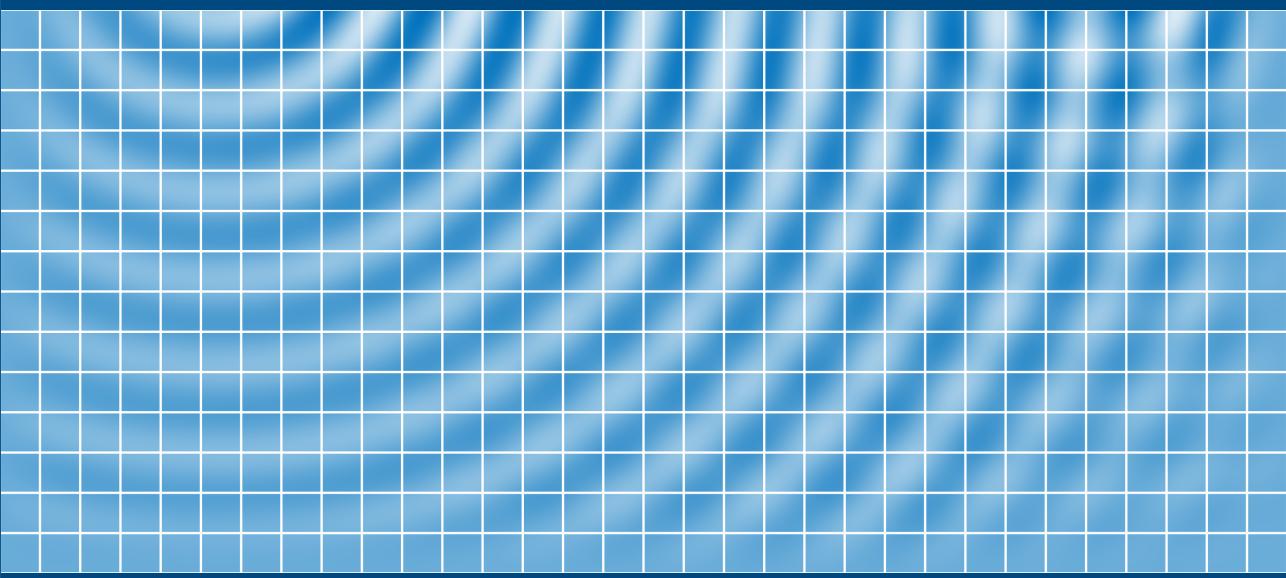
I2MB
INSTITUTO DE INGENIERÍA MECÁNICA Y BIOMECÁNICA

PhD Thesis

High-order techniques in the Cartesian grid Finite Element Method based on the discontinuous Galerkin formulation

Héctor Navarro García

Supervisors: J. J. Ródenas García J. M. Navarro Jiménez



Valencia, September 2025

PhD Thesis

High-order techniques in the Cartesian grid Finite Element Method based on the discontinuous Galerkin formulation

for the degree of

Doctor in Industrial Engineering and Production

presented by

Héctor Navarro García

at the

Department of Mechanical and Materials Engineering

of Universitat Politècnica de València

Supervised by

Dr. Juan José Ródenas García
Dr. José Manuel Navarro Jiménez

València, September 2025

PhD Thesis

High-order techniques in the Cartesian grid Finite Element Method based on the discontinuous Galerkin formulation

Presented by: Héctor Navarro García

Supervised by: Dr. Juan José Ródenas García

Dr. José Manuel Navarro Jiménez

QUALIFYING TRIBUNAL

President: Dr. Antonio Falcó Montesinos

Vocal: Dr. Sonia Fernández Méndez

Secretary: Dr. Francisco David Denia Guzmán

València, September 2025

Abstract

Every day, engineers in all industrial branches face increasingly complex problems and need a deeper understanding of the involved physical phenomena and more sophisticated tools to develop their designs. In most cases, the mathemathical models that explain the physics are known, and the main limitation lies in the validation and optimization of the designs. With prototyping relegated to the final stages due to its high cost, computer-based numerical analysis methods are the natural choice for these evaluations. The range of methods is considerable, with formulations specially adapted for specific cases. However, solving large and complex problems, both from the physics and domain perspectives, remains a challenge due to the significant resources these simulations require.

Transient electromagnetism problems addressed within industries such as aerospace, telecommunications, or defense fall into this category. Although they have historically relied on the finite-difference time-domain method due to its simplicity, the increasing complexity of simulations has fostered the use of the finite element method (FEM) thanks to its greater versatility. However, the standard formulation of the FEM does not meet the current needs of the industry in terms of either accuracy or efficiency.

In this thesis, different technologies in the FEM field are explored and combined into a new methodology, the Cartesian Grid Discontinuous Galerkin method (cgDG), which promotes the synergies that appear between them. The fundamentals of the method are high-order elements, the discontinuous Galerkin formulation, and discretization with Cartesian meshes. First, high-order elements have proven to be more efficient in capturing smooth solutions, requiring fewer degrees of freedom than their low-order equivalents to achieve the same level of accuracy. Second, the discontinuous Galerkin formulation exhibits interesting properties that allow it to faithfully capture

the physics while dividing the original potentially large problem into a set of smaller problems that can be solved more efficiently. Lastly, discretization with Cartesian meshes makes the mesh generation trivial, simplifies evaluation processes, provides great flexibility in capturing the problem domain, and allows for a reduction in the resources needed to solve a problem. The merge of these technologies into a single method holds great potential, but its development presents significant challenges: the computation of integration quadratures that achieve the expected convergence orders for high-order approximations, the design of strategies that locally improve the conditioning of the solution or the development of techniques that ensure the method's efficiency regardless of the domain's position within the Cartesian mesh.

The cgDG methodology proposed in this thesis brings together a range of innovative solutions to address each of these challenges. Implementations of cgDG have been developed to solve advection and electromagnetism problems in both two-dimensional and three-dimensional cases. The numerical results obtained demonstrate the effectiveness of each of the proposed solutions, as well as the good performance of the methodology even when applied in complex scenarios. With a view to increasing the efficiency of simulations and reducing user expertise requirements, the final stage of the thesis focused on investigating error and solution regularity indicators to guide an automatic mesh adaptation strategy. The preliminary results obtained for two-dimensional problems are promising and encourage continued work in this direction.

Resumen

Cada día, los ingenieros en las distintas ramas industriales afrontan problemas más complejos y necesitan tener un conocimiento más profundo de los fenómenos físicos involucrados y herramientas más sofisticadas para elaborar sus diseños. En la mayoría de casos, los modelos matemáticos que explican la física son conocidos y la principal limitación reside en la validación y optimización de los diseños. Con el prototipado relegado a las etapas finales dado su alto coste, los métodos de análisis numérico por ordenador son la elección natural para estas evaluaciones. La paleta de métodos es considerable con formulaciones especialmente adaptadas para casos concretos. Sin embargo, la resolución de problemas de gran tamaño y complejos tanto desde el punto de vista de la física como del dominio sigue siendo un reto debido a la gran cantidad recursos que requieren estas simulaciones.

Los problemas transitorios de electromagnetismo abordados dentro de industrias como la aeronáutica, telecomunicaciones o defensa entran dentro de esta categoría. Aunque históricamente han confiado en el método de las diferencias finitas en el dominio temporal por su simplicidad, la creciente complejidad de las simulaciones ha favorecido la adopción del método de elementos finitos (MEF) gracias a su mayor versatilidad. Sin embargo, la formulación tradicional del MEF no satisface las necesidades actuales de la industria ni en términos de precisión ni de eficiencia.

En esta tesis se exploran distintas tecnologías del ámbito del MEF y se combinan en una nueva metodología, el *Cartesian grid discontinuous Galerkin method* (cgDG), que potencia las sinergias a parecen entre ellas. Los pilares del método son: los elementos de alto grado, la formulación Galerkin discontinua y la discretización con mallas Cartesianas. Primero, los elementos de alto grado han demostrado ser más eficientes a la hora de capturar soluciones suaves requiriendo menos grados de libertad que

sus equivalentes de bajo orden para alcanzar un mismo nivel de precisión. Segundo, la formulación Galerkin discontinua tiene propiedades interesantes que le permiten capturar fielmente la física a la vez que divide el problema original de gran tamaño en un conjunto de problemas pequeños que se pueden resolver de forma más eficiente. Por último, la discretización con mallas Cartesianas vuelve trivial el proceso de generación de la malla, simplifica su evaluación, da una gran libertad a la hora de capturar el dominio del problema y permite reducir los recursos que son necesarios para resolver un problema. La combinación de estas tecnologías en un único método tiene un gran potencial, pero su desarrollo plantea grandes retos: el cálculo de cuadraturas de integración que permitan alcanzar los órdenes de convergencia esperados para las aproximaciones de alto grado, el diseño de estrategias que mejoren localmente el condicionamiento de la solución, el desarrollo de técnicas que aseguren la eficiencia del método con independencia de la posición del dominio dentro de la malla Cartesiana,...

La metodología cgDG que se propone en esta tesis agrupa un abanico de soluciones innovadoras que vienen a dar respuesta a cada uno de estos retos. Se han desarrollado implementaciones del cgDG para resolver problemas de advección y de electromagnetismo tanto para el caso plano como tridimensional. Lo resultados numéricos obtenidos demuestran la efectividad de cada una las soluciones propuestas así como el buen comportamiento de la metodología aún aplicada en escenarios complejos. Con la vista puesta en aumentar la eficiencia de las simulaciones y facilitar la labor de los usuarios, en una última etapa de la tesis se han investigado indicadores de error y regularidad de la solución que permitan guiar una estrategia de adaptación automática de la malla. Los resultados preliminares obtenidos para problemas planos resultan prometedores y animan a seguir trabajando en esta línea.

Resum

Cada dia, els enginyers en les diferents branques industrials afronten problemes més complexos i necessiten tindre un coneixement més profund dels fenòmens físics involucrats i ferramentes més sofisticades per a elaborar els seus dissenys. En la majoria de casos, els models matemàtics que expliquen la física són coneguts i la principal limitació residix en la validació i optimització dels dissenys. Amb el prototipatge relegat a les etapes finals donat el seu alt cost, els mètodes d'anàlisi numèrica per ordinador són l'elecció natural per a estes avaluacions. La paleta de mètodes és considerable amb formulacions especialment adaptades per a casos concrets. No obstant això, la resolució de problemes de gran grandària i complexos tant des del punt de vista de la física com del domini continua sent un repte degut a la gran quantitat recursos que requerixen estes simulacions.

Els problemes transitoris d'electromagnetisme abordats dins d'indústries com la defensa, aeronàutica o telecomunicacions entren dins d'esta categoria. Encara que històricament han confiat en el mètode de les diferencies finites en el domini temporal per la seua simplicitat, la creixent complexitat de les simulacions ha afavorit l'adopció del mètode d'elements finits (MEF) gràcies a la seua major versatilitat. No obstant això, la formulació tradicional del MEF no satisfà les necessitats actuals de la indústria ni en termes de precisió ni d'eficiència.

En esta tesi s'exploren diferents tecnologies de l'àmbit del MEF i es combinen en una nova metodologia, el *Cartesian grid discontinuous Galerkin method* (cgDG), que potencia les sinergies a semblen entre elles. Els pilars del mètode són: els elements d'alt grau, la formulació Galerkin discontínua i l'ús de malles Cartesianes. Primer, els elements d'alt grau han demostrat ser més eficients a l'hora de capturar solucions suaus requerint menys graus de llibertat que els seus equivalents de baix orde per a

aconseguir un mateix nivell de precisió. Segon, la formulació Galerkin discontínua té propietats interessants que li permeten capturar fidelment la física alhora que dividix el problema original de gran grandària en un conjunt de problemes xicotets que es poden resoldre de forma més eficient. Finalment, l'ús de malles Cartesianes torna trivial el procés de generació de la malla, simplifica l'avaluació, dona una gran llibertat a l'hora de capturar el domini del problema i permet reduir els recursos que són necessaris per a resoldre un problema. La combinació d'estes tecnologies en un únic mètode té un gran potencial, però el seu desenrotllament planteja grans reptes: el càlcul de quadratures d'integració que permeten aconseguir els ordes de convergència esperats per a les aproximacions d'alt grau, el disseny d'estratègies que milloren localment el condicionament de la solució, el desenrotllament de tècniques que asseguren l'eficiència del mètode amb independència de la posició del domini dins de la malla Cartesiana...

La metodologia cgDG que es proposa en esta tesi agrupa un ventall de solucions innovadores que venen a donar resposta a cada un d'estos reptes. S'han desenrotllat implementacions del cgDG per a resoldre problemes de advecció i d'electromagnetisme tant per al cas pla com tridimensional. El resultats numèrics obtinguts demostren l'efectivitat de cada una les solucions proposades així com el bon comportament de la metodologia encara aplicada en escenaris complexos. Amb la vista posada a augmentar l'eficiència de les simulacions i facilitar la labor dels usuaris, en una última etapa de la tesi s'han investigat indicadors d'error i regularitat de la solució que permeten guiar una estratègia d'adaptació automàtica de la malla. Els resultats preliminars obtinguts per a problemes plans resulten prometedors i animen a continuar treballant en esta línia.

Acknowledgements

The development of this doctoral thesis has been a long journey, not without its challenges, but one in which I was never alone. The people who accompanied me throughout this adventure were essential, and without them, reaching the culmination represented by this document would have been impossible. To all of them, I extend my most sincere gratitude:

To my supervisors, Dr. Juan José Ródenas and Dr. José Manuel Navarro, and also to Dr. Enrique Nadal —whose name may not appear on the cover but who was always there— for giving me the opportunity to work with them, for sharing their deep knowledge and passion for the field of computational mechanics, for guiding me not only academically but also personally and professionally, and for their patience and understanding throughout the process.

To researchers Dr. Rubén Sevilla, Dr. Antonio Huerta, Dr. Pedro Díez, and Dr. Matteo Giacomini, for embracing the collaborations proposed during this thesis, for generously offering their time, and for welcoming me into their research groups as one of their own.

To my colleagues at the Institute of Mechanical and Biomechanical Engineering of Valencia, and more broadly at the Department of Mechanical and Materials Engineering, for providing an exceptional working environment where long office hours never felt burdensome, and for their constant willingness to help. A special mention goes to Dr. David Muñoz and Dr. Antolín Martínez, who have become more than colleagues and who have brightened difficult moments, whether with sharp ideas during technical discussion or by lifting spirits during informal conversations. I also thank the Universitat Politècnica de València, which provided an outstanding framework for developing my research and training as a scholar.

To the ERASMUS+ programme, the Ministry of Science, Innovation and Universities (projects: FPU17/03993 and EST22/00489), the Ministry of Economy and Competitiveness (project: DPI2017-89816-R), and the Generalitat Valenciana (project: Prometeo/2021/046) for providing the financial support without which this thesis would never have been possible.

To Dr. Francisco Chinesta, who placed his trust in me and gave me the opportunity to begin working as a researcher even before completing this thesis.

To my family. I can never fully express my gratitude for the unconditional support of my parents. Even though the thesis may have created some distance at times and caused them more concern than I would have ever wished, they have been an endless source of understanding and help. And to my sister, always there to listen, to distract me when I needed it, to offer advice, and to move mountains to help me solve whatever challenge I faced.

To my partner. She has been by my side through it all offering stability, listening, supporting, empathizing, and providing exactly what I needed even before I had to ask. She has been the best companion I could ever have hoped for.

To all of you, I extend my most sincere and profound gratitude.
This thesis would not have been possible without your support.

Contents

Abstract **vii**

Resumen **ix**

Resum **xi**

Acknowledgements **xiii**

1 Introduction **1**
1.1 Motivation 1
1.2 Scope 2
1.2.1 Numerical simulation of hyperbolic problems 2
1.2.2 Towards efficient solvers with automated optimization of the discretization 9
1.3 Objective 11
1.4 Outline of the thesis 13

2 Foundations of the Cartesian grid discontinuous Galerkin method **15**
2.1 Statement of the model problem 16
2.1.1 The electromagnetic problem 17
2.1.2 Towards numerical simulation 20
2.2 Discontinuous Galerkin formulation 29
2.2.1 General setting: conservative laws 29
2.2.2 Semi-discrete system 30

2.2.3 Application to the electromagnetic problem 31
2.3 Cartesian grid Finite Element Method 33
2.3.1 Comments on integration in cgFEM 36
2.3.2 Implementation of Perfectly Matched Layers in cgFEM 40

3 The cgDG in the 2D case 43
3.1 Discontinuous Galerkin formulation in the cgFEM's framework 44
3.1.1 Spatial discretization 44
3.1.2 Evaluation of the operators in the weak form 45
3.1.3 Superposition of elements with non-overlapped support 50
3.2 Non-Uniform Rational B-Splines 54
3.2.1 Definition and properties 55
3.2.2 Knot insertion 56
3.2.3 The NURBSLab toolbox 58
3.3 NURBS-Enhanced Integration 59
3.3.1 Integration in NEFEM 59
3.3.2 NURBS-Enhanced integration in cgFEM 61
3.4 Integration Decomposition 63
3.4.1 Preprocessing geometry definition 64
3.4.2 Initial Domain Decomposition 66
3.4.3 Labeling stage 67
3.4.4 Refinement stage 69
3.5 Stabilized scheme 76
3.5.1 Stability indicator 76
3.5.2 Key features of the stabilization strategy 77
3.5.3 Detailed description of the stabilization algorithm 79
3.6 Numerical results 84
3.6.1 Verification 84
3.6.2 Scattering by a PEC circular cylinder 91
3.6.3 Scattering by a PEC NACA0012 aerofoil 95
3.6.4 Scattering by complex geometries 99

4 The cgDG in the 3D case 105
4.1 Domain definition and geometry pre-processing 106
4.2 Integration 121
4.2.1 Defining the topology of the integration mesh 121
4.2.2 Approximating integration domains 125
4.2.3 Implementation issues 135
4.3 Stabilisation strategies 138
4.3.1 Multiple reassignment 138
4.3.2 Composite extrapolation 142
4.4 Numerical results 150
4.4.1 Numerical integration in cgDG 3D 150
4.4.2 Performance of stabilization strategies 158

4.4.3 Scattering of PEC sphere 160

5 *hp*-Adaptive Strategies in cgDG **165**
5.1 Towards automatic hp-adaptive cgDG 166
5.2 Explicit indicators 168
5.2.1 Error indicator 168
5.2.2 Mesh adaptation criteria 172
5.3 Multigrid indicators 175
5.3.1 Set of calculation meshes 175
5.3.2 Error indicator 176
5.3.3 Auxiliary estimates of polynomial approximations in cgDG . . 178
5.3.4 Estimates on multigrid error indicator 181
5.3.5 Mesh adaptation criteria 184
5.4 *hp*-Adaptive strategy workflow 186
5.5 Inter-discretization field projection 192
5.6 Numerical results 194
5.6.1 Studies on the performance of error measures 194
5.6.2 Study on the performance of regularity indicators 205
5.6.3 Study on the performance of *hp*-adaptation strategy 208

6 Closure **219**
6.1 Conclusions 219
6.2 Future works 221

Appendices **224**

Bibliography **255**

Chapter 1

Introduction

1.1. Motivation

Since their emergence in the mid-20th century, numerical methods have grown in importance to become a fundamental pillar in the development of our society. The continuous advancements in computational mechanics—born from the merging of this branch of mathematics and computer science—have enabled industries to develop new and improved products to meet society's needs, markets to become more accessible by democratizing the economy, regulatory bodies to create guidelines to reduce the human impact on the planet, scientists to deepen their understanding of our environment, healthcare professionals to build protocols and treatments that enhance our life expectancy and quality of life, and so on. It can be said that the vast majority of processes sustaining our current standard of living rely on tools from computational mechanics.

Given its relevance, it is not surprising that the field has reached a remarkable level of maturity. However, users of these technologies face increasingly complex problems that surpass the capabilities of current tools, demanding the development of more efficient methods, more powerful computing stations, and better-optimized implementations. In this context, the present thesis aims to make a modest contribution in the mathematical side of this domain by proposing an advanced numerical method specifically designed for solving problems governed by hyperbolic systems of

partial differential equations (PDEs). This new method, named the Cartesian grid Discontinuous Galerkin method (cgDG) after the computational strategies in its core, would have applications in highly relevant fields such as meteorology, oceanography, acoustics, telecommunications, defense, energy, and more.

Historically, simulations tackled by the cgDG have been known to demand a vast amount of resources. On one hand, the models used within the various applicable computational methods must be constructed with great care, as an incorrect definition can lead to results that poorly reflect real behavior or produce simulations that are not feasible in practice. Analysts tasked with designing these models face a complex and labor-intensive task, where they must make many key decisions guided solely by their intuition and know-how. On the other hand, numerical methods require a significant amount of resources, both in terms of computational power and time. Even for the simplest configurations, the final solution is only obtained after building and solving very large systems of equations or, alternatively, a large number of smaller systems. In any case, the demands are considerable especially compared to simulations in other physical domains.

The cgDG aims to simplify and accelerate the modeling process by automating these steps as much as possible and reducing the analyst's responsibility for the quality of the final results. It also seeks to manage the required resources by adapting the formulation so it can be solved in the most efficient way possible using current computing systems. Ultimately, this thesis aims to contribute to the advancement of computational mechanics (and, by extension, society) by proposing a new method that enables robust, user-friendly, accurate, and efficient simulation of a class of problems that exceed the capabilities of current tools.

1.2. Scope

1.2.1. Numerical simulation of hyperbolic problems

Partial Differential Equations (PDEs) are an extremely powerful mathematical tool when it comes to modeling physical phenomena. Depending on their structure, and the characteristics this structure induces in the solution, they are classified into three categories: elliptic, parabolic, and hyperbolic. Table 1.1 summarizes the key aspects of each category.

To clarify the differentiating behaviors leading to this classification, a small theoretical example is presented. Suppose we have three problems defined over the domain $X \times T \in \mathbb{R}^2$ with solutions f_e, f_p, and f_h, where the subscripts e, p, and h indicate the type of PDE involved in the formulation: elliptic, parabolic, and hyperbolic, respectively. By introducing a perturbation at the point $p = (x_p, t_p) \in X \times T$, the definition of the problems changes, resulting in the solutions g_e, g_p, and g_h. As noted in the

PDE Type	Modeled phenomena	Solution response under perturbation	Practical examples
Elliptic	Equilibrium	The entire field is affected	Steady-state fluid flow, Electrostatics, Gravitational potential
Parabolic	Dissipation; Dispersion	The field is only partially affected	Heat conduction; Chemical or biological diffusion; Black-Scholes equations
Hyperbolic	Propagation	The field is affected within a domain that depends on the characteristic speed of information propagation in the medium	Mechanical vibrations; acoustic, seismic, and electromagnetic waves

Table 1.1: *Summary of PDE categories*

previous paragraph, this variation in boundary conditions affects each case differently depending on the type of PDE governing the system's behavior. Let us denote as $\mathcal{D}_i$ the subdomain of $X \times T$ where the condition $f_i \neq g_i$ holds. In Figure 1.1, the subdomains $\mathcal{D}_e$, $\mathcal{D}_p$, and $\mathcal{D}_h$ are shown as shaded regions. Firstly, the coincidence of $\mathcal{D}_e$ with $X \times T$ highlights the global nature of elliptic operators. In the parabolic case, there is a global response of the system in the x direction but not in the t direction; the solution changes at points to the right of p but remains unaltered to the left. This introduces the notion of evolution in a specific direction (left to right in this example). Finally, we observe that the extent of $\mathcal{D}_h$ progressively spreads from left to right starting at p. This means that the information contained in the perturbation does not instantly distribute in the x direction but instead propagates at a finite speed. This characteristic speed defines the boundary of $\mathcal{D}_h$, shown in Figure 1.1 with a dashed line.

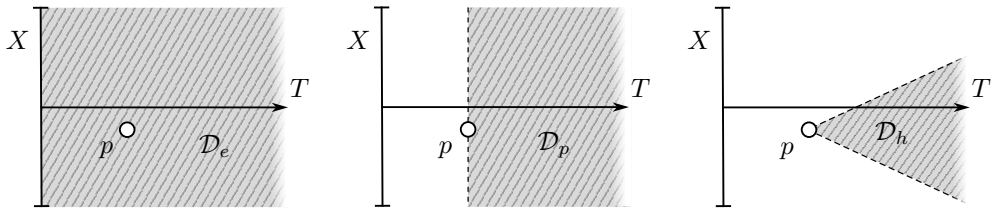


Figure 1.1: *Domains in which the solution changes after introducing a point perturbation in an elliptic (left), parabolic (center), and hyperbolic (right) problem*

The nature of the target problem is of primary importance when choosing a numerical method for its solution. The method must be capable of reproducing the system's

natural behavior as faithfully as possible. Over the years, methods have been proposed with specific characteristics that make them suitable for solving certain sets of problems. The need for increasingly powerful tools has led to the development of methods optimized for specific problems or even particular configurations.

This thesis aims to establish a methodology applicable to the simulation of hyperbolic systems in a broad sense. Although the model hyperbolic equation is the wave equation, we have chosen to focus on first order hyperbolic PDEs such as the advection problem and, primarily, the electromagnetic problem for its development and validation. The latter is of particular interest since its vectorial nature presents an added challenge and, if it is properly formulated, many other problems can be derived from it.

1.2.1.1. The role of spatial discretization and geometry approximation

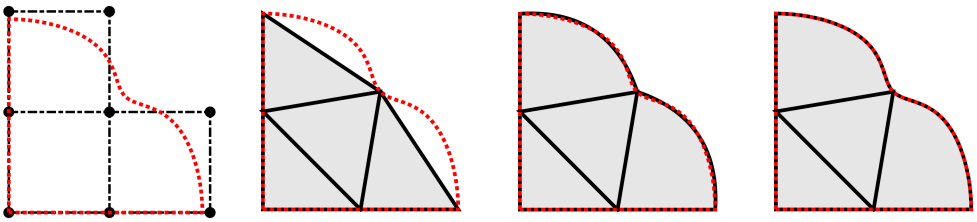
The finite-difference (FD) time-domain method (FDTD) presented by Yee [1] and developed by Taflove [2] has remained the predominant technique within the field of computational electromagnetics for the solution of Maxwell's equations in the time domain. However, in its simplest implementation, the FDTD method needs a structured mesh which is not capable of accurately representing complex geometries. In the simple example presented in Fig. 1.2, the dash dotted segments represent the links between the locations where the solution is computed. The FD approximation of the domain is nothing but the region closed by the outer segments while the target domain was that closed by the red dotted line. This domains are very dissimilar due to the structured distribution of evaluation points. Refining the grid reduces this gap but the staircase representation of the geometry can't be circumvented.

A variety of methods that propose the combination of the FDTD with finite element (FE) or finite volume (FV) methods has been studied [3, 4]. FE and FV methods possess improved geometry representation capabilities. See Fig. 1.2b to realize that even a coarse FE approximation of the domain composed by four elements (grey triangles) is clearly closer to the real geometry than the FD counterpart. However, none of them has overcome the main limitations of the FDTD scheme. These combined approaches also rely on a piecewise linear approximation of the boundary which originates numerical diffraction phenomena. Furthermore, the low-order interpolation of the solution introduces excessive numerical dispersion and/or dissipation, which limits its scope of application to simulations where the electromagnetic wave propagates along just a few wavelengths.

High-order methods are a powerful alternative to Yee's algorithm. The polynomial approximation of the geometry allows the boundary to be captured with greater precision which reduces the effect of the non-physical diffraction. In Fig. 1.2c, the computational and real domains almost match when polynomial subdomains are considered while keeping the number of elements in the mesh very low. Likewise, by

increasing the degree of the approximation, the dispersion and dissipation errors inherent to the numerical method are reduced.

The accurate representation of the geometry is critical in the framework of high-order methods since it can become the limiting factor in terms of accuracy. Previous research showed that the approximate representation of the geometry must be at least of the same order as the solution in order to maintain the convergence rate of the method [5]. In this regard, the use of isoparametric approaches with curved elements has traditionally been preferred even though some alternative methods exhibit better performance [6, 7]. The NURBS-Enhanced FE method (NEFEM) [5] is one of the strategies that belong to this second group. The ingredients that let NEFEM outperforms standard high-order isoparametric approaches are the consistent polynomial approximation of the solution and a more flexible representation of the geometry. NEFEM is a nodal method that constructs the shape functions directly in the physical space while uses advanced numerical integration strategies that consider a piece-wise rational representation of the domain. Enabling this geometric models is very convenient since most computer aid design (CAD) softwares use it, meaning that most domains are exactly captured. This is the case in Fig. 1.2d.



(a) *FD* ***(b)*** *Linear FE* ***(c)*** *Cuadratic FE* ***(d)*** *NEFEM*

Figure 1.2: *Computational domain decomposition derived for the same physical domain (red dotted line) as function of the numerical approach*

In any case, the spreading of all these approaches is limited, and the reason lies in the scarcity of algorithms capable of providing suitable decompositions. The automatic generation of curvilinear high-order meshes with good characteristics is an open problem, which is still challenging for complex geometries in three dimensions [8]. In the context of NEFEM, automated mesh generation procedures was only available for 2D geometries [9] until very recently when a 3D extension of the algorithm was proposed [10].

1.2.1.2. Optimality of high-order discontinuous Galerkin schemes

Among all the available high-order methods, FE techniques based on the discontinuous Galerkin (DG) formulation have gained increasing interest since their appearance in the 70s [11, 12]. These strategies exhibit some desirable features that alternative

approaches do not. Unlike for high-order FV or FD methods, the order of accuracy of the approximation depends on trial functions that are locally defined on each element. This featured is shared with standard FE methods, but DG methods alleviate the continuity constrain of the approximation between elements. Adopting the weak counterpart of this constrain, DG methods let easily adapt the order of the approximation at the element level regardless of the discretization used in its surroundings. Furthermore, when combined with explicit time marching schemes, the mass matrix arising from the weak formulation of the problem is block diagonal, which means that the global system can be split into a set of element-by-element problems that can be computed in parallel [13, 14]. Taking advantage of the multi-core architectures of modern CPUs, this feature significantly speeds up the computations. Finally, the use of numerical fluxes to weakly impose the continuity of the solution provides an easy way to stabilize the numerical scheme for convection dominated problems.

The combination of DG and explicit Runge-Kutta methods leads to algorithms with a high convergence rate in both space and time [12]. However, explicit methods obey the Courant-Friedrichs-Levy (CFL) [15] condition that limits the maximum size of the time step that can be used. Furthermore, the overall CFL condition of a numerical approach depends on local properties of the discretization. In meshes with very different element sizes, due to local h-refinement, the smallest element defines the maximum size of the time step. In the literature, a variety of methods that seeks to minimize the effect of this dependency has been proposed. Strong-stability preserving Runge-Kutta methods [16, 17] or the multistep Runge-Kutta methods [18, 19, 20] are some examples. As an alternative we may consider implicit methods [21, 22]. These time integrators do not suffer from instability issues even for large step sizes but the cost to pay is the solution of the global system of equations. The matrix of the system to be solved each time step is no longer block-diagonal. Hybrid methods called locally implicit Runge-Kutta methods that seek to combine the best of both worlds have been also proposed [23, 24, 25, 26].

1.2.1.3. Spatial discretization alternatives

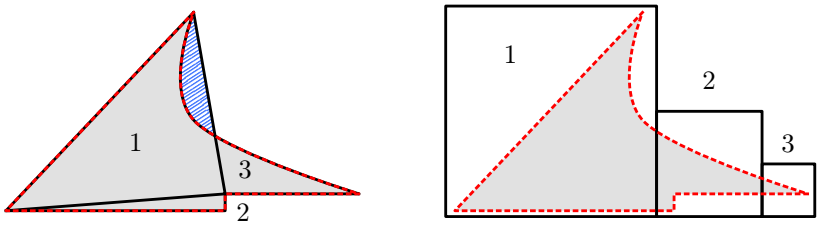
It has already been mentioned that mesh generation is a crucial stage in FE simulations. It is well-known that the base used to approximate the solution sets an upper bound to the achievable accuracy of the output. The degree, size and shape of the elements affects the base quality. Regarding the later, the accuracy of calculations deteriorates as the shape of the elements within the mesh deviates from the shape of reference elements. When analyses involve domains with complex boundaries, it can be particularly challenging to find a decomposition into subdomains suitable as a starting point for defining the approximation basis. Typically, elements located in the outer layers of the mesh exhibit highly distorted shapes and dissimilar sizes when attempting to reproduce the boundary of the physical domain. Indeed, within the high-order context, it is not uncommon to find tangled elements that are unsuitable for inclusion in the computational mesh. This is an inherent issue of conforming FE

methods and the target of advanced mesh generators [8, 9, 10] is to discover new algorithms that optimize the geometry and distribution of elements to each problem setting.

Non-conforming methods, also known as fictitious domain methods (FDM), tackle the mesh generation problem from a fundamentally different perspective. Instead of constructing an analysis mesh that conforms to the actual problem domain, FDM computes the mesh on a simpler, easy-to-mesh shape that completely contains the domain of interest. Domain decomposition remains necessary but is deferred to a later stages of the analysis workflow, where a broader range of techniques can be employed without adversely affecting the quality of the approximation basis. Broadly speaking, FDM approaches first construct the basis and then capture the physical domain by assigning different properties to each point within an element based on its location [27, 28].

Fig. 1.3 illustrates two discretizations, each consisting of three elements, for the same domain: one conforming and the other non-conforming. The conforming discretization shown in Fig. 1.3a exhibits the issues previously discussed. First, element 2 possesses a side significantly shorter than the others, as it must accommodate a geometric feature whose size is small compared to the characteristic length of the mesh elements. Second, element 3 is an example of tangled element. It is severely distorted to the extent that its curved side intersects the interface shared with element 1. This configuration is invalid because it implies that the solution is not unique within the region shaded in blue, where both elements 1 and 3 overlap. Additionally, this overlapping occurs in an area external to the domain (outlined by the dotted red line) in which the problem is defined.

In contrast, the non-conforming discretization shown in Fig. 1.3b consists of elements with regular shapes — in this case, squares. The approximation bases are defined considering these polygons, while the physics of the problem is exclusively evaluated in the gray-shaded region, precisely matching the domain where the problem is properly defined.



(a) *Conforming quadratic* ***(b)*** *Non-conforming*

Figure 1.3: *Comparison of*

The Cartesian grid finite element method (cgFEM) [29] falls within the FDM and its key feature is the use of a mesh made up of regular quadrilateral (2D) or regular hexahedral (3D) elements. This meshes are constructed through a recursive subdo-

main splitting process that leads to a spatial discretization that presents very convenient characteristics form the computational point of view. Firstly, the similarity of elements can be leveraged to evaluate, at the element-level, magnitudes and operators much more efficiently. Indeed, under certain conditions, traditionally expensive calculations can even be reduced to scalings of results obtained for a reference element. Second, the hierarchy of the mesh can be used to project information among meshes through h-refinement processes preventing from redundant calculations. Third, the structure of the mesh simplifies tasks related to locating points in elements or finding connectivities between entities of the mesh. The cgFEM has been successfully applied for the solution of both image and geometry based structural analysis in a range of problem such as linear elasticity [30], plasticity [31], contact [32, 33, 34], structural optimization [29, 35, 36], image-based analysis [37, 38] and bone remodeling controlled by mechano-regulatory models [39].

1.2.1.4. Numerical integration strategies

The cgFEM, as with other FDMs, employs sophisticated integration techniques to evaluate the weak form of operators over the corresponding problem domain. In the literature, there is a wide range of numerical integration techniques available, and each FE implements a specific one depending on the characteristics of the method and the particular problem being addressed. Generally speaking, integrals defined over complex domains are numerically approximated in three different ways:

- *Computing an equivalent integral over a reduced-dimensional domain*: Results from calculus and functional analysis can be applied to cast the original integral to a lower-dimensional space where its value is computable. In this category, we find strategies that use the divergence theorem and/or Green's theorem [40, 41, 42, 43, 44, 45] or Euler's theorem [46, 47, 48, 49]. For its application, it is necessary to work with integrands that have an analytic antiderivative [50], apply symbolic computation techniques, or approximate the result through numerical methods [51].

- *Applying the moment fitting method*: Moment fitting methods construct a system of equations that allows obtaining the coordinates and weights of the integration points. In its general setting, the resulting system is nonlinear on the coordinates which makes its resolution computationally expensive [52, 53]. Many authors circumvent this drawback fixing the evaluation points in advance [48, 54, 55] and solving linear systems of equations to compute the weights. The right-hand side of these systems consists of the integrals of a family of functions, and to evaluate them, it is often necessary to resort to any of the alternative integration strategies. Additionally, although there are works that impose additional restrictions to avoid it [56, 57, 58, 59], the generated quadratures are likely to contain points outside the domain and negative weights. The advan-

tage of moment fitting strategies is that they allow achieving quadratures with a reduced number of points, which are fast to evaluate.

- *Partitioning the original domain into simpler subdomains*: We can distinguish strategies where the subdomains are obtained through the recursive division of the element [53, 60, 61] and those with a coarser decomposition into subdomains conforming to the geometry [29, 62, 63, 64]. The former give rise to quadratures with a high density of points near the boundary and require corrections to avoid the staircase/linear representation of the geometry [65, 66]. Alternatively, the strategies that divide the integration domain into a set of simpler conforming subdomains result in more efficient quadratures at the cost of more expensive subdivision strategies and sophisticated mappings, which in many cases do not guarantee desirable properties such as non-negativity of the integration weights. The strategies falling in this category on the decomposition of the original domain are the most widely used.

1.2.2. Towards efficient solvers with automated optimization of the discretization

The performance of numerical analysis methods is closely tied to the selection of an appropriate approximation basis. Many studies have focused on developing automatic mesh adaptation methods tailored to each problem, numerical scheme, and analysis objective, see [67, 68] for a comprehensive overview. The essential components of an adaptation process are the identification of error sources and the definition of an adaptation strategy.

1.2.2.1. Error measurement

Although estimators for quantities of interest (usually based on solving the dual problem) are gaining increasing attention, in this work, we are particularly interested in procedures that consider global errors [69, 70]. The methodologies proposed in this line can be grouped into two categories: higher-order methods and residual methods.

Within the higher-order methods, we can distinguish extrapolation methods, such as Richardson's method [71, 72, 73, 74], which use analytical results about the behavior of approximations to infer more accurate solutions from uniformly refined solutions. These methodologies are simple and can achieve high accuracy. However, they rely on the use of a set of uniformly refined approximation in the asymptotic range of the scheme, which may become highly computationally demanding. In the category of higher-order methods, we also find recovery methods which locally compute a field derived from the solution with higher precision than the original. It is achieved either through the use of an intelligent sampling strategy [75, 76, 77] or by imposing constraints that the analytical solution must satisfy [78, 79, 80]. These methods address local problem that can be solve efficiently, mostly if parallel computing is applied.

The accuracy of the recovered field depends on the regularity of the exact solution, which make these methods fail in the presence of singularities.

Residual-based methods characterize the error in the solution by studying the violation of the differential equation governing the problem. Explicit methods [81, 82, 83] study the residual in the weak form, taking into account only the original approximation to obtain indirect approximations of the error in some norm, whereas implicit methods [67, 81, 83, 84] solve an additional problem to obtain the error distribution using an enriched approximation. Although usually lacking in accuracy, explicit methods often provide very fast indicators that are sufficient to get the error distribution needed to guide adaptation procedures. Conversely, the strength of implicit methods is their high accuracy and reliability but the computational cost associated to the resolution of the residual problem can become very high.

1.2.2.2. Alternatives strategies in mesh adaptation

Among the adaptation strategies [85, 86, 87], most implementations focus on h-refinement strategies [88, 89, 90, 91] due to their simplicity, maturity, flexibility, and better performance when applied to solutions with singularities. The p-version of the FE [82, 92, 93] is less common because its implementation is more complex and their range of application is limited. However, it is well-known that for the same number of degrees of freedom, high-order approximations are more efficient. When working with smooth solutions, p-refinement provides geometric convergence with respect to the number of degrees of freedom whereas h-refinement methods offer algebraic convergence [94, 95]. When solution regularity reduces, the p-FE methods also exhibits algebraic convergence but it has been proved that the rate of convergence is never smaller than that of the h-FE counterpart.

Fig. 1.4 is a result gathered from the seminal paper [96] that supports this statement. The model problem is the edge cracked square panel under plane strain conditions subjected to uniform tension and the quantity of interest is the relative error in the strain energy W. It is shown that, even for non-smooth solutions, the p-version outperforms the h-version since the practical rate of convergence of the former is twice of the later.

The combination of the h and p approaches, known as hp-adaptation methods [97, 98], is a promising area that is gaining increasing attention. However, implementing such procedures is complex. For them to be effective, in addition to the error distribution, a local estimate of the solution's regularity is needed. The most widespread approach relies on a multigrid strategy [99], where a uniformly refined approximation is solved, either in h [100], p [101, 102], or both [103]. Another common strategy specially suited to spectral methods is to express the solution as a series of functions such as Legendre [104, 105, 106], studying the decay rate of the coefficients to determine the optimal refinement strategy.

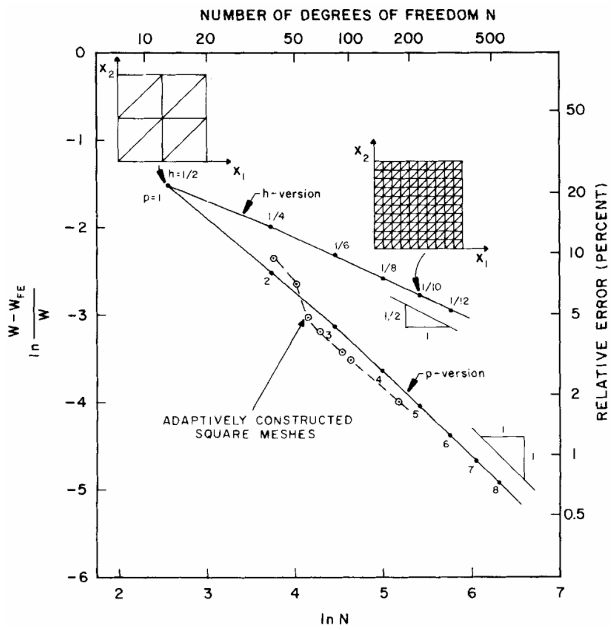


Figure 1.4: *Pioneer result extracted from [96] on the convergence of the h-version and p-version of the FE when applied to a problem with non-smooth analytical solution.*

1.3. Objective

The ultimate goal of this thesis is to develop a computational methodology specifically tailored for solving systems of hyperbolic PDEs —with a particular interest in the transient electromagnetic problem— in domains featuring complex geometries. To this end, both high-order DG methods and cgFEM exhibit highly valuable characteristics at different stages of the computational process. On one hand, high-order discontinuous approximations have proven particularly suitable for wave propagation problems, delivering the best results in terms of precision and efficiency among all alternatives reported in the literature. On the other hand, the cgFEM method proposes a meshing strategy that greatly simplifies the generation of approximation bases, while implementing advanced integration techniques capable of efficiently capturing the geometry of the domains. In this thesis, these two technologies are taken as a starting point and merged to obtain a new method. The aim is not only to retain the strengths they individually possess but also to discover and exploit synergies, so that the proposed methodology becomes a significant step forward in terms of flexibility, efficiency, and reliability.

Several intermediate objectives, valuable in their own right, have been identified. Achieving these will provide the necessary components to construct the desired methodology:

Obj.0 ***Implementation of high-order DG formulations within cgFEM.*** The first objective is to achieve an optimal combination of DG formulation, high-order element approximations, and geometry-independent Cartesian meshes. The goal is to build a framework that fully exploits the synergies among these technologies and serves as a solid, flexible foundation for future developments, both within and beyond the scope of this thesis.

Obj.1 ***Development of advanced integration techniques.*** The integration techniques used must be accurate enough not to limit the convergence of the method. We aim to design new strategies that allow the computation of integration quadratures capable of evaluating high-order integrals in domains expected to have complex and highly varied geometries. While efficiency remains a consideration, the focus is on ensuring that the proposed schemes consistently provide stable (strictly positive weights), easy-to-evaluate integration quadratures, while minimizing the discrepancies between the numerical and actual integration domains.

Obj.2 ***Development of stabilization techniques.*** It is well known that FDM suffers from stability issues since the solution is poorly conditioned on those elements barely intersecting the computational domain. Additionally, the minimum size of subdomains into which the computational domain is divided determines the stability of time integration schemes. The proposed method is not immune to these sources of instability. Indeed, the development of stabilization techniques have been developed to counteract them is a major concern. Given the general-purpose nature of the method being pursued, the techniques are designed to be applicable regardless of the modeled physical phenomena.

Obj.3 ***Development of automatic discretization adaptation procedures.*** The performance and competitiveness of numerical methods rely heavily on the efficient use of computational resources. This involves performing demanding computations or storing large volumes of data only when strictly necessary. In this context, mesh adaptation strategies have proven to be highly valuable tools. In this thesis, we aim to provide the tools that allow continuously adjusting the discretization in response to the evolving characteristics of the solution at each simulation time step, without requiring user intervention. Achieving this objective involves two key steps. First, it is necessary to define computationally inexpensive error indicators that offer meaningful insights into the error distribution. Second, optimizing both the polynomial order and the support of the approximation basis requires inferring additional properties of the exact underlying solution.

1.4. Outline of the thesis

In this introductory chapter, the motivation for the project has been presented, the objectives have been outlined, and a review of the most relevant works within each of the fields covered by this thesis has been conducted. Chapter 2 begins with the exposition of the model problem selected to develop the methodology, followed by brief introduction to the discontinuous Galerkin formulation and discretization with Cartesian grids.

The Cartesian grid Discontinuous Galerkin method (cgDG) is presented in Chapter 3. First, the formulation is developed at a general level and its particularities are studied to propose computational strategies that maximize the potential of the scheme (Obj.0). Then, the design of accurate integration quadratures (Obj.1) and the stabilization (Obj.2) of the spatial discretization for solving 2D problems are explained in depth.

The extension of the methodology to 3D problems is addressed in Chapter 4. This chapter is structured into three parts. First section is concerned about adapting the geometrical models for their analysis with Cartesian meshes (Obj.0). The next two sections revisit integration (Obj.1) and stabilization (Obj.2) proposing new methodologies that address the challenges posed by the introduction of the third spatial dimension.

In Chapter 5, first steps toward an automatic mesh adaptation scheme (Obj.3) are presented. Various alternatives for error quantification within the cgDG context applied to hyperbolic problems are studied. Two error indicators are derived, which are used to guide the mesh adaptation process in 2D simulations. The regularity of the solution is highly relevant in the adaptation process; estimates are presented and discussed.

Chapters 3 to 5 form the core of this thesis. All of them include a final section containing numerical analyses that have let us validate the proposed strategies and understand the effect of various model parameters. Likewise, numerical results for relevant problems are included to show the overall performance of the methodology.

The thesis concludes with Chapter 6. There, a range of final comments and conclusions regarding the proposed technologies are gathered. Some research lines that have emerged from this project and represent interesting objectives for future work are also identified.

At the end of the document, two appendices are attached. In Appendix A, the details of the formulation used to obtain projections within reference domains are presented. Appendix B is a compendium that includes many of the functions we have implemented for handling geometric entities, as well as the necessary information for their use. It serves as a reference that will accompany the toolbox made available to the scientific community.

Chapter 2

Foundations of the Cartesian grid discontinuous Galerkin method

This chapter provides a brief overview of the foundations upon which the Cartesian grid Discontinuous Galerkin (cgDG) method is built. First, the electromagnetic problem, the physical problem used as a model to develop and validate the methodology, is introduced. In Section 2.1, the equations governing electromagnetic phenomena are presented, along with some mathematical tools and manipulations that are useful for their simulation using numerical methods in a broad sense. These mathematical tricks have been implemented within the cgDG framework —with minor variations introduced to maximize its performance— and some of the application results presented in later chapters make use of them.

Next, Section 2.2 presents the Discontinuous Galerkin (DG) formulation of the model problem. After deriving it for a generic form of the model problem -applicable to a wider range of physics-, the DG formulation is then specified for the electromagnetic case. The relationships defined in Section 2.1 are revisited, and their equivalents within the numerical analysis strategy are provided. The DG formulation adopted here involves only spatial discretization, with an integrator for initial value problems (IVP) being responsible for advancing the solution in time. Relevant remarks on this topic are included in Section 2.2.2. The combination of these two technologies

provides significant advantages to the cgDG method but is also the source of the instability issues addressed in Sections 3.5 and 4.3.

Finally, the particularities of the discretization proposed in the Cartesian grid Finite Element Method (cgFEM) are presented. In Section 2.3, the mesh generation process is briefly introduced, and how it induces structure and hierarchy in the discretization —features that will be leveraged in the remaining chapters to enhance cgDG performance. The importance of integration in cgFEM is highlighted in Section 2.3.1, where the different types of integrals that appear within this methodology and their specific characteristics are reviewed. Section 2.3.2 concludes this section dedicated to cgFEM with some remarks on the advantages offered by Cartesian meshes when solving problems in infinite domains, a recurring situation in the context of computational electromagnetics.

2.1. Statement of the model problem

The numerical simulation of hyperbolic systems of partial differential equations (PDEs) is at the core of this thesis. In general terms and without delving into mathematical detail, hyperbolic PDEs are characterized by modeling systems whose state at any point near a hypersurface of the problem domain can be computed based on the state and its lower-order derivatives on that hypersurface. Taking the model hyperbolic equation, the 1D wave equation:

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0, \tag{2.1}$$

this means that it is possible to calculate the value of the conserved quantity u at a point in the domain (t, x) from the value of the field $u|_{t_0}$ and its first temporal derivative $(\partial u/\partial t)|_{t_0}$ at a specific instant t_0. This statement will always hold in a neighborhood of t_0 and it can be extended to any (t, x) if the solution is regular enough. Furthermore, this solution is characterized by a finite information propagation speed with magnitude c.

First-order PDEs also satisfy the properties of hyperbolic equations. Adopting Einstein's notation, a system of first-order hyperbolic PDEs in a domain $\mathrm{T} \times \Omega \in \mathbb{R}^{n+1}$ can be expressed as:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_k(\mathbf{U})}{\partial x_k} = \mathbf{S}(\mathbf{U}) \tag{2.2}$$

where $\mathbf{U}$ is the vector of conservative variables, $\mathbf{F}_k(\mathbf{U})$ is the flux in the k-th direction and $\mathbf{S}(\mathbf{U})$ is the vector gathering all source terms. This terminology is not arbitrary but follows from the connection between first-order PDEs and conservation laws. By integrating over Ω, applying the divergence theorem and assuming a certain level of regularity of the solution with respect to t, we arrive at:

$$\frac{d}{dt}\int_{\Omega}\mathbf{U}d\Omega+\int_{\partial\Omega}\mathbf{F}_k\left(\mathbf{U}\right)d\Gamma_k=\int_{\Omega}\mathbf{S}\left(\mathbf{U}\right)d\Omega \tag{2.3}$$

where one can clearly identify the terms corresponding to variation, transport, and generation/sink of the conserved quantities, which are characteristic of conservation laws.

The scientific and technical value of these relationships is enormous due to their generality. Indeed, they are fundamental laws of nature that enable the modeling of a vast number of phenomena regardless of their specific features. However, as highlighted in Sections 1.1 and 1.2, there are still many open challenges regarding their resolution with numerical methods.

In this thesis, we address the simulation of first-order hyperbolic PDE systems like the one presented in (2.2). This general structure is specified for the electromagnetic problem due to its wide variety of practically relevant applications and because it introduces the additional challenge that the variables involved in the equations are vector fields. Without loss of generality, the proposed methodology is developed and validated using this physics.

Remark. Simpler problems involving scalar quantities can be easily handled using the same strategies, and their formulation is derived in a similar way as for the electromagnetic problem. These less computationally demanding applications will be of interest when evaluating specific aspects of the methodology in Section 4.4. However, their specific formulation is not presented here for the sake of brevity.

2.1.1. The electromagnetic problem

2.1.1.1. Maxwell's equations

Maxwell's equations describe the relationship between the components of the electromagnetic field at those points where the field vectors are single-valued, bounded, continuous functions of position and time, and exhibit continuous derivatives. The Maxwell's equations in differential form write:

$$\nabla\times\mathbf{H}=\frac{\partial\mathbf{D}}{\partial t}+\mathbf{J}_i+\mathbf{J} \tag{2.4}$$

$$\nabla\times\mathbf{E}=-\frac{\partial\mathbf{B}}{\partial t}-\mathbf{M}_i \tag{2.5}$$

$$\nabla\cdot\mathbf{B}=q_m \tag{2.6}$$

$$\nabla\cdot\mathbf{D}=q_e \tag{2.7}$$

where we can distinguish Maxwell's curl equations on the top and Maxwell's divergence equations on the bottom, and the related quantities are:

- **E**: electric field intensity vector
- **D**: electric flux density vector
- $\bar{\mathbf{J}}$: impressed electric current density vector
- **J**: conduction electric current density vector
- q_e: electric charge density
- **H**: magnetic field intensity vector
- **B**: magnetic flux density vector
- $\bar{\mathbf{M}}$: impressed magnetic current density vector
- q_m: magnetic charge density

Since it is a particular case of first order hyperbolic PDE, Maxwell's equations can be expressed as conservation laws. If V_j represents the j component of the vector $\mathbf{V}$, the electromagnetic quantities in the dimensionless conservative form are:

$$\mathbf{U} = \begin{Bmatrix} D_1 \\ D_2 \\ D_3 \\ B_1 \\ B_2 \\ B_3 \end{Bmatrix}, \mathbf{F}_1 = \begin{Bmatrix} 0 \\ H_3 \\ -H_2 \\ 0 \\ -E_3 \\ E_2 \end{Bmatrix}, \mathbf{F}_2 = \begin{Bmatrix} -H_3 \\ 0 \\ H_1 \\ E_3 \\ 0 \\ -E_1 \end{Bmatrix}, \mathbf{F}_3 = \begin{Bmatrix} H_2 \\ -H_1 \\ 0 \\ -E_2 \\ E_1 \\ 0 \end{Bmatrix}, \mathbf{S} = \begin{Bmatrix} \bar{J}_1 + J_1 \\ \bar{J}_2 + J_2 \\ \bar{J}_3 + J_3 \\ \bar{M}_1 \\ \bar{M}_2 \\ \bar{M}_3 \end{Bmatrix} \quad (2.8)$$

When formulating Maxwell's equations, it is common to eliminate the terms $\bar{\mathbf{M}}$ and q_m since they are quantities without a physical equivalent that are introduced in the equations to achieve symmetry. On a practical level, they are used as a mathematical tool to ease the study of certain phenomena such as the macroscopic magnetic behavior of material or the electric current distribution in antennas. By the surface equivalence theorem [107], actual sources are replaced by equivalent ones that produce the same fields within a certain region. This approach makes the formulations of scattering and diffraction problems more amenable to approximations.

2.1.1.2. Constitutive laws

When the electromagnetic field passes through a material medium, it interacts with the charged particles within it. In this way, currents appear inside the material that affect the propagation of electromagnetic waves, causing them to behave differently than they would in a vacuum. The constitutive equations are responsible for modeling the macroscopic effect of the presence of these charged particles. The effect of the medium on the propagation of the electromagnetic wave is characterized based on three constitutive parameters: permittivity $\hat{\boldsymbol{\varepsilon}}$, permeability $\hat{\boldsymbol{\mu}}$, and conductivity $\boldsymbol{\sigma}$. These parameters relate electrical and magnetic quantities according to:

$$\mathbf{D} = \hat{\boldsymbol{\varepsilon}}\mathbf{E} \tag{2.9}$$

$$\mathbf{B} = \hat{\boldsymbol{\mu}}\mathbf{H} \tag{2.10}$$

$$\mathbf{J} = \boldsymbol{\sigma}\mathbf{E} \tag{2.11}$$

In the general case, these are parameters that vary from one point to another and are dependent on the intensity, frequency content, and propagation direction of the field. However, in this thesis, only linear, isotropic and non-dispersive materials with constant constitutive parameters $\hat{\varepsilon}$, $\hat{\mu}$ and σ are analyzed. The values of permeability and permittivity are usually expressed with respect to vacuum permittivity $\hat{\varepsilon}_0$ and permeability $\hat{\mu}_0$:

$$\hat{\varepsilon} = \varepsilon\hat{\varepsilon}_0 \quad , \quad \hat{\mu} = \mu\hat{\mu}_0 \tag{2.12}$$

where ε and μ are the relative permittivity and relative permeability of the medium, respectively.

2.1.1.3. Boundary conditions

At those points where there is an abrupt change in the properties of the propagation medium or in the current density vectors, the fields present discontinuities, and Equations (2.4) to (2.7) cannot be applied directly. To complete the modeling of the electromagnetic problem, boundary conditions are introduced that establish the relationship between the fields on either side of the discontinuity.

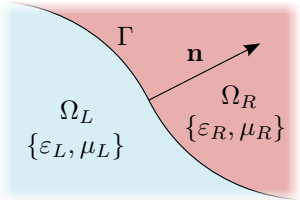


Figure 2.1: *Material interface diagram*

First, let us consider a **material interface** such as this presented in Fig. 2.1 where two media with constitutive parameters $\{\varepsilon_L, \mu_L\}$ and $\{\varepsilon_R, \mu_R\}$ meet. Let us assume that the shared boundary Γ is free of any charge or source and that $\mathbf{n}$ is its normal vector pointing from Ω_L to Ω_R. From weak form of Maxwell's equations, it is derived that the tangential components of the electric and magnetic field intensity vectors are continuous across the interface:

$$\mathbf{n} \times (\mathbf{E}_L - \mathbf{E}_R) = 0 \quad , \quad \mathbf{n} \times (\mathbf{H}_L - \mathbf{H}_R) = 0 \tag{2.13}$$

The continuity condition is also satisfied by the normal components of the electric and magnetic flux density vectors:

$$\mathbf{n} \cdot (\mathbf{D}_L - \mathbf{D}_R) = 0 \quad , \quad \mathbf{n} \cdot (\mathbf{B}_L - \mathbf{B}_R) = 0 \tag{2.14}$$

Even though the exact solution of the problem must satisfy both restrictions, (2.13) applies to Maxwell's curl equations while (2.14) applies to Maxwell's divergence equations.

A special case of material interface of particular interest in scattering problems (see Section 2.1.2.2 for further details) is that implying a **perfect electric conductor** (PEC). A PEC is an object made of a hypothetical material with infinite conductivity. This feature supposes that every electric field that falls upon surface of the object is reflected and no electric energy is absorbed into the PEC. If material medium on the R side of the interface is a PEC, the electric field vanishing on the boundary formulates:

$$\mathbf{n} \times \mathbf{E}_L = \mathbf{0}. \tag{2.15}$$

2.1.2. Towards numerical simulation

This section presents strategies that enable a more efficient approach to analyzing certain configurations of interest. These are generic strategies that are largely independent of the method used to solve Maxwell's equations. In fact, they rely on manipulating the problem formulation to obtain an equivalent version that offers some advantage, with discretization being deferred to a later stage. The selection of strategies presented here has been implemented within the cgDG method and will be referred to throughout this document.

2.1.2.1. Infinite domain approximation

Computations over exceptionally large domains are not uncommon in the context of computational electromagnetics. Electrically charged bodies cause electromagnetic field disturbances that can be felt at distances several orders of magnitude greater than the size of the body generating them. Understanding the morphology of the disturbed field, its global characteristics, or its interactions with other bodies located in its zone of influence is the aim of highly relevant practical problems such as radiation of bodies, scattering, antenna design, or characterization of metamaterials.

The difference in scale between the characteristic lengths of the domain of interest, the solution, and the involved bodies necessitates numerical approximations that can become practically intractable. These analyses must employ resolutions rich enough to capture the smallest scales, which implies that covering the entire domain would require handling an immense amount of information. Fortunately, many of these

problems admit simplifications that allow for a reduction in computational scale, making them solvable using a reasonable amount of resources.

In the results section of Chapters 3 to 5, the cgDG method is applied to problems of this type. In particular, results are provided for scattering problems and problems where the solution exhibits a periodic structure. The following sections present the mathematical tools that enable the resolution of these specific problems.

2.1.2.1.1. Radiation

Many studies in electromagnetism investigate the effects of introducing a local heterogeneity into a domain where an electromagnetic field with analytical expression is assumed. First, the aim is to understand the morphology and magnitude of the field variations in the vicinity of the heterogeneity. This is the region where the specific characteristics of the heterogeneity play the most important role, and understanding their pointwise effect on the electromagnetic field allows analysts to better grasp the physical phenomenon. As we move away from the heterogeneity, the specific field values typically lose importance, and the focus shifts to computing global quantities that characterize it. In propagation media with uniform properties, these quantities can be computed from the field values near the heterogeneity, making simulation of the entire domain unnecessary.

Practically, these analyses are addressed by restricting the computational domain Ω to the small portion of the physical domain Ω_{ph} that surrounds the medium's heterogeneity and by imposing boundary conditions on the fictitious boundaries that replicate the behavior of a wave propagating indefinitely. Applied to scattering problems (see Section 2.1.2.2 for further details), we focus solely on the field component originating from the heterogeneity, namely the scattered field. To ensure uniqueness of the solution in an exterior domain of Ω, the boundary condition on these artificial boundaries needs to approximate the Silver-Müller radiation conditions on the scattered field:

$$\lim_{\|\mathbf{x}\|_2 \to \infty} \left(\mathbf{x} \times (\nabla \times \mathbf{E}) + \|\mathbf{x}\|_2 \frac{\partial \mathbf{E}}{\partial t} \right) = \mathbf{0},$$
$$\lim_{\|\mathbf{x}\|_2 \to \infty} \left(\mathbf{x} \times (\nabla \times \mathbf{H}) + \|\mathbf{x}\|_2 \frac{\partial \mathbf{H}}{\partial t} \right) = \mathbf{0}$$

These equations simply state that the scattered field consists of outgoing waves only. Indeed, if we consider a point located on a sphere centered at the scatterers, we can see that the electromagnetic fields represent plane waves propagating outwards in the radial direction, whose amplitude decays as r^{-1} due to the sphericity.

Low order approximations of the conditions partially reflect the outgoing waves leading to spurious oscillations of the field vectors. The approximation is improved adding a **perfectly matched layer** (PML). The PML extends Ω in all directions and applies in each subdomain of the extensions specific variations of the material

properties such that the electromagnetic waves are transmited into the artificial layer without reflection toward the interior of the computational domain while they attenuate as they traverse the PML regions approaching the outer boundary. Here, we give a brief explanation on how we implemented the PML condition. We recommend the reader to consult [108, 109, 110, 111, 112, 113] for further details on artificial boundary conditions for truncated infinite domains.

The standard formulation of the PML for a 3D scattering problem without sources is:

$$\begin{aligned}
\frac{\partial \mathbf{U}_1}{\partial t} + \frac{\mathbf{F}_1(\mathbf{U})}{\partial x_1} &= \frac{\sigma_1}{\varepsilon}\mathbf{U}_1 \\
\frac{\partial \mathbf{U}_2}{\partial t} + \frac{\mathbf{F}_2(\mathbf{U})}{\partial x_2} &= \frac{\sigma_2}{\varepsilon}\mathbf{U}_2 \\
\frac{\partial \mathbf{U}_3}{\partial t} + \frac{\mathbf{F}_3(\mathbf{U})}{\partial x_3} &= \frac{\sigma_3}{\varepsilon}\mathbf{U}_3
\end{aligned} \tag{2.16}$$

where the vector of conservative variables $\mathbf{U}$ has been split as a sum of three terms: $\mathbf{U}_1$,$\mathbf{U}_2$ and $\mathbf{U}_3$; and σ_1,σ_2 and σ_3 are three variables that let the user control the attenuation of the waves in the PML domain. The definition of σ_i is addressed later in this section.

It is clear that the field splitting in the core of PML formulation implies solving a larger system of equations. In the 3D case, the number of equation seems to grow from 6 to 18. However, notice that 6 of the previous equations are trivial since they impose that the i component of the electric and magnetic field vectors in $\mathbf{U}_i$ are zero. A similar reasoning applies to 2D problems. Therefore, the resulting PML formulation implies solving a system of 12 and 4 equations in 3D and 2D cases, respectively. The equations in (2.16) can be rearranged so the final system becomes the original Maxwell's equations accompanied by a set of additional equations.

The interface of the PML with the domain of interest must coincide with a constant coordinate surface in the reference system used to formulate (2.16). Therefore, in our case, the boundary we artificially impose on Ω_{ph} must be an orthogonal box (a rectangle in the 2D case) with its faces parallel to the Cartesian planes. The outer boundary of the PML does not need to meet any geometric constraints. However, it is common, and in our particular case convenient, for the outer boundary of the PML to be obtained as a shift of the inner boundary in one direction of the reference system. The PML domain $\bar{\Omega}$ is divided into regions according to their position with respect to Ω. In Fig. 2.2, an example of the truncation of an infinite domain in Ω and its extension with the PML is shown. The decomposition of the PML according to Cartesian directions gives rise to 8 regions $\bar{\Omega}_{ij}$, where the subscripts ij show the position with respect to Ω in the x_1x_2 directions.

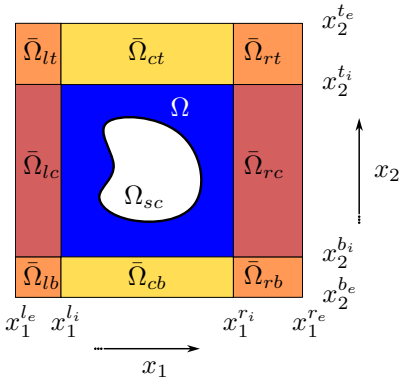


Figure 2.2: *Layout of the PML regions for a 2D scattering problem.*

For this problem, the variables σ_1 and σ_2 are defined in such a way that the waves attenuate once they leave Ω and reach the outer edge of $\bar{\Omega}$ with minimal amplitude without affecting the field in Ω. Each set of equations in (2.16) causes wave attenuation in one of the Cartesian directions; the system of equations involving the flow in direction i characterizes how the wave amplitude evolves in direction i. The variables σ_i are usually defined as continuous increasing functions with a zero value in Ω. In our implementation, they will depend on two parameters, σ_0 and n, which represent the maximum conductivity and the degree of absorption, respectively. The formulation for a two-dimensional problem like the one presented in Fig. 2.2 would be:

$$\sigma_1(x_1) = \begin{cases} \sigma_0 \left(\frac{x_1 - x_1^{j_i}}{x_1^{j_e} - x_1^{j_i}} \right)^n & \text{if} \quad x_1 \in \bar{\Omega}_{jk} \text{ with } j \in \{l, r\} \\ 0 & \text{if} \quad x_1 \in \bar{\Omega}_{ck} \end{cases} \tag{2.17}$$

$$\sigma_2(x_2) = \begin{cases} \sigma_0 \left(\frac{x_2 - x_2^{j_i}}{x_2^{j_e} - x_2^{j_i}} \right)^n & \text{if} \quad x_2 \in \bar{\Omega}_{jk} \text{ with } k \in \{t, b\} \\ 0 & \text{if} \quad x_2 \in \bar{\Omega}_{jc} \end{cases} \tag{2.18}$$

For the σ_3 parameter in the three-dimensional case, σ_3 will have a similar expression.

The standard formulation of the PML with the division of the field into two artificial components is only weakly well-posed [114] meaning that the norm of the solution is bounded (up to exponential growth in time) not only by the norm of the initial data, but also by the norm of the initial spatial derivatives. This can cause the problem to become ill-posed when small changes are introduced to the initial conditions. To eliminate this potential source of instability, the PML formulation can be changed to another type as presented in [115]. The **strongly-well posed PML** eliminates the need to divide the fields into artificial components and instead makes use of first and second-order temporal integrals of the electromagnetic field.

Let us study this alternative formulation for a problem involving a linear isotropic electrical conductor medium with a conductivity value σ. We will also assume that

the medium's permittivity and permeability are not time-varying. The formulation proposed in [115], like the original PML, introduces a set of parameters σ_i that allow control over wave attenuation. In this case, the parameters acting on the same component of the electric and magnetic field are considered equal, $\sigma_i = \sigma_{i+3}$. The governing equations are expressed in terms of the auxiliary variables $\widetilde{\mathbf{U}}$, which are obtained from the vector of conservative variables $\mathbf{U}$ and its first temporal integral $\mathbf{U}^{(1)}$:

$$\widetilde{\mathbf{U}} = \mathbf{U} + \mathbf{W}\mathbf{U}^{(1)} \tag{2.19}$$

where $\mathbf{W}$ is the matrix containing the parameters σ_i on its diagonal.

The parameters σ_i appear in several terms of the reformulation of Maxwell's equations, acting linearly on $\mathbf{U}$ and its temporal integrals. Using:

$$\sigma_A^B = \sum_{i \in A} \sigma_i - \sum_{j \in B} \sigma_j \tag{2.20}$$

the matrix form of these operators is:

$$\mathbf{W}_0 = -\mathrm{diag}\left(\sigma + \sigma_{2,3}^1, \sigma + \sigma_{1,3}^2, \sigma + \sigma_{1,2}^3, \sigma_{2,3}^1, \sigma_{1,3}^2, \sigma_{1,2}^3\right) \tag{2.21}$$

$$\mathbf{W}_1 = -\mathrm{diag}\left(\sigma\sigma_{2,3}^1 + \sigma_1^2\sigma_1^3, \sigma\sigma_{1,3}^2 + \sigma_2^1\sigma_2^3, \sigma\sigma_{1,2}^3 + \sigma_3^1\sigma_3^2, \sigma_1^2\sigma_1^3, \sigma_2^1\sigma_2^3, \sigma_3^1\sigma_3^2\right) \tag{2.22}$$

$$\mathbf{W}_2 = -\mathrm{diag}\left(\sigma\sigma_2\sigma_3, \sigma\sigma_1\sigma_3, \sigma\sigma_1\sigma_2, 0, 0, 0\right) \tag{2.23}$$

Finally, the system of equations to be solved is:

$$\frac{\partial \widetilde{\mathbf{U}}}{\partial t} + \frac{\partial \mathbf{F}_k\left(\widetilde{\mathbf{U}}\right)}{\partial x_k} = \mathbf{W}_0\mathbf{P}\widetilde{\mathbf{U}} + \mathbf{W}_1\mathbf{P}^2\mathbf{U}^{(1)} + \mathbf{W}_2\mathbf{P}^3\mathbf{U}^{(2)} \tag{2.24}$$

where $\mathbf{P}$ is the diagonal matrix that relates the vector of conservative variables to the electromagnetic field intensities according to:

$$\mathbf{P}\mathbf{U} = \begin{Bmatrix} \mathbf{E} \\ \mathbf{H} \end{Bmatrix} \tag{2.25}$$

and $\mathbf{U}^{(2)}$ represents the second temporal integral of the vector of conservative variables.

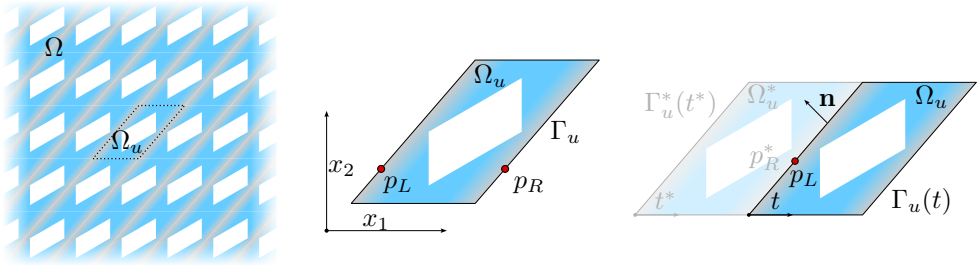
Remark. In [116], a study on the effect of the PML parameters on the effectiveness of its standard formulation can be found. A PML design that presents a good balance between accuracy and computational load is a domain extension $d \in [\lambda, 2\lambda]$, a quadratic absorption degree $n = 2$, and a maximum conductivity σ_0 defined based on d as $\sigma_0 = 60d^{-1}$.The role and physical meaning of the parameters σ_i for the strongly well posed formulation is identical to that used in the standard PML. The PML is also split into regions regarding the reference directions and the same distribution of σ_i applies, meaning the recommended values for the domain extension, degree absorption and maximum conductivity are the aforementioned ones.

2.1.2.1.2. Periodicity

Periodic boundary conditions (PBC) are commonly introduced to solve problems defined over very large domains that exhibit an internal repetitive structure characterizing the solution. In such cases, simulating the global problem is impractical from a computational standpoint and the strategies presented in the previous section are not applicable since the media is non-uniform. However, a local problem can be formulated over the unit cell —this internal structure that repeats periodically throughout the domain— which is computationally manageable and provides all the necessary information to construct the solution over the full domain. Imposing PBC on the fictitious boundaries of the unit cell enables these calculations.

In Fig. 2.3, the global and local versions of the same problem are shown. The domain Ω in Fig. 2.3a extends indefinitely in all directions, but its internal structure follows a pattern. The juxtaposition of the domain Ω_u (a subdomain outlined with a dotted line) ends up generating a domain identical to Ω, meaning that Ω_u is a unit cell of Ω. If the solution to the problem depends solely on the internal structure of the domain, it will exhibit the same repetitive pattern and can be computed over Ω_u by applying the appropriate boundary conditions.

In Fig. 2.3b, the problem posed on the unit cell is presented. The boundary $\Gamma_u(t)$ is a fictitious boundary that did not exist in the original problem and upon which PBCs must be imposed to recover the global solution. If we look at the points p_L and p_R on Γ_u, we see that p_L would spatially coincide with p_R of the neighboring cell immediately to the left of the depicted cell; by symmetry, p_R in Ω_u would also coincide with point p_L of the cell to the right. Therefore, the periodicity condition at p_L can be imposed using the formulation for a material interface, considering that the electromagnetic field outside Ω_u is the one present at p_R.



(a) Infinite periodic domain *(b) Unit cell* *(c) Periodic boundary condition*

Figure 2.3: *Local problem derived for an infinite domain with periodic internal structure*

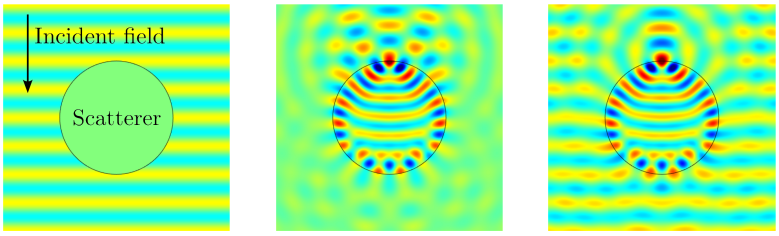
Following notation in Fig. 2.3c, let us use an asterisk to identify the entities and quantities associated with a hypothetical neighboring cell Ω_u^* viewed from a generic point p_L on Γ_u whose local coordinate on the boundary is t_L. Geometrically, it is possible to identify another point p_R^* on Γ_u^* that coincides with p_L in space and whose local coordinate on the boundary is t_R^*. For periodicity to hold, the value of any

quantity must be the same at p_R^* and at $p_R = \Gamma_u(t_R^*)$. Once the mapping $t_L \to t_R^*$ is established, the definition of the PBC is complete. One only needs to impose (2.13) considering that:

$$\mathbf{E}_L = \mathbf{E}\left(\Gamma_u\left(t_L\right)\right) \ , \ \mathbf{E}_R = \mathbf{E}\left(\Gamma_u\left(t_R^*\right)\right)$$
$$\mathbf{H}_L = \mathbf{H}\left(\Gamma_u\left(t_L\right)\right) \ , \ \mathbf{H}_R = \mathbf{H}\left(\Gamma_u\left(t_R^*\right)\right)$$

2.1.2.2. Scattered formulation

Scattering problems aim to study the behavior of an incident electromagnetic wave when it hits an obstacle, known as the scatterer. The main objective is to determine both the total field and the scattered field. The later being the portion of the total field resulting from the interaction between the incident wave and the obstacle. In Fig. 2.4, a simple illustrative example is presented. It shows the scattering of a downward-propagating plane wave as it encounters a dielectric cylinder (see Fig. 2.4a). Fig. 2.4b and Fig. 2.4c respectively show the perturbations in the original field caused by the presence of the obstacle and the total field. These kind of problems are particularly important, as they are at the core of many technologies and applications such as medical imaging, non-destructive testing, remote sensing, and wireless communications, to name a few.



(a) *Incident field* **(b)** *Scattered field* **(c)** *Total field*

Figure 2.4: *Relevant fields in the scattering of a dielectric cylinder*

In many scattering simulations, the underlying physical phenomenon is linear, which allows for the application of the superposition principle. In such cases, distinguishing between the incident fields $\{\mathbf{E}^i, \mathbf{H}^i\}$ and the scattered fields $\{\mathbf{E}^s, \mathbf{H}^s\}$, and expressing the total electromagnetic field $\{\mathbf{E}^t, \mathbf{H}^t\}$ as their sum, becomes both possible and convenient. Substituting:

$$\mathbf{E} = \mathbf{E}^t = \mathbf{E}^s + \mathbf{E}^i \ , \quad \mathbf{H} = \mathbf{H}^t = \mathbf{H}^s + \mathbf{H}^i \tag{2.26}$$

into (2.2), the original problem can be approached as two independent sub-problems: one focused on the propagation of the incident wave, and the other on computing the scattered field. The first sub-problem is resolved in advance, as the selected incident

wave must be compatible with Maxwell's equations. Thus, the simulation focuses solely on solving the second part. In this approach, the influence of the incident wave on the scattered field is modeled using the analytical representation of the incident wave. This methodology eliminates the need to propagate the incident field from the boundary of the computational domain to the scatterer, thereby reducing computational cost and enhancing solution accuracy. This alternative treatment is known as the scattered formulation (see [2] for further details).

When examining the details of the formulation, the structure of the problem remains the same. The equations in (2.2) still hold, but some terms are modified. First, the scattered field $\{\mathbf{E}^s, \mathbf{H}^s\}$ becomes the new unknown field $\{\mathbf{E}, \mathbf{H}\}$, which means the components E_j and H_j in (2.8) now refer to the j-th components of the scattered electric and magnetic field vectors. Second, the effect of the incident field interacting with the medium is accounted for by adding the following term to the source vector:

$$\mathbf{S}^i = \begin{Bmatrix} (1-\varepsilon)\dfrac{\partial E_1^i}{\partial t} \\ (1-\varepsilon)\dfrac{\partial E_2^i}{\partial t} \\ (1-\varepsilon)\dfrac{\partial E_3^i}{\partial t} \\ (1-\mu)\dfrac{\partial H_1^i}{\partial t} \\ (1-\mu)\dfrac{\partial H_2^i}{\partial t} \\ (1-\mu)\dfrac{\partial H_3^i}{\partial t} \end{Bmatrix} \tag{2.27}$$

Finally, suitable boundary conditions must be derived from those outlined in Section 2.1.1.3. This is a straightforward task, given that the original expressions apply to the total field and the linear relation (2.26).

Remark. It is important to note that, according to the notation used here, the unknown field is denoted without subscripts. The adopted formulation in each context determines whether it refers to the total field or the scattered field. The nature of the analysis dictates the formulation used, and it will only be specified explicitly in cases where ambiguity exists and the distinction is relevant.

2.1.2.3. Transverse modes of the electromagnetic field

When solving electromagnetic problems analytically, it is common to refer to modes to identify the different field configurations that satisfy the boundary conditions and are solutions of the initial value problem. The modes that appear most frequently in these analyses are the so-called transverse modes, which are characterized by the fact that either the electric field, the magnetic field, or both are always perpendicular to a certain direction. These modes are referred to as transverse electric (TE), transverse magnetic (TM), and transverse electromagnetic (TEM), respec-

tively; and they naturally arise in the study of waveguides, transmission lines, cavities, or radiation phenomena. Closely related to transverse modes is the polarization of electromagnetic fields, which has countless consequences and applications, notably in lasers, wireless and optical fiber telecommunications, and radar.

From the perspective of numerical analysis, restricting the possible solutions of a problem to only transverse modes allows for a significant simplification in its simulation. In fact, this is one of the necessary conditions for addressing real electromagnetic problems through 2D simulations. More specifically, the z dimension of a 3D problem can be eliminated from the analysis by restricting it to the 2D case if the following conditions are met:

a) The geometry is invariant w.r.t. z.

b) The field is transverse electric TE^z or transverse magnetic TM^z w.r.t. z.

c) The properties of the propagation medium are invariant w.r.t. z. Additionally, the medium possesses some internal symmetry and is oriented in such a way that it does not cause couplings that would invalidate the transverse field hypothesis.

d) The sources respect the invariance in z.

The isotropic medium assumption introduced in Section 2.1.1.2 induces more symmetry in the results. For a TE^z problem, not only the E_3 component of the electric field is null, but the H_1 and H_2 components of the magnetic field are so. Substituting the known quantities into Maxwell's equations, it can be seen that some of them become trivial, and the number of equations in the PDE system can be reduced. Under the TE^z mode assumption, the vector in Equation (2.8) becomes:

$$\mathbf{U} = \begin{Bmatrix} \varepsilon E_1 \\ \varepsilon E_2 \\ \mu H_3 \end{Bmatrix} \quad , \quad \mathbf{F}_1 = \begin{Bmatrix} 0 \\ H_3 \\ E_2 \end{Bmatrix} \quad , \quad \mathbf{F}_2 = \begin{Bmatrix} -H_3 \\ 0 \\ -E_1 \end{Bmatrix} \quad , \quad \mathbf{S} = \begin{Bmatrix} S_1 \\ S_2 \\ S_6 \end{Bmatrix} \tag{2.28}$$

In TM^z problems an equivalent result is obtained but, in this case, the vectors in Equation (2.8) are:

$$\mathbf{U} = \begin{Bmatrix} \mu H_1 \\ \mu H_2 \\ \varepsilon E_3 \end{Bmatrix} \quad , \quad \mathbf{F}_1 = \begin{Bmatrix} 0 \\ -E_3 \\ -H_2 \end{Bmatrix} \quad , \quad \mathbf{F}_2 = \begin{Bmatrix} E_3 \\ 0 \\ H_1 \end{Bmatrix} \quad , \quad \mathbf{S} = \begin{Bmatrix} S_4 \\ S_5 \\ S_3 \end{Bmatrix} \tag{2.29}$$

2.2. Discontinuous Galerkin formulation

In this section, we give a brief description of the DG formulation applied to a system of first order partial differential equations in general and Maxwell's equation in particular. The reader is referred to [12, 117, 118, 119] for further information.

2.2.1. General setting: conservative laws

Starting from the general form of conservation laws using Einstein's notation (2.2), we aim to derive their DG formulation. Let us consider a bounded domain Ω and its decomposition into a set of non-overlapping subdomains Ω_e in such a way that $\Omega = \bigcup_e \Omega_e$. We locally approximate the solution via a polynomial interpolation defined by a set of nodal values on Ω_e. The functions used to interpolate constitute a basis of the space of polynomials up to degree p in Ω_e, denoted by $\mathcal{P}_p(\Omega_e)$. The test functions considered to derive the weak form are polynomials also contained in the space $\mathcal{P}_p(\Omega_e)$ and they are gathered in a vector of the same size as $\mathbf{U}$, namely $\mathbf{W}$. Multiplying (2.2) by $\mathbf{W}$, integrating over Ω_e and integrating by parts, we obtain the following weak form in an element:

$$\int_{\Omega_e} \mathbf{W} \cdot \frac{\partial \mathbf{U}_e}{\partial t} d\Omega - \int_{\Omega_e} \frac{\partial \mathbf{W}}{\partial x_k} \cdot \mathbf{F}_k(\mathbf{U}_e)\, d\Omega + \int_{\partial\Omega_e} \mathbf{W} \cdot \mathbf{F}_{\mathbf{n}}(\mathbf{U}_e)\, d\Gamma = \int_{\Omega_e} \mathbf{W} \cdot \mathbf{S}(\mathbf{U}_e)\, d\Omega \quad (2.30)$$

where $\mathbf{U}_e$ represents the restriction of the vector of conservative variables $\mathbf{U}$ to the element domain and $\mathbf{F}_{\mathbf{n}}$ represents the flux in the normal direction $\mathbf{n}$, being $\mathbf{n}$ the outward unit normal vector to the boundary $\partial\Omega_e$:

$$\mathbf{F}_{\mathbf{n}}(\mathbf{U}) = \mathrm{n}_k \mathbf{F}_k(\mathbf{U}) \quad (2.31)$$

To account for the discontinuous nature of the approximation, the normal flux is replaced by a numerical normal flux $\boldsymbol{\mathcal{F}}_{\mathbf{n}}(\mathbf{U}_e, \mathbf{U}_{\text{out}})$ that depends not only on the solution within the element $\mathbf{U}_e$ but also depends on the solution on the neighbouring element $\mathbf{U}_{\text{out}}(\mathbf{x})$, defined as:

$$\mathbf{U}_{\text{out}}(\mathbf{x}) = \lim_{\delta \to 0^+} \mathbf{U}(\mathbf{x} + \delta\mathbf{n}) \quad \forall \mathbf{x} \in \partial\Omega_e \quad (2.32)$$

Choosing the numerical flux is a crucial step as it dictates the stability and the accuracy of the numerical scheme. Some of the most widespread approaches consist of using a centred flux, an upwind flux or a linear combinations of the two previous schemes, see [120, 121, 122, 123, 124, 125]. For linear hyperbolic problems, such as the one considered here, a typical choice consists of using a flux splitting approach.

This formulation is based on dividing the flux across the boundary of the element into outgoing (superscript +) and incoming (superscript -) fluxes regarding the positive and negative eigenvalues of the Jacobian matrix $\mathbf{A_n} = \partial \mathbf{F_n} / \partial \mathbf{U}$, respectively:

$$\mathbf{F_n}(\mathbf{U}) = \mathbf{F_n^+}(\mathbf{U}) + \mathbf{F_n^-}(\mathbf{U}) \tag{2.33}$$

The upwind numerical flux is computed by considering that the incoming component of the flux is due to the distribution of electromagnetic field on the outside, while the outgoing flux depends on the distribution inside the element:

$$\boldsymbol{\mathcal{F}}_\mathbf{n}(\mathbf{U}_e, \mathbf{U}_{\text{out}}) = \mathbf{F_n^+}(\mathbf{U}_e) + \mathbf{F_n^-}(\mathbf{U}_{\text{out}}) \tag{2.34}$$

Substituting (2.34) in (2.30) and integrating by parts again, we obtain the following weak form in an element:

$$\int_{\Omega_e} \mathbf{W} \cdot \frac{\partial \mathbf{U}_e}{\partial t} d\Omega + \int_{\Omega_e} \mathbf{W} \cdot \frac{\partial \mathbf{F}_k(\mathbf{U}_e)}{\partial x_k} d\Omega + \int_{\partial\Omega_e} \mathbf{W} \cdot \left[\boldsymbol{\mathcal{F}}_\mathbf{n}(\mathbf{U}_e, \mathbf{U}_{\text{out}}) - \mathbf{F_n}(\mathbf{U}_e)\right] d\Gamma = \int_{\Omega_e} \mathbf{W} \cdot \mathbf{S}(\mathbf{U}_e)\, d\Omega \tag{2.35}$$

Considering the jump operator $[\![\mathbf{G}(\mathbf{U}_e)]\!] = \mathbf{G}(\mathbf{U}_{\text{out}}) - \mathbf{G}(\mathbf{U}_e)$, in which $\mathbf{G}$ represents any operator, the difference between fluxes can be simplified leading to:

$$\int_{\Omega_e} \mathbf{W} \cdot \frac{\partial \mathbf{U}_e}{\partial t} d\Omega + \int_{\Omega_e} \mathbf{W} \cdot \frac{\partial \mathbf{F}_k(\mathbf{U}_e)}{\partial x_k} d\Omega + \int_{\partial\Omega_e} \mathbf{W} \cdot [\![\mathbf{F_n^-}(\mathbf{U}_e)]\!] d\Gamma = \int_{\Omega_e} \mathbf{W} \cdot \mathbf{S}(\mathbf{U}_e)\, d\Omega \tag{2.36}$$

Note that the integral of the flux over the boundary penalizes the difference between the incoming flux due to the internal and external fields, namely the incoming flux jump. For an internal edge, both fields are elemental solutions and the weak continuity of the global solution is imposed as a result. For an external edge, i.e. an edge over $\partial\Omega$, the outer state is not defined. As usual in the framework of hyperbolic equations, the Rankine-Hugoniot jump conditions are employed to compute the incoming flux jump that, when integrated over the boundary, translates into a weak enforcement of the boundary conditions. In Section 2.2.3 we will particularize this formulation to the electromagnetic problem and gather the specific expressions used to weakly impose the most relevant boundary conditions.

2.2.2. Semi-discrete system

The next step towards the numerical resolution of (2.24) would be to define the basis or bases of the space $\mathcal{P}_p(\Omega_e)$ where $\mathbf{W}$ and $\mathbf{U}$ live and, with it, the unknowns

of the problem. Using a nodal approach like the one adopted by the methodology proposed in this thesis, evaluating the integrals present in (2.36) and grouping terms, one arrives at a system of ordinary differential equations. The structure of this system is:

$$\mathbf{M}\frac{d\mathbf{U}^h}{dt} + \mathbf{R}\left(\mathbf{U}^h\right) = \mathbf{0} \tag{2.37}$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{U}^h$ is the vector which contains the nodal values of the conservative variables and $\mathbf{R}\left(\mathbf{U}^h\right)$ represents the vector of residuals.

Remark. For more details on the terms involved in (2.37), refer to Section 3.1, where their formulation is derived taking into account the characteristics introduced by discretization with Cartesian meshes. However, it is worth noting that the structure presented here is general and appears regardless of the bases chosen for $\mathbf{W}$ and $\mathbf{U}$.

Equation (2.37) is integrated in time using an explicit Runge-Kutta type method. The use of high-order explicit integrators of this type decouples the system of equations (2.37). The splitting of the global problem into a set of smaller local ones has important advantages from the computational point of view, such as a great reduction in the size of the systems of equations to be solve, leading to faster calculations and memory saving, and the possibility of applying parallel processing. However, for the algorithm to behave in a stable way, it is necessary to set a time step for the integration that complies with the CFL stability condition, which, in the particular case of the classic Runge-Kutta, is formulated:

$$\Delta t \leq \mathcal{C}\frac{h}{c\,p^2} \tag{2.38}$$

being $\mathcal{C}$ a positive constant, h a length that quantifies the minimum size of the spatial discretization, c the speed of light in the medium and p the degree of the polynomial approximation.

2.2.3. Application to the electromagnetic problem

After presenting the DG formulation of the conservative problem in its general form, the following sections will address the specific aspects of its application to the resolution of the electromagnetic problem. Specifically, expressions are provided to compute the numerical fluxes that need to be considered at the interfaces of the problem. The starting point for these expressions are the equations presented in Sections 2.1.1.3 and 2.1.2.1, which are combined with the Rankine-Hugoniot jump condition to derive the upwind formulation of the problem.

2.2.3.1. Boundary conditions

From (2.36), we notice that every subdomain Ω_e in Ω interacts with its environment through the jump in the incoming flux $\mathbf{F}_{\mathbf{n}}^{-}(\mathbf{U})$. Once the eigenvalue problem for the Jacobian matrix $\mathbf{A}_{\mathbf{n}}$ is solved, we arrive to the following expression of $\mathbf{F}_{\mathbf{n}}^{-}$:

$$\mathbf{F}_{\mathbf{n}}^{-}(\mathbf{U}) = \frac{1}{2}\begin{Bmatrix} -\mathbf{n}\times\mathbf{H} + \sqrt{\frac{\varepsilon}{\mu}}(\mathbf{n}\times(\mathbf{n}\times\mathbf{E})) \\ +\mathbf{n}\times\mathbf{E} + \sqrt{\frac{\mu}{\varepsilon}}(\mathbf{n}\times(\mathbf{n}\times\mathbf{H})) \end{Bmatrix} \tag{2.39}$$

With this result, we conclude that to impose a certain boundary condition we need to associate to it values for the their tangential components of the electric and magnetic field intensity vectors, or directly the values of the incoming flux.

Remark. In Section 2.1, we noted that appropriate boundary conditions for Maxwell's curl equations require continuity in the tangential components of the electric and magnetic fields. Since the DG formulation was stated from (2.2) - an alternative formulation of Maxwell's curl equations - it is reasonable to expect that similar conditions on the tangential components apply to its boundary treatment. This expectation is confirmed in (2.39).

When the properties of the media change across the interface, the boundary conditions can not be plugged directly into (2.39) and equivalent values applicable on each side of the interface must be computed in advance. The equivalent tangential components of the electromagnetic field intensity vector are computed imposing the Rankine-Hugoniot jump condition across the characteristics in the phase plane.

Let us consider the domain Ω_L with constitutive parameters $\{\varepsilon_L, \mu_L\}$ that meets the domain Ω_R with material properties $\{\varepsilon_R, \mu_R\}$. Let us assume that the shared boundary is free of any charge or source and that $\mathbf{n}$ is its normal vector pointing from Ω_L to Ω_R. The boundary condition that applies for this **material interface** in the continuous setting is (2.13). The compliance of Rankine-Hugoniot jump conditions leads to the equivalent values that will be plugged into this relation for the DG scheme:

$$\mathbf{n}\times\mathbf{E}_{\text{out}} = \mathbf{n}\times\frac{c_L\varepsilon_L\mathbf{E}_L - \mathbf{n}\times\mathbf{H}_L + c_R\varepsilon_R\mathbf{E}_R + \mathbf{n}\times\mathbf{H}_R}{c_L\varepsilon_L + c_R\varepsilon_R} \tag{2.40}$$

$$\mathbf{n}\times\mathbf{H}_{\text{out}} = \mathbf{n}\times\frac{c_L\mu_L\mathbf{H}_L + \mathbf{n}\times\mathbf{E}_L + c_R\mu_R\mathbf{H}_R - \mathbf{n}\times\mathbf{E}_R}{c_L\mu_L + c_R\mu_R} \tag{2.41}$$

where $c_L = \frac{1}{\sqrt{\varepsilon_L\mu_L}}$ and $c_R = \frac{1}{\sqrt{\varepsilon_R\mu_R}}$ are the propagation velocities of the electromagnetic wave in each medium. Computing the boundary condition applicable from the other side of the interface is straightforward.

Dirichlet or **periodicity** boundary conditions are treated as material interfaces. When these boundary conditions are considered, the electromagnetic field beyond the contour of Ω_L is known so they can be directly plugged into (2.40).

Scattering problems usually deal with **perfect electric conductors** (PEC). Using the scattering formulation, the electric field vanishing on the boundary formulates:

$$\mathbf{n} \times \mathbf{E}_L + \mathbf{n} \times \mathbf{E}_L^i = \mathbf{0}. \tag{2.42}$$

This expression is enough to constrain the electric field in the DG setting:

$$\mathbf{n} \times \mathbf{E}_{\text{out}} = \mathbf{0} \quad \text{and} \quad \mathbf{n} \times \mathbf{E}_{\text{out}} = -\mathbf{n} \times \mathbf{E}_L^i$$

for the total and scattered formulation, respectively. To define the tangential component of the magnetic field further information is needed. Knowing that the permittivity of a PEC is unbounded, the imposition of Rankine-Hugoniot jump conditions drives to:

$$\mathbf{n} \times \mathbf{H}_{\text{out}} = \mathbf{n} \times \mathbf{H}_L + \mathbf{n} \times \frac{\mathbf{n} \times \mathbf{E}_L - \mathbf{n} \times \mathbf{E}_{\text{out}}}{c_L \mu_L} \tag{2.43}$$

2.2.3.2. Infinite domain approximation

The physical domain Ω_{ph} may be both a closed domain or an infinite domain. When Ω_{ph} is an infinite domain, the computational domain Ω is restricted to the scatterers and its surroundings. The boundary condition on these artificial boundaries needs to approximate the Silver-Muller radiation conditions, it is the scattered field must consist of outgoing waves only. We use a first order **absorbing** boundary condition (ABC) that imposes null incoming flux at the outer boundary, namely

$$\mathbf{F}_{\mathbf{n}}^{-}(\mathbf{U}_{\text{out}}) = \mathbf{0}. \tag{2.44}$$

The approximation is improved adding a **perfectly matched layer** (PML) that minimizes waves reflections on the artificial boundary of Ω. Both standard and strongly-well posed PMLs may be indistinctly used. In the DG framework, solving the extended system involved in the standard scheme requires to compute as many new numerical fluxes as equations were added. The interested reader is referred to [111] for a comprehensive derivation of these additional fluxes. Regarding the strongly-well posed formulation, it is not necessary to derive new numerical fluxes since the resulting system of equations is equivalent to Maxwell's equations plus some lower-order terms that act as source terms.

2.3. Cartesian grid Finite Element Method

As mentioned in Section 1.2, Cartesian grid Finite Element Method (cgFEM) is a Fictitious Domain Method (FDM) characterized by the use of meshes exclusively made of regular quadrilateral elements in 2D or regular hexahedral elements in 3D. In the following, we present the key ideas of this discretization strategy. Even though

they are also valid for 3D problems, in this exposition we focus on the 2D case, for the sake of simplicity.

The process to generate the analysis mesh starts defining the fictitious domain. In cgFEM, the fictitious domain is a squared domain that contains the physical domain, namely the bounding box $\mathcal{B}$. Next, a set of grids containing all possible elements that may appear in the analysis mesh is computed. The grid pile is constructed following an iterative procedure in which the current grid is computed by halving the elements of the previous grid in all space dimensions. The starting point is a grid that contains just an element of the same size as $\mathcal{B}$, which we refer to as the reference element. The procedure lasts for a number of iterations specified by the user which defines the minimum element size. Note that, if d is the number of spatial dimensions of the problem, the grid obtained after iteration l, namely the level-l grid, contains 2^{2l} identical elements which are similar to the reference element, that represents level-0.

Within this stage, each element may be identified by the virtual coordinates of its bottom-right vertex. The virtual coordinates system is a $\mathbb{R}^{d+1}$ coordinate system with its origin on the upper-left corner of the bounding box and whose basis, in relation to the Cartesian basis $\{\mathbf{e}_1, \mathbf{e}_2\}$, are $\{\bar{\mathbf{e}}_1(l), \bar{\mathbf{e}}_2(l), l\} = \left\{h(l)^{-1}\mathbf{e}_1, -h(l)^{-1}\mathbf{e}_2, l\right\}$, where $h(l)$ represents the size of the element for a given level l. In Figure 2.5, we present level-0 and level-1 grids and the virtual coordinates associated to its elements.

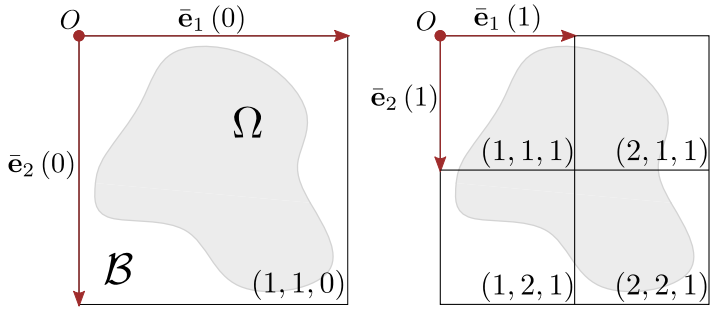


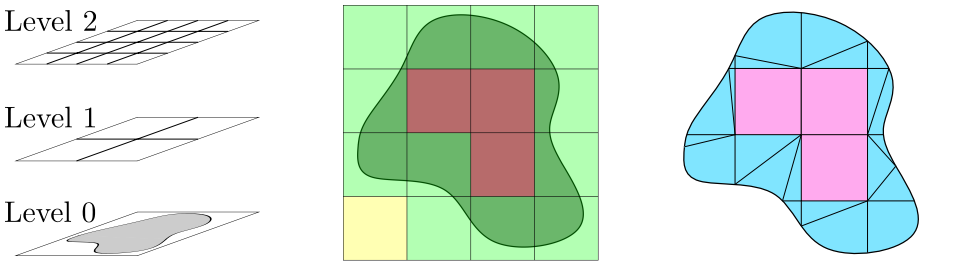
Figure 2.5: *Virtual coordinates associated with the level 0 (left) and level 1 (right) elements.*

The next step is to find the intersections of the geometry with the grid pile. Depending on the location of the elements with respect to the boundary of the physical domain, these are classified into external, internal and boundary elements. Considering the computational domain Ω, the element e with domain K_e is classified as:

- external element if $K_e \cap \Omega = \emptyset$.
- internal element if $K_e \cap \Omega = K_e$
- boundary element if $\{K_e \cap \Omega\} \supset K_e$

In Figs. 2.6a and 2.6b, we show the grid pile up to level 2 and the classification of the elements in the top level grid computed for an arbitrary domain.

The analysis grid is built up by selecting internal and boundary elements from the grid pile in such a way that there is no overlapping among elements and the physical domain is completely covered. This can be easily verified by using the virtual coordinates to explore the relations among elements in the grid pile.



(a) *Grid pile*

(b) *Element class depending on its relative position to the boundary: external, internal, boundary*

(c) *Problem domain split into integration subdomains: internal, boundary*

Figure 2.6: *Stages in analysis mesh generation*

To obtain the high order nodal distribution of an element, the nodal distribution of the reference element is mapped to the physical elements. Notice that, by construction, this mapping is affine. Therefore, the Jacobian matrix of the transformation is constant and there is no loss of consistency in the interpolation, i.e., a polynomial interpolation of degree p in local coordinates of the reference element preserves its degree when transferred to Cartesian coordinates. As exposed later, these features provide great advantages for numerical integration.

Taking as starting point the analysis mesh, we need to define a set of subdomains that fits the geometry, see Fig. 2.6c. These subdomains are employed to design the numerical quadratures used to evaluate the terms of the weak form of the problem. This way, the actual computational domain Ω is considered, instead of $\mathcal{B}$. The procedure used to obtain the integration subdomains element-wise depends on the element type. Internal elements are treated as standard elements and, therefore, the integration domain matches with the element domain. In the case of boundary elements, the portion of the physical domain within the element splits into simpler subdomains (triangles or quadrilaterals) and special integration quadratures are defined over each of them. In Section 2.3.1, this crucial aspect of FDM methods is studied in greater depth, highlighting specific features of cgFEM that distinguish it from other non-conforming methods.

Concerning the imposition of boundary conditions in cgFEM, the fact that the boundary is not conforming with the mesh means that there are some differences in comparison with the standard finite element method. Since the mesh nodes are not placed on the boundary, generally speaking, the essential boundary conditions can not be strongly imposed. Alternatively, they are imposed in weak form. In the case of Neumann boundary conditions, the integral along internal boundaries gives rise to terms in the RHS of the system of equations at all degrees of freedom (DOFs) of

the element. Unlike in the standard finite element methods, where these terms only appear on the DOFs tied to nodes located on the contour.

Processing Dirichlet boundary conditions is significantly more complex. There is a range of techniques that allow the imposition of Dirichlet conditions in a weak form [126, 127, 128], among which the penalty [129, 130], Lagrange multiplier [27, 131] or the Nitsche's [28, 132, 133, 134] methods stand out. In the cgFEM we use a strategy developed in our research group and which has been validated for the elastic problem [30]. The condition on the primal variable is imposed through a Lagrange multiplier field evaluated at the integration points of the curve defining the Dirichlet boundary of the physical domain. This field leads to an over-constrained system of equations, making it necessary to introduce a stabilization term in the formulation that ensures compliance with the inf-sup condition. To restrict the range of values that these multipliers can take and thus ensure optimal convergence, a stabilization term is added to the functional that penalizes the difference between the Lagrange multiplier field and a smoothed field on the dual variable obtained by post-processing of the solution of the problem. This stabilization strategy results in an iterative process that ensures optimal convergence to the problem's exact solution through refinement. The interested reader is addressed to [30, 31].

2.3.1. Comments on integration in cgFEM

There is a wide variety of techniques that can be applied for the evaluation of the weak form of operators in the context of FDM. In Section 1.2, the main characteristics of many of these techniques were discussed, and three categories were identified. They differ on how they approximate the problem: computing an equivalent integral over a reduced-dimensional domain; applying the moment fitting method; or partitioning the original domain into simpler subdomains. The cgFEM, in line with most FDM methods, implements different integration strategies that fall within this last category.

To formulate and understand the particularities of this approach, it is convenient to differentiate various reference coordinate systems. In Fig. 2.7, a graphical representation of these reference systems and their relations is shown. First, the physical coordinates system $\mathbf{x}$ is the one used to state the governing equations of the problem. Second, the reference element's coordinate system $\boldsymbol{\xi}$ used to construct the interpolation functions for the nodal values. Third, the reference domain's coordinate system $\boldsymbol{\tau}$ where precomputed numerical integration quadratures lives. When evaluating different magnitudes it is more convenient to work within a certain coordinate system. Therefore, the coordinate mappings between them need to be available. In the following, we note as $\mathbf{a} \rightarrow \mathbf{b}$ the mapping of elements in $\mathbf{a}$ into $\mathbf{b}$.

First remark concerns the mapping $\boldsymbol{\xi} \rightarrow \mathbf{x}$. Note that, because of the Cartesian structure of the mesh in the cgFEM, it is an affine transformation without rotation that formulates

$$\boldsymbol{\xi} \rightarrow \mathbf{x}: \quad \boldsymbol{\xi}_i \rightarrow \frac{1}{2}\mathbf{h}_e \odot \boldsymbol{\xi}_i + \mathbf{x}_e, \tag{2.45}$$

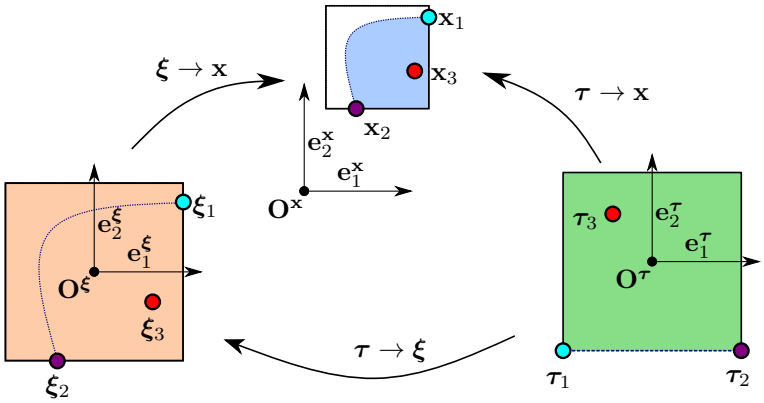


Figure 2.7: *Diagram showing the local, physical and integration spaces (from left to right) and their mappings. Circles of the same color represent the same point in each space while the images of the contour are plotted with doted lines.*

where $\mathbf{h}_e$ and $\mathbf{x}_e$ represent the size and the centre of the element e, respectively, and $\mathbf{a} \odot \mathbf{b}$ is the Hadamard product of the vectors $\mathbf{a}$ and $\mathbf{b}$. The element's size $\mathbf{h}_e$ is directly related to the size of the bounding box $\mathbf{h}_0$ used to generate the grid pile through the element's level l_e, which leads to:

$$\boldsymbol{\xi} \to \mathbf{x}: \quad \boldsymbol{\xi}_i \to \frac{\mathbf{h}_0 \odot \boldsymbol{\xi}_i}{2^{l_e+1}} + \mathbf{x}_e, \tag{2.46}$$

and its inverse:

$$\mathbf{x} \to \boldsymbol{\xi}: \quad \mathbf{x}_i \to 2^{l_e+1} (\mathbf{x}_i - \mathbf{x}_e) \odot \mathbf{h}_0^{\odot -1}. \tag{2.47}$$

where $\mathbf{a}^{\odot -1}$ is the Hadamard inverse of the vector $\mathbf{a}$.

Some important features of the cgFEM are directly related with the integration of the weak operator. To illustrate the distinguishing features, let us consider the weak form of the divergence operator acting on a vector field $\mathbf{u} : \mathbf{x} \to \mathbb{R}^d$, where d notes the dimension of the problem, for the k-th element of the mesh. If V and U are the spaces of test and trial functions, the bilinear form $a : U^d \times V \to \mathbb{R}$ that represents the weak divergence operator for the problem is:

$$a(\mathbf{u}, v) = \int_{\Omega_k} v \nabla_{\mathbf{x}} \cdot \mathbf{u} d\Omega \tag{2.48}$$

Let us consider the calculation of this integral when the spaces of test and trial functions are reduced to the space of polynomial up to degree p, namely $V, U \subset \mathcal{P}_p$. If those polynomials are defined as functions of the physical coordinates, meaning $\mathbf{x} = \boldsymbol{\xi}$, the integrand of the weak form in this same space is a polynomial of degree $2p-1$. Now we will study how the set of transformations introduced by finite element methods modifies the degree of the integral that is actually evaluated.

The most common approach is to numerically evaluate the integral using precomputed integration quadratures over a reference domain in $\boldsymbol{\tau}$, which requires evaluating the **integral over a domain other than the original**. This procedure introduces the transformation's Jacobian $\mathbf{J}_{\boldsymbol{\tau}\to\mathbf{x}}$ into the integrand to reflect the change in the measure. If we consider a polynomial transformation of the same order as the approximation, it is $\boldsymbol{\tau} \to \mathbf{x} \in \mathcal{P}_p^d$, we will obtain an integrand order equal to $(2p-1)(dp) + d(p-1)$.

$$a_{\boldsymbol{\tau}}(\mathbf{u}, v) = \int_{\Pi} v \nabla_{\mathbf{x}} \cdot \mathbf{u} \, |\mathbf{J}_{\boldsymbol{\tau}\to\mathbf{x}}| \, d\Pi \tag{2.49}$$

Let us consider the evaluation of the integral within a **standard finite element solver**. The isoparametric approximation of the solution implies another modification since the interpolation functions of the nodal values are constructed on a reference domain. Since space $\mathbf{x}$ and $\boldsymbol{\xi}$ are not equivalent any more, the evaluation of the divergence operator involves calculating the inverse of the Jacobian matrix of the transformation. The weak form becomes:

$$a_h(\mathbf{u}, v) = \int_{\Pi} v \left(\nabla_{\boldsymbol{\xi}} u_i\right) \left[\mathbf{J}_{\boldsymbol{\xi}\to\mathbf{x}}^{-1}\right]_{:,i} |\mathbf{J}_{\boldsymbol{\tau}\to\mathbf{x}}| \, d\Pi \tag{2.50}$$

where $\nabla_{\boldsymbol{\xi}}$ is the differential operator with respect to coordinate system $\boldsymbol{\xi}$, and $[\mathbf{A}]_{:,i}$ represents the i-th column of the matrix $\mathbf{A}$. The terms of the Jacobian matrix are again polynomials of degree $p-1$, which implies that its inverse will be rational. Consequently, the definition of the polynomial basis in a space other than the physical one makes it impossible to exactly evaluate the integrand of the final expression using standard quadratures for polynomials regardless of the space in which the integration is performed. In practice, quadratures for polynomials are still used, but the number of points is increased. Quadratures of degree $(2p-1) + d(p-1) + n$ are considered, where n will be a natural number that allows capturing the effect of the rational factor with sufficient precision.

Finally we study the particularities of the integration in cgFEM. First, let us consider **Cartesian elements entirely located within the computational domain**. As shown in (2.46), the transformation from the reference element system to the physical one $\boldsymbol{\xi} \to \mathbf{x}$ is affine. In particular, it corresponds to a translation and scaling of the global reference system, and thus the Jacobian matrix of the transformation and its inverse will be diagonal matrices with constant terms. This allows the exact evaluation of the integral value, even setting $n = 0$. The transformation from the integration space to the physical one $\boldsymbol{\tau} \to \mathbf{x}$ has exactly the same characteristics, so the determinant of its Jacobian matrix will also be a constant. This means that the integrand that is finally evaluated is a polynomial of degree $2p-1$. Thanks to the use of Cartesian elements, we can evaluate the operators without integration errors and with fewer evaluations of the integrand than in the general case. The cgFEM pair of expression 2.50 simplifies to:

$$
\begin{aligned}
a_{iC}(\mathbf{u}, v) &= \left[\mathbf{J}^{-1}_{\boldsymbol{\xi}\to\mathbf{x}}\right]_{i,i} |\mathbf{J}_{\boldsymbol{\tau}\to\mathbf{x}}| \int_{\Pi} v\,(\partial_{\xi_i} u_i)\, d\Pi \\
&= \mathrm{c}_i \int_{\Pi} v\,(\partial_{\xi_i} u_i)\, d\Pi
\end{aligned}
\tag{2.51}
$$

The integration of the **Cartesian elements located on the boundary** is more complex because the integration domain only covers part of the total element domain. The comments made for the transformation $\boldsymbol{\xi} \to \mathbf{x}$ remain valid since the relationship between these two domains does not change. However, the relationship between both and the integration domain is modified. There are different strategies for calculating these integrals. The approach we will use in this thesis consists of dividing the integration domains into subdomains with simpler geometries that can be obtained through the transformation of a finite set of reference domains. Regarding this mapping, the two cases illustrated in Fig. 2.8 are the approaches that have been studied in the cgFEM framework:

1. Polynomial approach [135]: A polynomial is fitted to a set of sampled points (red squares in Figure 2.8) over each portion of the boundary delimited by intersection with the element's boundaries or over changes in the definition of the geometry (green squares in Figure 2.8). The integration subdomains are then computed considering this polynomial approximation of the exact geometry.

2. NEFEM approach [63]: The parametric definition of the boundary provided by the CAD is taken into account. This approach allows to remove any geometric error induced by a polynomial approximation of the boundary.

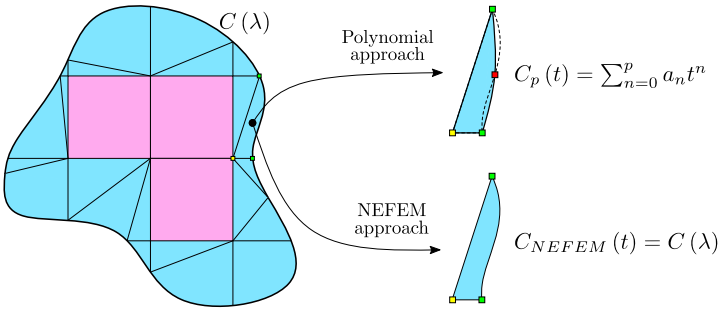


Figure 2.8: *Example of different resulting integration subdomain depending on the boundary approximation: polynomial (top), NEFEM (bottom)*

In any case, Equation (2.51) for an intersected element is reformulated as:

$$\begin{aligned} a_{bC}(\mathbf{u}, v) &= \sum_{j=1}^{nc} \int_{\Omega_{k,j}} v \nabla_{\mathbf{x}} \cdot \mathbf{u} d\Omega \\ &= \left[\mathbf{J}^{-1}_{\boldsymbol{\xi} \to \mathbf{x}}\right]_{i,i} \sum_{j=1}^{nc} \int_{\Pi} v \left(\partial_{\xi_i} u_i^{(j)}\right) \left|\mathbf{J}_{\boldsymbol{\tau} \xrightarrow{j} \mathbf{x}}\right| d\Pi \\ &= c_i \sum_{j=1}^{nc} \int_{\Pi} v \left(\partial_{\xi_i} u_i^{(j)}\right) \left|\mathbf{J}_{\boldsymbol{\tau} \xrightarrow{j} \mathbf{x}}\right| d\Pi \end{aligned}$$

where nc represents the number of subdomains into which Ω_k is decomposed, $\boldsymbol{\tau} \xrightarrow{j} \mathbf{x}$ is the transformation from the integration space to the physical space for subdomain j, and $u_i^{(j)}$ is the i-th component of $\mathbf{u}$ expressed in the $\boldsymbol{\tau}$ system. If we consider the polynomial approach with a polynomial transformation of degree r from the integration space to the physical space, we encounter an integrand of degree $(2p-1)(dr)+d(r-1)$, which can be exactly integrated with the available quadratures. The increase in degree is due both to the effect of the Jacobian determinant and to the composition of u_i with the mapping $\boldsymbol{\tau} \xrightarrow{j} \boldsymbol{\xi}$, which is affine. Alternatively, if NURBS-enhanced integrations [136] are used, the transformations will be rational and it will not be possible to evaluate the integral exactly. In these cases, as in the isoparametric approximation, quadratures of degree $(2p-1)+d(r-1)+n$ will be used, where n must be high enough so that the integration errors do not affect the convergence of the method (see [6] for further details).

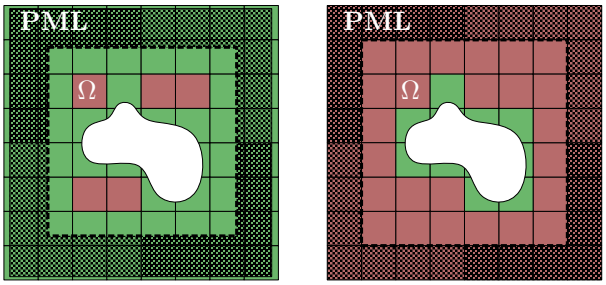
Proper integration of boundary elements is a key aspect in the development of this thesis. In Sections 3.3 and 3.4 and Section 4.2 we provide a thorough explanation of the strategies we have designed to ensure that the integration of intersected elements is accurately computed in the 2D and 3D frameworks, respectively.

2.3.2. Implementation of Perfectly Matched Layers in cgFEM

It is worth mentioning that the PML strategies fit perfectly within the cgFEM framework. The Cartesian structure of the mesh offers an ideal setting to define the subdomains of the PML. Furthermore, by matching the boundary of these subdomains with the element edges, the increase in computational cost associated with the arising of a greater number of intersected elements is avoided.

In Fig. 2.9, two possible PML configurations for the same problem are illustrated. The domain of interest, Ω, extends from the central body to the interface with the PML, which is represented by a dashed line, while the shaded region corresponds to the PML subdomains. Mesh elements are color-coded based on their classification: red for internal elements and green for intersected ones. Comparing Fig. 2.9a

and Fig. 2.9b clearly highlights the advantage of aligning the outer boundary of Ω with mesh interfaces. As discussed in Section 2.3.1, evaluating intersected elements is significantly more complex and computationally demanding than evaluating internal ones. With a well-designed PML, the number of intersected elements can be drastically reduced, which in turn lowers the overall computational cost. In this specific example, the number of intersected elements is reduced from 59 to only 11.



(a) *Straightforward setting* ***(b)*** *Optimized setting*

Figure 2.9: *PML in the cgFEM context*

Generating PMLs that conform to the mesh arises naturally if the process is approached in reverse. One can start from the geometry of the object responsible for generating or scattering the electromagnetic field and define propagation distances desired in each medium. The position and size of the fictitious domain from which the mesh is generated and the size of the elements can then be adjusted so that domains consider in the simulation approximately meets those propagation distances. This approach also simplifies the task for the analyst, who no longer needs to explicitly define the outer geometry of the domain of interest and the PML, but only needs to decide on their dimensions.

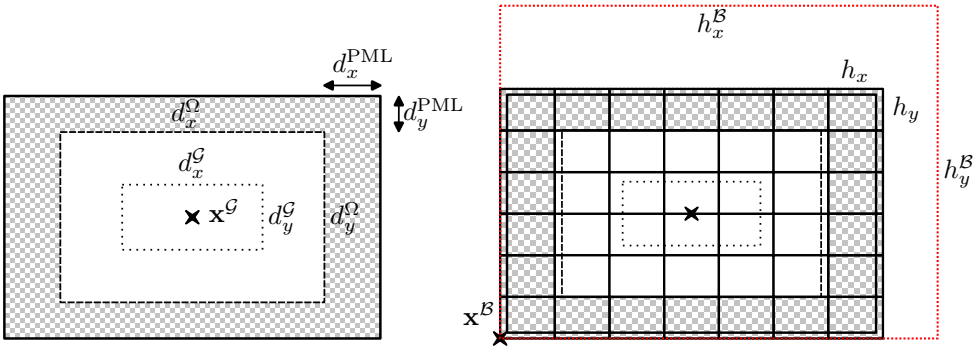


Figure 2.10: *Mesh-based definition of the PML in cgDG*

Let us consider a setup like the one shown in Fig. 2.10, where the aim is to simulate over a domain Ω with dimensions $\mathbf{d}^{\Omega} = d_x^{\Omega}, d_y^{\Omega}$, with the geometry $\mathcal{G}$ centered and a PML that symmetrically extends Ω a distance $\mathbf{d}^{\mathrm{PML}} = d_x^{\mathrm{PML}}, d_y^{\mathrm{PML}}$. Once an element size $\mathbf{h} = h_x, h_y$ is defined, the number of elements in each domain along each direction can be computed:

$$\mathbf{n}^{\Omega} = \left\lceil \left(\mathbf{d}^{\mathcal{G}} + 2\mathbf{d}^{\Omega} \right) \oslash \mathbf{h} \right\rceil \tag{2.52}$$

$$\mathbf{n}^{\mathrm{PML}} = \left\lfloor \mathbf{d}^{\mathrm{PML}} \oslash \mathbf{h} \right\rceil \tag{2.53}$$

The maximum number of elements in each direction is limited by the mesh level, so the minimum mesh level containing a sufficient number of elements must be identified:

$$l = \log_2 \left(\max \left(\mathbf{n}^{\Omega} + 2\,\mathbf{n}^{\mathrm{PML}} \right) \right) \tag{2.54}$$

The dimension of the fictitious domain used to generate the Cartesian mesh is defined by $\mathbf{h}$ and l, while to position it, we start from the center of the geometry $\mathbf{x}^{\mathcal{G}}$ and apply an offset determined by $\mathbf{n}^{\Omega}$, $\mathbf{n}^{\mathrm{PML}}$, and $\mathbf{h}$:

$$\mathbf{h}^{\mathcal{B}} = 2^l\,\mathbf{h} \quad , \quad \mathbf{x}^{\mathcal{B}} = \mathbf{x}^{\mathcal{G}} - \mathbf{h} \odot \left(\frac{\mathbf{n}^{\Omega}}{2} + \mathbf{n}^{\mathrm{PML}} \right) \tag{2.55}$$

Fig. 2.10 reveals that the dimensions used in the simulation will not match the target model exactly, but rather provide the best approximation that conforms to the mesh and uses a uniform element size. The discrepancy between models usually has no adverse consequences since the analyst typically chooses the distances to ensure that the model is large enough to capture the wave's development in each domain. With the approach we have presented, the domains in the simulation will be larger (yet more computationally efficient).

Let us conclude this section with one comment on the consequences of this procedure in later stages of the simulation. When constructing the computational mesh, a pre-selection of elements is made based on the domains obtained from $\mathbf{n}^{\Omega}$ and $\mathbf{n}^{\mathrm{PML}}$, which are then classified according to their location with respect to $\mathcal{G}$. All elements farther from $\mathbf{x}^{\mathcal{B}}$ are classified as external. As far as PML is concern, the equations from Section 2.1.2.1 are applied to the elements within the $\mathbf{n}^{\mathrm{PML}}$ outermost layers of the mesh and the coordinates used in (2.17) and (2.18) can be easily computed from $\mathbf{x}^{\mathcal{B}}$, $\mathbf{h}$, $\mathbf{n}^{\mathrm{PML}}$, and $\mathbf{n}^{\Omega}$.

Chapter 3

The cgDG in the 2D case

In this chapter, we detail the implementation of the cgDG method in the 2D context. Starting from the formulation presented in Section 2.2, we explain how we applied the unfitted discretization and we leveraged the special features of cgFEM to maximize the performance of the method. There are two main difficulties to address: design of appropriate integration quadratures and stabilization of time-integration scheme. First, the use of high order elements comes with the demand of highly accurate integration schemes to ensure their convergence. Second, the support of the piecewise functions used to approximate the solution is independent of the element size for elements on the boundary of the computational domain. When the support of any of these functions is very small, the maximum stable time step of the discretization suffers a drastic reduction, hugely enlarging the calculation time. Regarding their importance, independent sections are devoted to describe the strategies that have been developed to overcome these initial limitations of the method.

3.1. Discontinuous Galerkin formulation in the cgFEM's framework

3.1.1. Spatial discretization

Let us recall the DG formulation of the electromagnetic problem presented in Section 2.2:

$$\int_{\Omega_e} \mathbf{W} \cdot \frac{\partial \mathbf{U}_e}{\partial t} d\Omega + \int_{\Omega_e} \mathbf{W} \cdot \frac{\partial \mathbf{F}_k(\mathbf{U}_e)}{\partial x_k} d\Omega + \int_{\partial\Omega_e} \mathbf{W} \cdot [\![\mathbf{F}_{\mathbf{n}}^{-}(\mathbf{U}_e)]\!] d\Gamma = \int_{\Omega_e} \mathbf{W} \cdot \mathbf{S}(\mathbf{U}_e)\, d\Omega \quad (3.1)$$

where $\mathbf{W}$ is the vector containing all test functions, $\mathbf{U}_e$ is the vector of conservative variables restricted to the domain Ω_e, $\mathbf{F}_k(\mathbf{U}_e)$ is the hyperbolic flux in the k-th direction, $[\![\mathbf{F}_{\mathbf{n}}^{-}(\mathbf{U}_e)]\!]$ is the jump in the incomming component of the flux through the contour $\partial\Omega_e$ with outward normal vector $\mathbf{n}$ and $\mathbf{S}(\mathbf{U}_e)$ is the vector gathering all source terms.

In the framework of the FDM, the calculated solution only makes physical sense when evaluated on the problem's domain. Therefore, the approximate field $\mathbf{U}^h(\mathbf{x}, t)$ is mathematically represented by piece-wise defined function:

$$\mathbf{U}^h(\mathbf{x}, t) = \begin{cases} N_i(\mathbf{x})\, \mathbf{U}_i^h(t) & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \in \Omega_e \\ \mathbf{0} & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \notin \Omega_e \end{cases} \quad (3.2)$$

where $\mathbf{U}_i^h(t)$ is the vector that contains the values of the conservative variables evaluated at the node i of the element e at the instant t and $N_i(\mathbf{x})$ represents the shape functions of degree p associated with the node i evaluated at the physical coordinate $\mathbf{x}$. This approximate solution is contained in the polynomial space formed by the tensor product of one-dimensional polynomials up to degree p with support in Ω_e, $\mathcal{Q}^p(\Omega_e)$. A Fekete-nodal distribution [137, 138] is considered to maximize the accuracy of the resulting approximation. Note that this function space is different from the one stated in section 2.2.

Regarding the construction of the test functions, a similar formulation is used:

$$W^h(\mathbf{x}) = \begin{cases} N_i(\mathbf{x})\, W_i^h & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \in \Omega_e \\ 0 & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \notin \Omega_e \end{cases} \quad (3.3)$$

In the following, we consider that flux operators $\mathbf{F}_k$ and $\mathbf{F}_{\mathbf{n}}^{-}$ and the source operator $\mathbf{S}$ are linear, as usual for the electromagnetic problem. Likewise, the analytical

expression of the source term will be substituted by a polynomial approximation obtained by interpolating the values of the source evaluated at the nodes. Plugging (3.2) and (3.3) into (2.36) and taking into account the aforementioned assumptions we obtain:

$$W_i^h M_{ij} \frac{d\mathbf{U}_j^h}{dt} + W_i^h C_{ijk} \mathbf{F}_k \left(\mathbf{U}_j^h\right) + W_i^h \mathbf{\Phi}_i = W_i^h M_{ij} \mathbf{S} \left(\mathbf{U}_j^h\right) \tag{3.4}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}_k$ is the convection matrix in the k-th direction and $\mathbf{\Phi}$ represents the electromagnetic flux through the boundary of the domain within an element. More in detail, the definition of these terms regarding the shape functions is:

$$M_{ij} = \int_{\Omega_e} N_i N_j d\Omega \tag{3.5}$$

$$C_{ijk} = \int_{\Omega_e} \partial_k N_i N_j d\Omega \tag{3.6}$$

$$\mathbf{\Phi}_i = \int_{\partial\Omega_e} N_i \mathbf{F}_{\mathbf{n}}^- \left(\mathbf{U}_{out}\right) d\Gamma - \int_{\partial\Omega_e} N_i N_j \mathbf{F}_{\mathbf{n}}^- \left(\mathbf{U}_j^h\right) d\Gamma \tag{3.7}$$

Remark. Notice that equation (3.4) is a system of ordinary differential equations such as that anticipated in Section 2.2.2:

$$\mathbf{M} \frac{d\mathbf{U}^h}{dt} + \mathbf{R}(\mathbf{U}^h) = \mathbf{0} \tag{3.8}$$

with the vector of residuals $\mathbf{R}(\mathbf{U}^h)$ being:

$$R_i(\mathbf{U}^h) = C_{ijk} \mathbf{F}_k \left(\mathbf{U}_j^h\right) + \mathbf{\Phi}_i - M_{ij} \mathbf{S} \left(\mathbf{U}_j^h\right) \tag{3.9}$$

3.1.2. Evaluation of the operators in the weak form

Next step in the analysis implies the numerical integration of the operator in the weak form. We apply coordinate transformation (remember that $\mathbf{a} \rightarrow \mathbf{b}$ notes the mapping from space $\mathbf{a}$ into space $\mathbf{b}$) to map the domain of the integral from the physical space $\mathbf{x}$ into the space $\boldsymbol{\tau}$ in which the domain correspond to a reference one for which an integration quadrature is available. Fig. 3.1 is a reminder of the coordinates systems we will be working with and their transformations.

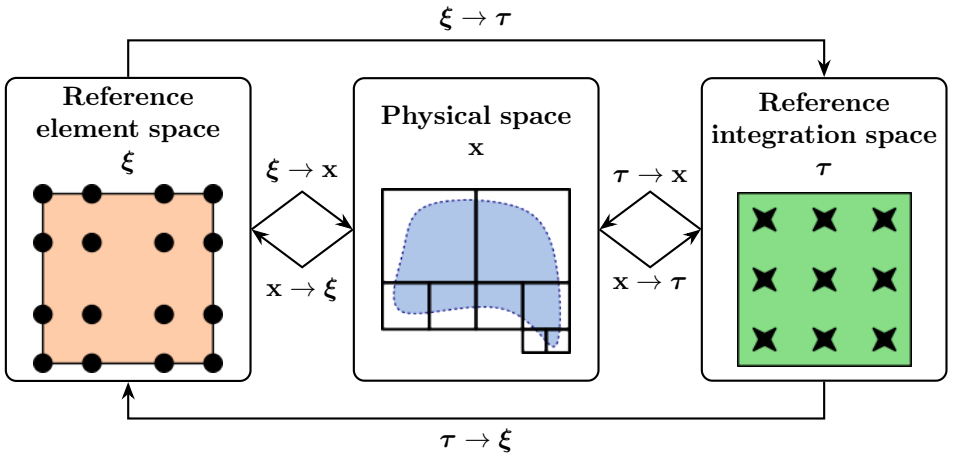


Figure 3.1: *Coordinate systems involved in the evaluation of the integrals arising in the cgFEM discretization.*

Notice that, N_i in equations (3.2) and (3.3) depends on the physical coordinate $\mathbf{x}$. However, it is inaccurate since these functions are actually defined in reference element space $\boldsymbol{\xi}$, namely $N_i = N_i(\boldsymbol{\xi})$. Strictly speaking, $N_i(\mathbf{x})$ computes in two stages: first mapping from physical to reference element coordinates (see $\mathbf{x} \to \boldsymbol{\xi}$ in Fig. 3.1), then evaluating $N_i(\boldsymbol{\xi})$. This license was adopted for the sake of simplicity but is reversed in the following developments.

3.1.2.1. Operators over Ω_e

Let us begin by studying the integrals over the interior of the domain. We focus on the case of the mass matrix. Applying the coordinate transformations defined above:

$$M_{ij} = \sum_{n=1}^{n_{is}} \int_{\Pi_n} N_i\left(\boldsymbol{\tau} \overset{n}{\to} \boldsymbol{\xi}\right) N_j\left(\boldsymbol{\tau} \overset{n}{\to} \boldsymbol{\xi}\right) \left|\mathbf{J}_{\boldsymbol{\tau} \overset{n}{\to} \mathbf{x}}\right| d\Pi \tag{3.10}$$

where n_{is} is the number of integration subdomains within the element, $N_i(\boldsymbol{\tau} \overset{n}{\to} \boldsymbol{\xi})$ note the composition of the shape function of the node i with the mapping $\boldsymbol{\tau} \overset{n}{\to} \boldsymbol{\xi}$ associated with the integration subdomain n and $\mathbf{J}_{\boldsymbol{\tau} \overset{n}{\to} \mathbf{x}}$ represents the Jacobian of the transformation $\boldsymbol{\tau} \overset{n}{\to} \mathbf{x}$.

Following Section 2.3, *internal elements* are treated as standard finite elements. Those elements contain a single integration subdomain that matches the element domain so the superscripts n are removed for simplicity. Both the reference domain for integration and the reference element are invariant and similar in shape. Therefore, it is clear that the mapping $\boldsymbol{\tau} \to \xi$ is affine, it is actually a scaling, and element-independent. Together with equation (2.46), it is easy to prove that the Jacobian

of the transformation $\boldsymbol{\tau} \rightarrow \mathbf{x}$ is not only constant but defined by the mesh level the element belongs to. Let us define as $\boldsymbol{\xi} \xrightarrow{l_0} \mathbf{x}$ and $\boldsymbol{\tau} \xrightarrow{l_0} \mathbf{x}$ the mappings associated with the 0-level element, their corresponding Jacobians compute:

$$\mathbf{J}_{\boldsymbol{\tau}\rightarrow\mathbf{x}} = 2^{-l}\mathbf{J}_{\boldsymbol{\tau}\xrightarrow{l_0}\mathbf{x}} \quad , \quad \mathbf{J}_{\boldsymbol{\xi}\rightarrow\mathbf{x}} = 2^{-l}\mathbf{J}_{\boldsymbol{\xi}\xrightarrow{l_0}\mathbf{x}} \tag{3.11}$$

Plugging these relations in (3.10) and considering d spatial dimension we obtain:

$$M_{ij} = 2^{-d\,l}\left|\mathbf{J}_{\boldsymbol{\tau}\xrightarrow{l_0}\mathbf{x}}\right| \int_{\Pi} N_i\left(\boldsymbol{\tau}\rightarrow\boldsymbol{\xi}\right) N_j\left(\boldsymbol{\tau}\rightarrow\boldsymbol{\xi}\right) d\Pi \tag{3.12}$$

If we study the integrand of the expression, we observe that it only depends on the interpolation functions and the mapping from the integration space to the reference element space. On the one hand, the interpolation functions are completely defined from a nodal distribution which is specific of the degree of approximation p used. On the other hand, the mapping $\boldsymbol{\tau} \rightarrow \boldsymbol{\xi}$ is an affine transformation common to all the internal elements. Gathering this results, we can rewrite the expression as:

$$M_{ij} = 2^{-d\,l} M_{ij}^{0,p} \tag{3.13}$$

where

$$M_{ij}^{0,p} = \left|\mathbf{J}_{\boldsymbol{\tau}\xrightarrow{l_0}\mathbf{x}}\right| \int_{\Pi} N_i\left(\boldsymbol{\tau}\rightarrow\boldsymbol{\xi}\right) N_j\left(\boldsymbol{\tau}\rightarrow\boldsymbol{\xi}\right) d\Pi \tag{3.14}$$

represents the mass matrix of the 0-level element of degree p.

Proceeding in a similar way with the convection matrix we arrive at the expression:

$$C_{ijk} = 2^{1-d\,l} C_{ijk}^{0,p} \tag{3.15}$$

where

$$C_{ijk}^{0,p} = \left|\mathbf{J}_{\boldsymbol{\tau}\xrightarrow{l_0}\mathbf{x}}\right| \left(\mathbf{J}_{\boldsymbol{\xi}\xrightarrow{l_0}\mathbf{x}}\right)^{-1}_{kl} \int_{\Pi} \partial_l N_i^p\left(\boldsymbol{\tau}\rightarrow\boldsymbol{\xi}\right) N_j^p\left(\boldsymbol{\tau}\rightarrow\boldsymbol{\xi}\right) d\Pi \tag{3.16}$$

represents the convection matrix of the 0-level element of degree p and ∂_l symbolizes the derivative with respect to local coordinates.

With this formulation we compute the mass and convection matrices of any of the n_{ie} internal elements of the mesh by scaling the reference matrices. This allows us to save a large amount of memory space, since we move from having to save 2 n_{ie} matrices to 2 n_{ip} matrices, where n_{ip} stands from the number of different degrees of approximations that have been used for the spatial discretization.

In the case of *boundary elements*, the domain Ω_e does not match the element domain K_e and the integral is computed as a sum of integrals over simpler subdomains which are obtained by splitting Ω_e. Subsequently, one of the techniques mentioned in section 2.3 is applied to obtain the transformation functions $\boldsymbol{\tau} \rightarrow \mathbf{x}$ and $\boldsymbol{\tau} \rightarrow \boldsymbol{\xi}$. It is not possible to do any simplification since the number of terms of the summation

in (3.10) will depend on the geometry and approximation of Ω_e and the integrand of each of these terms will depend on the transformations $\boldsymbol{\tau} \rightarrow \mathbf{x}$ and $\boldsymbol{\tau} \rightarrow \boldsymbol{\xi}$, whose definitions comes from the geometry and position in the element of each integration subdomain. Therefore, the domain integrals of each of the intersected elements are treated independently according to the general formulation. It is worth to remark that, even though the computational burden associated to these integrals is higher than that associated to internal elements, the number of intersected elements is small with respect to the total amount of elements in the mesh.

3.1.2.2. Operators over $\partial\Omega_e$

Regarding the electromagnetic flux through the boundary of the domain within an element $\boldsymbol{\Phi}_i$, we differentiate two groups depending on the characteristics of the integrals:

1. *Integrals of interfaces between elements*: the external and internal fields are defined by the solution of the finite elements analysis in the neighbour element and in the current element, respectively. Likewise, the values are computed by interpolating the solutions of those nodes located on the boundary under study, since the influence of the rest of the nodes is null. Due to the Cartesian structure of the mesh and its independence from geometry, all element faces possess a constant normal, which is aligned with one of the Cartesian directions. As a consequence, the incoming flow operator is identical for all points on the face and, since it is linear with respect to the conservative field, it can be taken from the integrand in (3.7). Furthermore, this operator can only respond to $2d$ expressions, which correspond to the Cartesian directions. For the 2D case: $\left\{\mathbf{F}^-_{\bar{\mathbf{n}}} \text{ with } \bar{\mathbf{n}} \in \{(1,0),(0,1),(-1,0),(0,-1)\}\right\}$. With this in mind, we can express (3.7) as:

$$\boldsymbol{\Phi}_i = m_{ik}\mathbf{F}^-_{\bar{\mathbf{n}}}\left(\mathbf{U}^h_k\right) - m_{ij}\mathbf{F}^-_{\bar{\mathbf{n}}}\left(\mathbf{U}^h_j\right) \tag{3.17}$$

   where the sub-indices j and k identify the rows in $\mathbf{U}^h$ that correspond to the nodes of the element and its neighbour, respectively, and:

$$m_{ij} = \int_{\partial\Omega_e} N_i N_j d\Gamma \tag{3.18}$$

   is a mass matrix of dimension $d-1$.

   In turn, this set of integrals can be subdivided into:

   1.a *Integrals over complete edges*: m_{ij} corresponds to the mass matrix of a cartesian element of one lower dimension than the dimension of the problem. Following a similar development to that presented for the volume

integrals, for a given element of degree p, we can express m_{ij} as a rescaling of a mass matrix of dimension $d-1$ calculated for the 0-level element considering weighting functions of degree p and test functions of the degree n, where n is the degree of the polynomial field being evaluated (internal or external):

$$m_{ij} = 2^{(1-d)\,l} m_{ij}^{0,pn} \tag{3.19}$$

$$m_{ij}^{0,pn} = \left|\mathbf{J}_{\boldsymbol{\tau} \xrightarrow{l_0} \mathbf{x}}\right| \int_{\partial\Pi} N_i^p\left(\boldsymbol{\tau} \to \boldsymbol{\xi}\right) N_j^n\left(\boldsymbol{\tau} \to \boldsymbol{\xi}\right) d\pi \tag{3.20}$$

This way of obtaining boundary integrals on complete edges reduces the amount of information that needs to be stored in memory. However, it must be considered that, for h-refinement non-uniform meshes, an element may be connected through its boundary with more than one neighbouring elements or that the integral on the face of the neighbouring element does not cover its entire length but only part of it. This situation can be seen in Figure 3.2, where two different examples of interfaces between elements and their coding are represented. The encoding we use is composed of four numbers grouped in two pairs of two numbers. The first number of each pair is the ratio between the element side dimension and the size of the interface being integrated, while the second number represents the position of the interface on the element edge. It is possible to identify all the typologies of integrals over non-intersected element faces and calculate their respective reference matrices for the 0-level element. Something similar happens in the case of using p-refining strategies, where a mass matrix $m_{ij}^{0,pn}$ will be obtained for each combination of degrees in the mesh.

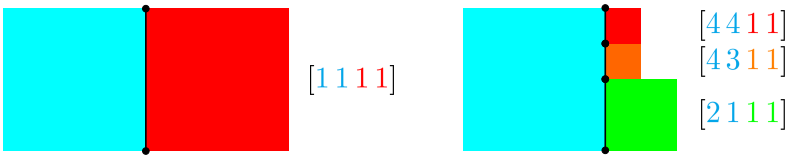


Figure 3.2: *Topology of two different internal faces and code for processing.*

1.b *Integrals over intersected edges*: When the element edge intersects the boundary of the physical domain, the integrand is null over the outer section and the integral restricts to $\partial K_e \cap \Omega$. This domain is different for each of the flux integrals over intersected faces and, therefore, their mass matrices are calculated individually according to:

$$m_{ij}^{e} = \int_{\partial K_e \cap \Omega} N_i N_j d\Gamma \tag{3.21}$$

2. *Integrals over the boundary*: The calculation of the incoming numerical flux will be subjected to the boundary conditions imposed on the contour being integrated, while the conservative variable distribution due to the numerical solution in the element will be a function of the calculated values for all its nodes, not only those on the element's boundary. This last characteristic is a feature of FDM methods and is a consequence of the physical domain boundary being internal to the mesh. Integrals over boundaries will be calculated following the general expression (3.7).

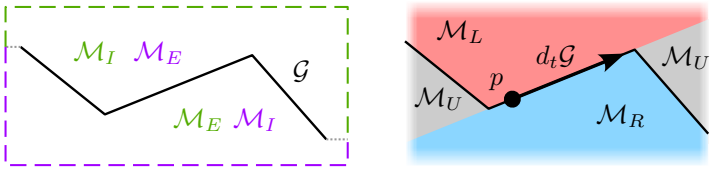
As a previous step to integral definition, we need to identify what elements meet at each of the interfaces in a process known as neighbour search. In the context of cgFEM, we can use the virtual numbering presented in Section 2.3 to find out the connectivity between elements. Using the virtual coordinates this process becomes trivial. Furthermore, we can also identify the set of different combination of degrees p, level jumps Δl and relative positions of neighbour elements in the mesh, reducing the amount of computations.

3.1.3. Superposition of elements with non-overlapped support

Intersected elements may contain disjointed regions where the solution is physically independent, see Fig. 3.4d. If we use the standard discretization of cgFEM, the polynomial approximation would be shared by all the regions contained within the element. Trying to represent with a single smooth function a set of independent fields leads to a significant increase in the local error associated to these elements and the pollution of the global solution. This problem becomes more relevant as the size of the elements increases and the details of the geometry become smaller than the characteristic length of the mesh. This situation is quite usual in high-order methods.

First, it is important to point out that we work with domains defined by their boundaries. In such cases, the geometries merely indicate the presence of an interface between materials. The properties of the media on either side of the interface are not specified until a criterion is introduced to distinguish them. A common approach that leverages the fact that the geometry is closed, is to classify the media as either internal or external with respect to the geometry. However, this approach presents limitations. When analyzing a portion of the interface in isolation, it becomes impossible to determine which medium lies on each side, as the geometry is locally open. In Fig. 3.3a, the segment of the geometry $\mathcal{G}$ represented in solid-line yields an ambiguous classification of the interior $\mathcal{M}_I$ and exterior $\mathcal{M}_E$ media, as it is unclear whether the interface closes above (green classification) or below (violet classification). To resolve this ambiguity, this thesis adopts an alternative strategy: assigning meaning to the direction in which the geometry is traversed. According to this convention, the media are identified as lying to the left or right of the traversal direction along the interface.

This allows the properties of the medium on each side to be defined locally. Consequently, the material properties at any point in space can be inferred from its relative position to a point (with a well-defined tangent) on the geometry. This strategy enables the isolated analysis of interface segments and facilitates parallel processing of intersected elements. As illustrated in Fig. 3.3b, the tangent vector $d_t\mathcal{G}$ at a point p on the geometry is sufficient to assign the left $\mathcal{M}_L$ and right $\mathcal{M}_R$ material properties to the red and blue regions, respectively. The material in the gray regions remains undetermined $\mathcal{M}_U$ as those points are not visible from p.

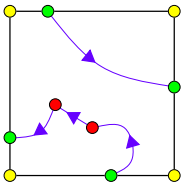


(a) *Global approach* ***(b)*** *Local approach*

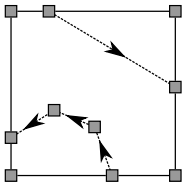
Figure 3.3: *Comparison of two strategies for domain labeling when applied to a section of the geometry.*

In the following, we describe the algorithm that we employ to detect isolated regions. The reader can find an example of application in Fig. 3.4. If we focus on a single element on the boundary, the output of the intersection routine is the set of curves internal to the element and the segments of the curves limited by intersection points and endings of the curves such that any point on the segment is internal to the element. Next, we split the contour of the element into a set of segments taking into account the intersection points and the corners (see Fig. 3.4a). Then we obtain the connectivity of these segments and curves (hereafter the edges) through the intersection points and the corners (hereafter the vertices). The edges on the geometry are directed paths according to the definition of the geometry (see Fig. 3.4b). To find an independent region in the element, we identify the tail of a directed edge, move to the successor and go through the graph always taking the first edge form left to right at the vertices until we arrive to the starting vertex. This procedure is executed taking different starting points until all directed edges have been traversed once. The loops computed this way are classified as Type I. Then, it repeats changing the turning criterion to always take the first path form right to left. The new loops are classified as Type II (see Fig. 3.4c).

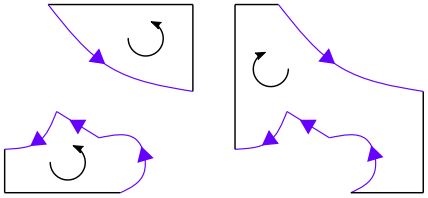
As aforementioned, the region at both sides of the geometry is completely defined and the turning criterion at the vertices characterizes the physical properties of the domain inside each loop. By the end of this algorithm, we have obtained all independent regions within the element and their physical properties. For the sake of simplicity, let us assume that there is just one propagation medium. Once those regions outside the physical domain are removed, the remaining regions are necessarily isolated regions (see Fig. 3.4d).



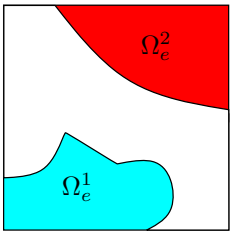
(a) *Geometric information: portions of the boundary inside the element (purple segments), edges of the element (black segments), ending points of curves (red circles), intersection points (green circles), corners of the element (yellow circles).*



(b) *Directed graph: directed edges (dashed lines with arrows showing direction), undirected edges (solid lines) and nodes (grey squares).*



(c) *Computed loops of Type I (left) and Type II (right)*



(d) *Isolated regions Ω_e^1 (blue) and Ω_e^2 (red): in this example we consider that the computational domain is on the left-hand side of the geometry and, therefore, Type II loops are removed.*

Figure 3.4: *Algorithm employed to detect isolated regions within an element*

In this work, the spatial discretization is modified in those elements that contain two or more regions of the physical domain disjoined to each other (from now on referred to as isolated regions). Let us consider the element e located on the boundary of the physical domain Ω in such a way that their intersection leads to a set of $\mathrm{n_{ir}}$ isolated regions in it, namely $\Omega_e = \cup_{r=1}^{\mathrm{n_{ir}}} \Omega_e^r$, (see Fig. 3.5b). The set of nodes $\mathfrak{N}_e$ associated with the element e is split into $\mathrm{n_{ir}}$ subsets $\{\mathfrak{N}_e^1, \mathfrak{N}_e^2, ..., \mathfrak{N}_e^{\mathrm{n_{ir}}}\}$ and the shape functions N_i used to interpolate the nodal values are redefined considering each subset $\mathfrak{N}_e^r$ separately. The approximate field $\mathbf{U}^h(\mathbf{x}, t)$ and the test functions $W^h(\mathbf{x})$ in a boundary element with multiple isolated regions using standard summation notation formulate as:

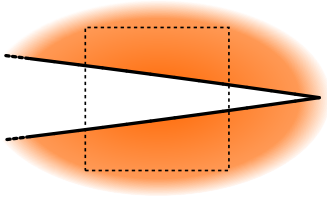
$$\mathbf{U}^h(\mathbf{x}, t) = \begin{cases} \sum_{i \in \mathfrak{N}_e^r} N_i(\mathbf{x})\, \mathbf{U}_i^h(t) & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \in \Omega_e^r \\ \mathbf{0} & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \notin \Omega_e^r \end{cases} \tag{3.22}$$

and

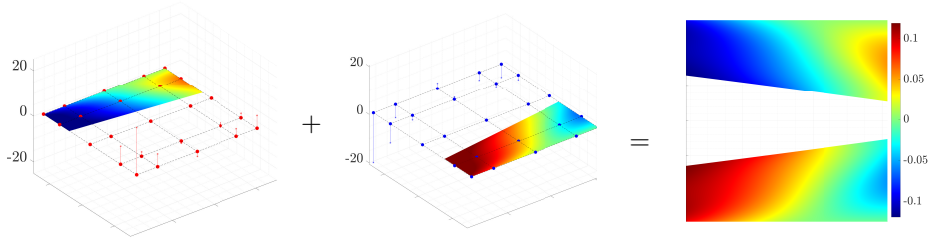
$$W^h(\mathbf{x}) = \begin{cases} \sum_{i \in \mathfrak{N}_e^r} N_i(\mathbf{x})\, W^h & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \in \Omega_e^r \\ \mathbf{0} & \text{if } \mathbf{x} \in K_e \text{ and } \mathbf{x} \notin \Omega_e^r \end{cases}, \tag{3.23}$$

respectively. The final discretization is completely independent form one region to another so this strategy allows the use of different polynomial degrees for each of them. Indeed, we employ an independent Fekete nodal distribution mapped on K_e for each subset $\mathfrak{N}_e^r$. An example of how the dissociation strategy works is shown in Fig. 3.5.

In [139] a similar dissociation approach was introduced to handle discontinuities internal to the elements in the context of 2D elasticity problems. That same approach was later on applied to other problems, see [140] and the references therein. Our implementation surpasses existing approaches in the literature in the sense that the number of isolated regions within an element is not restricted, the regions are bounded by the exact representation of the geometry provided by the CAD software, the FE approximation is defined at region level and high order polynomial basis are considered. Moreover, the DG solver adopted in this thesis let us to weakly impose the continuity of the solution across those interfaces that lie on the computational domain without the need of adding any additional term to the original formulation.



(a) *Geometry-mesh superposition*



(b) *Nodal values (z-coordinate) and distributions (colored regions).*

(c) *Resulting field*

Figure 3.5: *Example of two independent fields within a single element. The approximate solution on the top region Ω_e^1 corresponds to the interpolation of the nodal values at nodes drawn in red $\mathfrak{N}_e^1$ while the nodal values at blue nodes $\mathfrak{N}_e^2$ define the approximate solution on the bottom region Ω_e^2. The sets of red and blue nodes are linked to the same element, namely $\mathfrak{N}_e = \{\mathfrak{N}_e^1, \mathfrak{N}_e^2\}$, but the support of the polynomial interpolations differ from one to the other. Thus, the resulting approximate solution on each isolated region is independent.*

3.2. Non-Uniform Rational B-Splines

In Section 3.1.3, the representation of computational domains in cgDG was briefly introduced. The choice of a contour-based modeling strategy is not arbitrary, but responds to practical considerations. Most geometrically complex domains are created using Computer-Aided Design (CAD) software, which internally relies on boundary representation (B-Rep) to define shapes. The advantages of incorporating B-Rep into the computational pipeline are examined in detail in Section 4.1, where its structure, features and application in the context of three-dimensional geometries are explored.

This section focuses on the mathematical models used to define the contours themselves, independently of their integration into B-Rep structures. A simulation framework consistent with the geometric model —such as the one pursued in this thesis— must be capable of operating directly with the native representation of the geometry. This philosophy is central to cgDG. Evidence of this is found in the contributions of this thesis in the area of numerical integration. See techniques presented in Sections 3.3, 3.4 and 4.2 aiming to avoid geometric simplifications of the integration domains.

The formulation adopted for defining geometric contours is based on Non-Uniform Rational B-Splines (NURBS, see [141, 142]). The NURBS are interpolation functions that have become a CAD industry standard thanks to their favorable features:

- NURBS provide a unified mathematical basis for representing both analytic and freeform shapes without the need for approximations.
- The generation and modification of NURBS is intuitive, facilitating design and manipulation tasks.
- NURBS algorithms are fast and numerically stable.
- NURBS are invariant under common geometric transformations.

The following sections provide a brief introduction to NURBS curves, emphasizing the aspects most relevant for their integration into the cgDG methodology. Section 3.2.1 presents the definition and formulation of NURBS, highlighting key properties of the basis functions and resulting curves. This mathematical framework plays a direct role in the construction of integration quadratures, as elaborated in Sections 3.3 and 4.2, while the algorithms in Sections 3.4 and 4.1 exploit the underlying information provided by the geometric model. This underlying information can be further enriched through the refinement strategy introduced in Section 3.2.2, which allows modification of the NURBS parameterization without altering the geometric shape. This capability proves useful and the refinement strategy is employed across several of the proposed algorithms. Lastly, Section 3.2.3 introduces the custom library developed as part of this thesis, which implements all NURBS-related functionalities required to support the techniques that have been developed.

3.2.1. Definition and properties

NURBS curves are a generalization of B-Spline curves. A B-Spline curve is a parametric curve defined over a non-decreasing sequence $U = \{u_0, u_1, \ldots, u_m\}$ of real numbers, i.e. $u_0 \leq u_1 \leq \ldots \leq u_m$, which is the piecewise polynomial interpolation of a sequence of points $\mathcal{P} = \{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n\}$. Following common terminology, u_i are the knots, U is the knot vector, $\mathbf{P}_i$ are the control points, and $\mathcal{P}$ is the control polygon. Note that the u_i do not need to be distinct; we define the multiplicity k of a knot u_i as the number of times its value repeats in U, i.e. $k \in N^+$ such that $u_i = u_j = u_{j+1} = \ldots = u_{j+k-1}$. On the other hand, the set of knot values without repetitions is identified as break points.

The degree p of the polynomial interpolation basis $\{N_{i,p}(u)\}$ is given by the cardinality of U and $\mathcal{P}$ according to the expression $m = p + n + 1$. Its definition is constructed iteratively according to the expression:

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & \text{if} \quad u_i \leq u < u_{i+1} \\ 0 & \quad \text{otherwise} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned} \tag{3.24}$$

Finally, the expression for the B-Spline curve is:

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u) \mathbf{P}_i. \tag{3.25}$$

The additional ingredient that results in NURBS is assigning a weight w_i to each control point $\mathbf{P}_i$, which we will assume to be strictly positive, i.e. $w_i > 0$. Thus, a NURBS curve is also a piecewise polynomial interpolation but, unlike in B-Spline curves, the terms are now weighted by these weights:

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}. \tag{3.26}$$

This expression can be reformulated using the rational basis functions:

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{3.27}$$

as

$$\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p}(u) \mathbf{P}_i. \tag{3.28}$$

Both bases are closely related, with $N_{i,p}(u)$ being a special case of $R_{i,p}(u)$ when $w_i = 1$. This means they share the following properties:

P.1 Partition of unity: $R_{i,p}(u) = 1$ for all $u \in [u_0, u_m]$.

P.2 Nonnegativity: $R_{i,p}(u) \geq 0$ for all i, p, and u.

P.3 Local support: $R_{i,p}(u) = 0$ if u is outside the interval $[u_i, u_{i+p+1})$. As a result, in any given knot span $[u_j, u_{j+1})$, at most $p + 1$ of the $R_{i,p}(u)$ are non-zero, namely the functions $R_{j-p,p}(u), \ldots, R_{j,p}(u)$.

P.4 Continuity: All derivatives of $R_{i,p}(u)$ exist within the interior of the ith to $(i+p)$th knot spans. At a knot, $R_{i,p}(u)$ is $p - k$ times differentiable, where k is the multiplicity of the knot.

In Fig. 3.6, three distinct curves obtained from the same control polygon are presented for comparison. What differentiates the curves from one another is the interpolation basis, represented on the right side, which corresponds to a different parameterization in each case. On the X-axis, the coordinates of the break points are shown, with the knot multiplicity within the knot vector indicated by the superscript. It can be observed how changes in the knot vector directly affect the shape of the interpolation functions, modifying their maximum value, the point at which it is reached, and even the number of support spans, and with it, the polynomial degree. On the right, the physical curves as well as their associated control polygons are shown. The numbers indicate the ordering of the points within the control polygon, while the size of each circle refers to the weight associated with each point. We can observe that the weights also affect the interpolation functions, primarily influencing their amplitude.

The properties P.1,P.2,P.3 and P.4 of the interpolation basis provide NURBS curves with several important geometric characteristics:

1. Affine invariance: an affine transformation applied to the curve is done by applying it to the control points.

2. Strong convex hull property: if $u \in [u_i, u_{i+1})$, then $\mathbf{C}(u)$ lies within the convex hull of the control points $\mathbf{P}_{i-p}, \ldots, \mathbf{P}_i$.

3. Variational diminishing property: the number of intersection points between a straight line and $\mathbf{C}(u)$ is always less than or equal to the number of intersection points with the control polygon.

4. Local approximation: modifying $\mathbf{P}_i$ or w_i only affects the portion of $\mathbf{C}(u)$ in the interval $[u_i, u_{i+p+1})$.

3.2.2. Knot insertion

The definition of a NURBS curve is not unique, and modifications can be made to the initial definition without affecting its geometry. Knot insertion or refinement of the knot vector is one of these transformations and is particularly relevant when

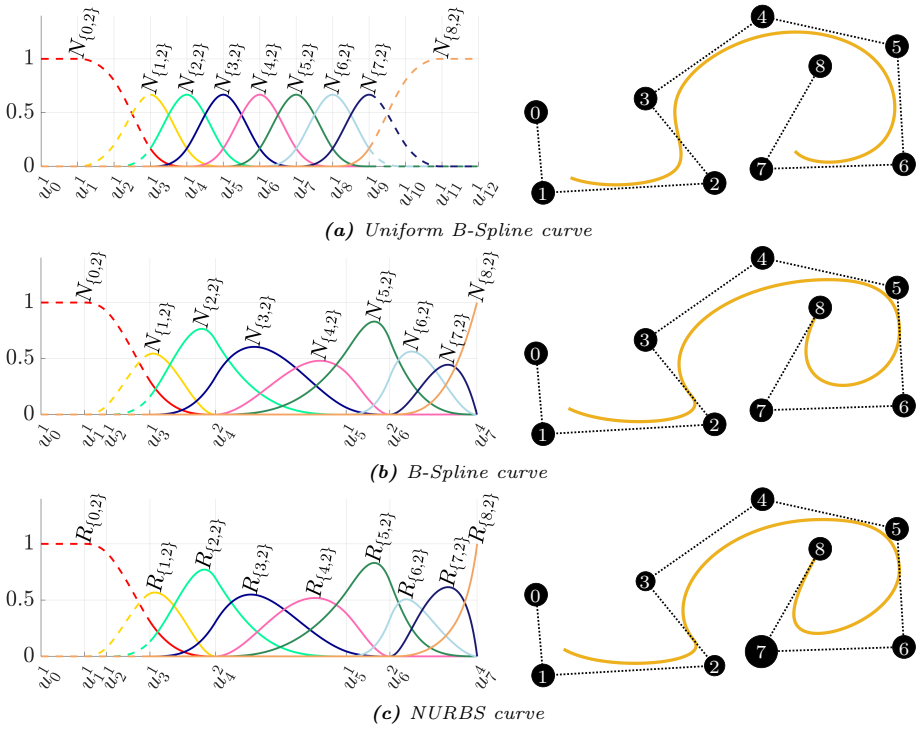


Figure 3.6: *Three 3rd-order parametric curves with the same $\mathcal{P}$. Interpolation basis (on the left) and physical representation (on the right).*

manipulating NURBS. The problem consists of obtaining the control polygon $\overline{\mathbf{P}}_i$ and the weights $\overline{w}_i$ such that:

$$\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i = \sum_{j=0}^{\overline{n}} \overline{N}_{j,p}(u) \overline{w}_j \overline{\mathbf{P}}_j \tag{3.29}$$

where $\overline{N}_{j,p}(u)$ is defined over the knot vector $\overline{U}$, which is the union of the knot vector U and the set of knots to be introduced. Expanding this equation results in a linear system with a unique solution that can be efficiently solved.

There are multiple practical applications of this procedure. Knot insertion, combined with the property of local support, allows for local modifications of the curve during the design phase. It can also be used to evaluate the curve and its derivatives. If knots that are already present in the knot vector are added, increasing their multiplicity makes the control polygon a closer approximation to the curve around that parameter value. In fact, when a knot u_i has a multiplicity k greater than the degree p of the curve, at least one control point $\overline{\mathbf{P}}_j$ is located on the

curve point $\mathbf{C}(u_i)$, and the curve at that point is tangent to the control polygon, $\mathbf{C}'(u_i) = \lambda_j(\overline{\mathbf{P}}_j - \overline{\mathbf{P}}_{j-1}) = \lambda_{j+1}(\overline{\mathbf{P}}_{j+1} - \overline{\mathbf{P}}_j)$.

Curve division is another widespread application. When a knot reaches a multiplicity greater than the degree of the curve, the parameter value splits the curve into two independent segments since no basis function extends its support to both sides of the knot.

The refinement process will be very useful in the cgDG context. In Section 3.4.1 , it will allow us to modify the definition of the NURBS to align the endpoints of the curves with the endpoints of the control polygon. Curves that exhibit this characteristic are called clamped curves. Additionally, it enables dividing curves at points of interest, such as intersections, break points, etc.

3.2.3. The NURBSLab toolbox

Compared to other simulation approaches, the geometric model plays a particularly central role in the context of cgDG. The pursuit of a methodology capable of solving equations over a domain that closely matches the one defined by the user has led to the integration of the geometric model into the core of the cgDG framework. It is directly involved in processes such as mesh generation, element classification, quadrature design and the application of boundary conditions, among others. In fact, many of the contributions presented in this thesis are techniques targeting to incorporate geometric models into cgDG.

To implement, validate, and refine these strategies, a comprehensive set of tools for working with NURBS is essential. At the outset of this research, the available libraries for NURBS — among which *NURBS Toolbox* by D.M. Spink [143] stands out — were few and quite limited. None of them provided all the tools, usage flexibility, or editing capabilities required by our developments. Additionally, these libraries typically relied on rigid data structures, necessitating the creation of translation layers that significantly hindered computational performance.

To address these limitations, a custom library was developed from scratch in the MATLAB® programming language. This library gathers a wide range of functionalities for the evaluation, manipulation and visualization of NURBS in both 2D and 3D settings, supporting curves and surfaces alike. We extended its capabilities beyond the specific needs of this thesis since it was conceived to fill the gap identified during our initial survey and to provide a broadly applicable resource for the community. In this regard, its implementation also follows a generalist design philosophy, avoiding the use of complex or restrictive structures. The toolbox is made available to the community by the name of *NURBSLab* along with a detailed breakdown of the functions that make it up, which can be found in Appendix B.

3.3. NURBS-Enhanced Integration

A fundamental step in the workflow of any finite element solver is the evaluation of operators in the weak form. Most finite element methods (FEM) discard the geometric model after meshing and perform the integration over the finite elements. Ultimately, it turns into the actual domain being approximated by another domain whose boundary is defined by polynomial segments, typically linear. The geometric discrepancies between these domains have been documented to significantly reduce the results' accuracy in many scenarios (see [144]).

Various alternatives have been proposed to overcome this limitation, including isogeometric analysis (IGA) [145] and the NURBS-enhanced finite element method (NEFEM) [136]. Both technologies feature geometric consistency by regarding NURBS models within the integration stage. IGA use NURBS not only to model the geometry but also as basis for approximation. This paradigm shift allows them to capture the geometry definition exactly but it also increases the complexity of the codes and meshes still depend on geometry features which may lead to locally overrefined meshes. On the contrary, NEFEM is mostly a standard high-order FEM except for those elements along the boundary. The solution in these elements still approximates using polynomial functions but sophisticated, NURBS-Enhanced (NE) integration strategies are used to effectively capture the geometry of the boundaries.

NE integration strategies may be applied beyond the NEFEM setting and are attractive due to their capability to efficiently handle complex geometries even with high-order approximations. The 2D version of cgDG leverages NE integration strategies to achieve geometry consistency. Section 3.3.1 briefly presents the general integration setting and its main limitations. Following, Section 3.3.2 describes its implementation in the cgDG's context. Section 3.3.2.1 concludes this section on NE integration proposing a improved version that broadens the application range of these strategies allowing integration domains with up to two curved contours. Later in this chapter, Section 3.4 addresses the robust and automatic computation of integration domains in cgDG that fulfill geometric restrictions highlighted by the end of Section 3.3.1.

3.3.1. Integration in NEFEM

The integral of an arbitrary function $f(\mathbf{x})$ within a NEFEM boundary element is computed, in a similar fashion as previously studied integrals, regarding the physical space $\mathbf{x}$ and space of the reference element for integration $\boldsymbol{\tau}$:

$$\int_{\Omega_e} f(\mathbf{x}) d\Omega = \int_{\Pi} f\left(\boldsymbol{\tau} \underset{\mathbf{u}}{\rightarrow} \mathbf{x}\right) \left|\mathbf{J}_{\boldsymbol{\tau} \underset{\mathbf{u}}{\rightarrow} \mathbf{x}}\right| d\Pi \tag{3.30}$$

The main difference here is that the mapping $\boldsymbol{\tau} \underset{\mathbf{u}}{\rightarrow} \mathbf{x}$ considers exactly the same definition of the boundary as the CAD model of the geometry. $\boldsymbol{\tau} \underset{\mathbf{u}}{\rightarrow} \mathbf{x}$ is computed in

two steps. First, the reference domain in $\boldsymbol{\tau}$ is mapped into the domain $[u^i, u^e] \times [0, 1]$ in the semiparametric space $\mathbf{u}$, where $\{u^i, u^e\}$ are points in the parametric space of the boundary. Second, a transformation of the form:

$$\mathbf{u} \to \mathbf{x} := \mathbf{C}(u_1)(1 - u_2) + \mathbf{V}u_2 \tag{3.31}$$

that interpolates the image of the curve $\mathbf{C}$ defining the boundary of the domain and a point $\mathbf{V}$ in $\mathbf{x}$ is applied. In order to ensure convergence, the points u^i and u^e need to be chosen so the mapping $\mathbf{u} \to \mathbf{x}$ has a unique definition. Given that the function of the curve varies from one knot span to another, the integral is splitted regarding the changes in the definition of the contour. For an integral that extends along a curve spanning ns knot spans, the domain Ω_e splits into a set of non-overlapping subdomains $\{\Omega_{e,i} : i \in [1, ns]\}$ such that $\Omega_{e,i} \cap \Omega_{e,i+1}$ is the straight segment connecting a point where the curve definition changes with the internal vertex $\mathbf{V}$. With this new division, the evaluation of the integral becomes:

$$\int_{\Omega_e} f(\mathbf{x}) d\Omega = \sum_{i=1}^{ns} \int_{\Omega_{e,i}} f(\mathbf{x}) d\Omega = \sum_{i=1}^{ns} \int_{\Pi} f\left(\boldsymbol{\tau} \xrightarrow[\mathbf{u}]{i} \mathbf{x}\right) \left|\mathbf{J}_{\boldsymbol{\tau} \xrightarrow[\mathbf{u}]{i} \mathbf{x}}\right| d\Pi \tag{3.32}$$

It should be noted that standard polynomial integration quadratures fail to compute the exact value of the integral. As we introduce NURBS definition of the boundary, both the mapping $\mathbf{u} \to \mathbf{x}$ and the integrand considered for numerical integration became rational. In [136], several options are evaluated, and the most suitable approach is found to be the use of tensor products of one-dimensional quadratures. In particular, the studies show that, for an interpolation element of degree p, using quadratures with $p+2$ points in the parameter direction and $p+1$ in the complementary direction achieves the maximum level of accuracy. The number of points in the first direction can be reduced to $p+1$ if the curve parameterization is smooth, meaning that the ratio of lengths between the parameter and its physical space representation changes slowly.

An important aspect to consider is that the mapping $\mathbf{u} \to \mathbf{x}$ is linear in the second direction. This means that in cases where the boundary is complex or highly curved, the projection of space Π may extend beyond the domain Ω_e. In Fig. 3.7, the projection of the segment connecting the blue points through the reference domain has an image in physical space such that the red-marked segment falls outside Ω_e. This extension of the integration domain is problematic for functions defined in Ω_e because sampling points may fall outside their region of support. One way to handle these cases is to extrapolate the function values, but this is not always possible. In the NEFEM context, the mapping $\mathbf{u} \to \mathbf{x}$ is required to be bijective to avoid this situation. Geometrically, it is achieved if the vertex opposite the curve in boundary elements is visible from all points on the curve within the element. Such a constraint renders the mesh generation process quite complex and it is a major barrier limiting the use of NEFEM.

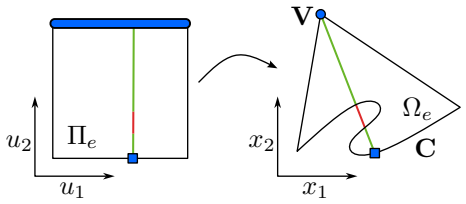


Figure 3.7: *Example of mapping that extends over the domain's boundary*

When evaluating integrals over the boundary, a similar procedure is followed. The result will be quadratures projected from the parametric space of the curves.

3.3.2. NURBS-Enhanced integration in cgFEM

This thesis considers high-order interpolations, which necessitate a sufficiently accurate integration strategy to ensure that errors arising from domain approximation do not compromise the theoretical convergence of the method. To address this requirement, the NE integration techniques have been incorporated into the cgFEM framework. Following [29], integration over boundary elements is performed through subdomain decomposition. However, unlike the cgFEM that mostly considers subdomains with polynomial shapes, the cgDG employs NE techniques on subdomains over the boundary to retain the exact rational representation of the geometry.

Consider a boundary element $\mathcal{I}_e$ of the cgFEM mesh containing the physical domain Ω_e, which is divided into a series of integration cells $\{\Omega_i\}$ such that the curve does not change its definition within any of them. Additionally, two sets, c_S and c_N, are defined such that $\{\Omega_k : k \in c_S\}$ and $\{\Omega_k : k \in c_N\}$ represent the set of internal subdomains and the set of subdomains with a side on the boundary, respectively. If the function is defined in the the reference element coordinate system, the integral over the boundary element is:

$$\begin{aligned}\int_{\mathcal{I}_e} f(\mathbf{x})d\Omega &= \int_{\Omega_e} f(\mathbf{x})d\Omega \\ &= \sum_{i\in n_S}\int_{\Pi} f(\boldsymbol{\tau} \xrightarrow{i} \boldsymbol{\xi}) \left|\mathbf{J}_{\boldsymbol{\tau}\xrightarrow{i}\mathbf{x}}\right| d\Pi + \sum_{i\in n_N}\int_{\Pi} f(\boldsymbol{\tau} \xrightarrow[\mathbf{u}]{j} \boldsymbol{\xi}) \left|\mathbf{J}_{\boldsymbol{\tau}\xrightarrow[\mathbf{u}]{j}\mathbf{x}}\right| d\Pi.\end{aligned} \tag{3.33}$$

The transformations $\boldsymbol{\tau} \xrightarrow{i} \boldsymbol{\xi}$ and $\boldsymbol{\tau} \xrightarrow{i} \mathbf{x}$ are affine, allowing simplifications compared to the general case. Additionally, it is possible to apply quadratures with a reduced number of points (lower-order quadrature or changing the reference domain) while maintaining the same level of accuracy. Applying these modifications, the integral over a boundary element becomes:

$$\int_{\mathcal{I}_e} f(\mathbf{x})d\Omega = \sum_{i\in n_S} \left|\mathbf{J}_{\boldsymbol{\tau}\xrightarrow{i}\mathbf{x}}\right| \int_{\Pi'} f(\boldsymbol{\tau} \xrightarrow{i} \boldsymbol{\xi})d\Pi' + \sum_{i\in n_N} \int_{\Pi} f(\boldsymbol{\tau} \xrightarrow[\mathbf{u}]{j} \boldsymbol{\xi}) \left|\mathbf{J}_{\boldsymbol{\tau}\xrightarrow[\mathbf{u}]{j}\mathbf{x}}\right| d\Pi. \quad (3.34)$$

3.3.2.1. An improved version of the NE integration

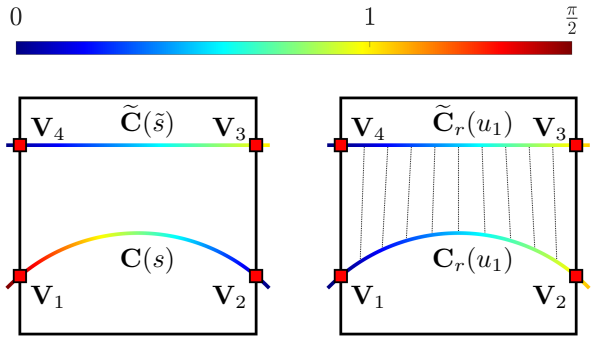
NE integration in NEFEM only considers domains with one curved edge, which requires cell refinement in some situations. To overcome this limitation, a version of the NE integration technique that can handle two curved edges has been developed. Consider a curved square subdomain with vertices $\{\mathbf{V}_i : i \in [1,4]\}$ arranged counterclockwise, as shown in Fig. 3.8. If $\{\mathbf{C}(s^i), \mathbf{C}(s^e)\}$ and $\{\widetilde{\mathbf{C}}(\tilde{s}^i), \widetilde{\mathbf{C}}(\tilde{s}^e)\}$ are equivalent to $\{\mathbf{V}_1, \mathbf{V}_2\}$ and $\{\mathbf{V}_3, \mathbf{V}_4\}$, respectively, it is possible to obtain a mapping from an unique auxiliary parametric space $\mathbf{u}$ to the physical space $\mathbf{x}$.

To achieve this, the curves are reparameterized so that both are supported on the same parameter range, and the parameter associated with $\mathbf{V}_1$ is the same as that associated with $\mathbf{V}_4$. These two conditions imply that $\mathbf{V}_2$ and $\mathbf{V}_3$ are also obtained for the same parameter value. For the new parameterization to be valid, the function must be monotonic and without changes in definition. In this work, polynomial transformations are generally considered, although affine transformations are typically used. Using linear mappings does not increase the NURBS order (allowing the number of integration points to be reduced), and only the definition of one curve needs to be modified. Considering the reparameterization of the curves $\mathbf{C}(s)$ and $\widetilde{\mathbf{C}}(\tilde{s})$ as $\mathbf{C}_r(u_1)$ and $\widetilde{\mathbf{C}}_r(u_1)$, respectively, the mapping from the parametric space to the physical space, namely $\mathbf{u} \to \mathbf{x}$, becomes:

$$\mathbf{u} \to \mathbf{x} := (1-u_2)\mathbf{C}_r(u_1) + u_2\widetilde{\mathbf{C}}_r(u_1) \quad (3.35)$$

Although Equation (3.35) has been formulated for square cells with two opposing curved sides, by collapsing points, it can be applied to the configurations shown in Fig. 3.9. If we set $\mathbf{V}_2 = \mathbf{V}_3$, we obtain a triangle with two curved sides. If, additionally, $\mathbf{V}_1 = \mathbf{V}_4$, we obtain a domain bounded by two curves. These configurations will arise during the process of subdividing intersected elements into integration cells.

When applying integration quadratures, the criteria will be the same as for the standard NE case. Since the same parameterization is used for both curves that bound the domain, the same quadrature can be used for both. However, changes in definition along either boundary may introduce intervals where different integration rules must be applied. The quadrature for the planar domain is obtained by tensor product with the quadrature applied in the affine direction of the transformation.



(a) *Initial parameterization of the curves*

(b) *Reparameterized curves. Dotted lines represent isoparametric curves when (3.35) is considered.*

Figure 3.8: *Reparameterization of curves to enable unique mapping over subdomains with multiple curved edges. The color axis used to represent the parameter's value at each point of the curves has been plotted on top.*

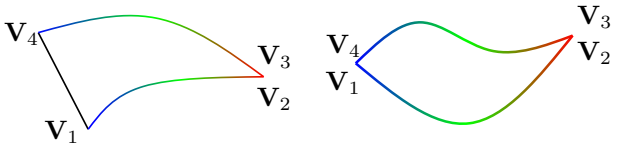


Figure 3.9: *Curved subdomains with reparameterized curves and collapsed points.*

3.4. Integration Decomposition

The design of integration quadratures for intersected elements begins with the computation of integration subdomains. The portion of the physical domain contained within an element must be split into regions with simpler geometric shapes. Specifically, these regions must be mappable from reference domains for which standard quadratures are defined, typically triangles and squares in the planar case. As discussed in Section 3.3.1, NURBS-Enhanced (NE) integration introduces an additional constraint: these transformations must be bijective.

This section presents a fully automated methodology that consistently computes such integration subdomains. The approach is based on robust geometric indicators that err on the side of caution, ensuring that only NE-integrable subdomains are accepted. By leveraging the properties of NURBS, it avoids solving nonlinear problems while preserving robustness, thereby accelerating computations. The most relevant properties supporting the proposed methodology are summarized below:

P.1 The endpoints of a clamped NURBS curve coincide with the endpoints of the control polygon.

P.2 A clamped NURBS curve is tangent to its control polygon at the endpoints.

P.3 The NURBS curve is contained within the convex hull of the control polygon that defines the curve for each span.

P.4 The number of intersections between a NURBS curve and a plane will always be less than or equal to the number of intersections of its control polygon.

A central concept in the methodology is that of geometric visibility, which was mention in Section 3.3.1. This notion plays a pivotal role in ensuring the bijectivity of the mappings from reference integration domains to physical subdomains. Accordingly, before detailing the adaptation strategy, it is appropriate to first provide a formal definition of visibility:

Definition 3.4.1. For an arbitrary domain Ω, we say that a point $p \in \Omega$ is visible from another point $q \in \Omega$ if the line segment $\overline{pq} \subset \Omega$.

The structure of this section is as follows. Section 3.4.1 presents the changes in the mathematical model of the domain that need to be applied before enter into the precedure in order to achieve robustness. Next, Section 3.4.2 introduces two procedures for generating initial subdomains based on point clouds and topological information. These subdomains possess simple geometric shapes, but the associated mappings accounting for the NURBS contour definitions are not guaranteed to be bijective. Sections 3.4.3 and 3.4.4 respectively address the evaluation and enforcement of bijectivity. First, strategies are presented for predicting whether a given subdomain is suitable for NE integration, and for identifying the cause of invalidity when applicable. Finally, the refinement actions applied to each subdomain category ultimately yielding to NE integrable decompositions are described.

3.4.1. Preprocessing geometry definition

The subdomain decomposition procedure requires geometric models that satisfy specific conditions. It is assumed that all NURBS curves defining the domain boundary are clamped and defined over a single parametric interval. Additionally, the geometry is required to intersect element edges only at curve endpoints. These conditions can be readily satisfied by applying the knot insertion strategy described in Section 3.2.2 to the curves in the original geometric model. This strategy enables the subdivision of curves at geometric breakpoints and mesh-geometry intersection points.

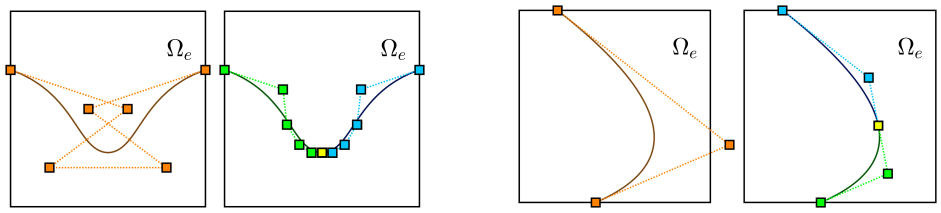
Next, let us introduce the concept of the polygon associated with a domain. These polygons are particularly valuable, as they offer a more tractable representation of the essential geometric features of the domain:

Definition 3.4.2. The polygon $\mathcal{D}^\star$ associated with the domain $\mathcal{D}$ whose boundary is defined by the union of NURBS curves $\{\mathbf{C}_1, \ldots, \mathbf{C}_m\}$ with control polygons $\{\mathcal{P}_1, \cdots, \mathcal{P}_m\}$ is the bounded set of points enclosed by the concatenated control polygons of the curves.

The algorithm described in Section 3.4.4 is not capable of producing a valid decomposition when the polygon of the domain within an element is self-intersecting. This algorithm attempts to divide the domain by identifying internal points with broad visibility of the boundary. However, considering the local domain classification criterion introduced in Section 3.1.3, a self-intersecting polygon will always contain sections of the boundary that remain hidden from any interior point. Furthermore, such polygons inevitably define regions in the plane where the classification of points is not consistent.

Two scenarios have been identified that lead to polygons with these characteristics:

- **Tangled control polygon** (Fig. 3.10a): Although, by hypothesis, the domain is continuous and, therefore, its boundary has no intersections, in higher-order curves with complex geometry, the control polygon may fold onto itself.
- **Control point outside the element domain** (Fig. 3.10b): By construction, subdomain boundaries that fall along element edges cannot be intersected by the geometry. However, the control polygons associated with the curves may intersect the element boundary as they are only approximations of the actual curves.



(a) *Tangled control polygon* ***(b)*** *Control point outside the element domain*

Figure 3.10: *Scenarios of nearly self-intersected subdomains*

From Fig. 3.10 we realize that the root of the problem lies in the discrepancy between the actual NURBS curves and their representation via control polygons. To address this, we adopt an iterative refinement strategy in which the knot vector of any NURBS curve whose control polygon exhibits intersections is split. In Fig. 3.10, orange curves correspond to control polygons that intersect. Dividing the curves at the point marked in yellow produces green and blue control polygons that now meet the desired characteristics.

This process is guaranteed to converge, since refining the knot vector causes the control polygon to approach the geometry of the actual curve. The final outcome

is a simple polygon (a polygon without intersections or holes) suitable for further processing using the general algorithm.

3.4.2. Initial Domain Decomposition

The proposed methodology is structured as an iterative refinement process in which an initial decomposition is progressively improved until all required constraints are satisfied. The only requirement imposed on the initial decomposition is that it consists of subdomains with simple shapes. This allows the use of well-established techniques such as constrained Delaunay triangulation. Additionally, this thesis explores an alternative decomposition approach based on the marching squares algorithm. This second method offers significantly greater computational efficiency but is more limited in applicability since it can only be employed when the intersection pattern between the geometry and the element boundary is simple enough.

3.4.2.1. Marching Squares

The marching squares algorithm [146] was developed to reconstruct the boundary of a domain based on a regular grid of points, each classified as either inside or outside the domain. By examining the classification of the four vertices of each grid cell, one can define a total of 16 distinct patterns representing all possible configurations. When accounting for rotational symmetries, this set reduces to 6 unique patterns. For each pattern, the inferred position of the domain boundary within the cell must be consistent with the vertex classifications. In 5 of these cases, the domain's location is well-defined. The last pattern is ambiguous regarding the precise placement of the boundary but does not appear if the grid is fine enough.

Fig. 3.11 illustrates the outcome of the marching squares-based decomposition strategy for each of the five unambiguous patterns. The key idea is that knowing the classification of an element's corners is sufficient to determine the topology of both the subdomains and edges within it. Indeed, not only their topologies but also their shapes and positions relative to both the domain and the element. This information can be computed in advance and stored for later use, thereby eliminating the need for subdomain decomposition during the integration loop. Moreover, the decomposition can be optimized to minimize the number of subdomains. The implementation presented in this thesis considers both triangular and quadrilateral subdomains, which ultimately reduces the number of required integration points and lowers the overall computational cost.

Unlike the original algorithm, where the boundary is approximated by straight segments passing through the midpoint of the subdomain edge, for integration purposes, the intersection points between the Cartesian grid and the boundary are retrieved, along with the boundary's definition within each element. This allows the boundary to be recovered exactly.

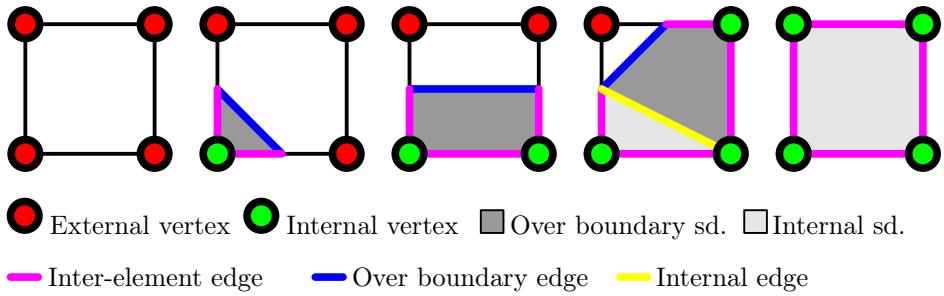


Figure 3.11: *Marching cube patterns and topology of the decomposition with subdomain (sd.) and edge classification*

This process is highly efficient, as the topological information is precomputed, stored, and, within the loop, only needs to associate the corresponding information with each entity.

3.4.2.2. Delaunay Triangulation

The procedure based on the marching squares algorithm can only be applied in cases where the geometry definition within the subdomain does not vary, and no edge of the element has been intersected more than once. The limitation arises from the piecewise linear approximation of the boundary and the nodal coding of its relative position.

In cases where the marching squares strategy cannot be applied, decomposition into subdomains via constrained Delaunay triangulation is used. The points considered for triangulation are the intersections of the boundary with the mesh axes, the element nodes, and the points where the boundary definition changes. Additionally, the connectivity information of the points on the boundary is provided to ensure that none of the subdomains are intersected by the boundary. For the design of integration quadratures, only the subdomains located within the domain of interest are considered.

3.4.3. Labeling stage

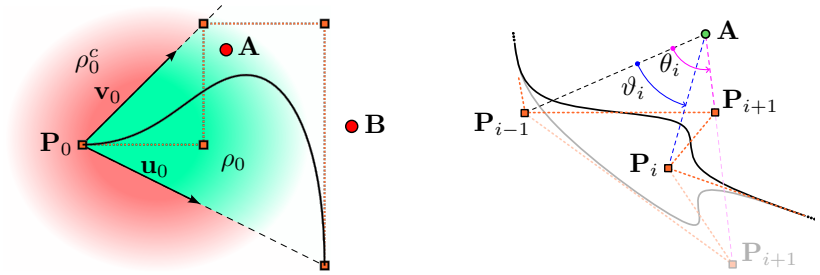
The geometry of the integration subdomains can be complex, and to effectively apply NE integration, it is necessary to ensure that the visibility condition outlined in Section 3.3 is met.

Consider a single subdomain with a curved side resulting form any of the initial domain decompositions presented in Section 3.4.2. The verification process begins by obtaining the equation of the NURBS curve corresponding to the section of the boundary within the subdomain. Since the decomposition into subdomains is per-

formed considering the NURBS definition changes and the intersection points, all curves generated at this stage contain necessarily a single span.

Two vectors, $\mathbf{u}_0$ and $\mathbf{v}_0$, are calculated, originating from the first vertex of the curve $\mathbf{P}_0$. The vectors are defined such that the angle between them is minimized, and all control polygon points can be expressed as $\mathbf{P}_0 + a_0\mathbf{u}_0 + b_0\mathbf{v}_0$, where a_0, b_0 are positive values. We denote ρ_0 as the region of space where a_0, b_0 take positive values.

According to Property P.3, we can be certain that any point located in the region ρ_0^c can be connected to $\mathbf{P}_0$ by a straight line segment without intersecting the NURBS curve. In contrast, for points located in the region ρ_0, this cannot be guaranteed without explicitly checking for intersections with the proper curve. As illustrated in Fig. 3.12a, both points $\mathbf{A}$ and $\mathbf{B}$ lie within ρ_0; however, while $\mathbf{A}$ is visible from $\mathbf{P}_0$, the curve obstructs the line of sight between $\mathbf{P}_0$ and $\mathbf{B}$. The same procedure is applied at the endpoint $\mathbf{P}_n$ of the curve to define the region ρ_n.



(a) Auxiliary coordinate system for self-intersection checking

(b) Auxiliary angles for full visibility checking

Figure 3.12: *Geometric tools used to measure subdomain's validity*

The first validation criterion will be to check whether the vertex $\mathbf{U}$ opposite to the curved side is located in the region $\rho_0^c \cap \rho_n^c$. If not, there is a possibility that the subdomain's boundary self-intersects. For this reason, the studied subdomain is marked as *self-intersected* subdomain for further treatment. The adjacent subdomain that connects through a potentially intersected edge is also marked as invalid.

The inspection process continues by verifying full visibility. Consider a NURBS curve defined by a control polygon in non-homogeneous coordinates $\{\mathbf{P}_0, \ldots, \mathbf{P}_n\}$, and let $\mathbf{A}$ be an observation point. According to Property P.4, the entire curve is visible from $\mathbf{A}$ if the entire control polygon is visible from that point. This condition is satisfied if, for all $i \in \{1, 2, \ldots, n-1\}$, the angle $\theta_i := \angle(\mathbf{P}_{i-1}, \mathbf{A}, \mathbf{P}_i)$ is less than or equal to the angle $\vartheta_i := \angle(\mathbf{P}_{i-1}, \mathbf{A}, \mathbf{P}_{i+1})$, and if the total sum of the angles $\{\theta_0, \vartheta_0, \ldots, \vartheta_{n-1}\}$ does not exceed 2π.

As illustrated in Fig. 3.12b, a case where $\theta_i > \vartheta_i$ indicates that the segment $\overline{\mathbf{P}_{i-1}\mathbf{P}_i}$ partially or fully occludes the segment $\overline{\mathbf{P}_i\mathbf{P}_{i+1}}$ from the viewpoint $\mathbf{A}$, or vice versa. Applying this visibility criterion from the vertex $\mathbf{U}$ opposite the curved side of a boundary subdomain to the portion of the curve contained within it allows for the

detection of subdomains that likely exhibit non-bijective mappings. Such subdomains are considered invalid and are labeled as *surjective* subdomains.

The labeling procedure results in triangle subdomains being classified as valid or invalid among which self-intersected and surjective subdomains are differentiated (see Fig. 3.13).

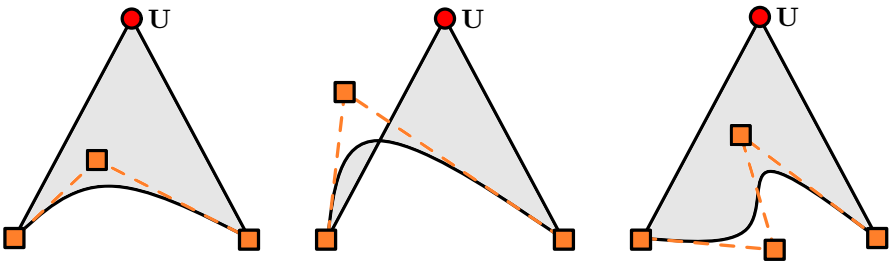


Figure 3.13: *Categories of integration subdomains: valid (left), self-intersected (center) and surjective (right)*

It is important to note that the criteria used for this labeling rely on approximations of the visibility condition between points. These approximations are accurate when the classification is positive but uncertain when the classification is negative. As a result, the labeling is conservative: valid subdomains may be incorrectly classified as invalid, but invalid subdomains are never accepted as valid. This restrictive approach ensures the robustness of the method while the use of approximations significantly reduces computational costs.

3.4.4. Refinement stage

The refinement strategy builds upon the polygons associated with curved domains (see Definition 3.4.2), the properties of NURBS curves (see Properties P.1, P.2, P.3, and P.4), and the visibility polygons associated with points. We begin by introducing the concept of a visibility polygon, followed by several remarks on its computation.

Definition 3.4.3. The visibility polygon of a point $\mathbf{p}$ within a domain Ω, which may or may not be bounded, is defined as the set of points in Ω that are visible from $\mathbf{p}$:

$$\mathcal{V}_{\mathbf{p}}^{1} = \{\mathbf{q} \in \Omega : \overline{\mathbf{pq}} \subset \Omega\}$$

Several algorithms exist for computing visibility polygons, depending on the properties of the input domain. In the case of a simple polygon, the visibility polygon of an interior point can be computed using the algorithm proposed in [147] and later improved in [148]. This method is optimal and performs the computation in linear time.

The proposed decomposition strategy employs an extended version of the algorithm developed by [148]. The original algorithm cannot be used directly in our

context because it fails when any three points in the set composed of the polygon vertices and the viewpoint are collinear. Such configurations frequently occur in our application. Fig. 3.14 shows a representative example. A curve has been subdivided at $\mathbf{P}_3$ into two clamped curves, resulting in a new domain polygon that includes three collinear points: $\mathbf{P}_2$, $\mathbf{P}_3$, and $\mathbf{P}_4$. These points must necessarily be aligned due to the continuity of the curve at $\mathbf{P}_3$ and in accordance with Properties P.1 and P.2. The modifications introduced in our implementation remove this limitation from the original algorithm, enabling its application in our framework.

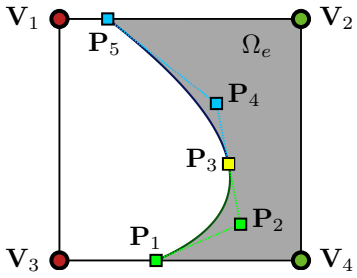


Figure 3.14: *Example of an invalid polygon for applying Lee's visibility algorithm*

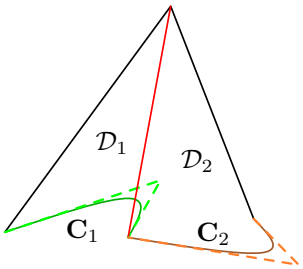
The procedure is organized into two phases, which are applied iteratively until the domain decomposition is completed. First, the geometric model of the portion of the original domain to be processed is identified and adapted. Then, an NE-integrable subdomain is defined over a curved boundary of the current working domain. At the end of each iteration, a portion of an initially marked invalid subdomain will have been successfully resolved, reducing the extent of the domain to be treated in the next iteration.

3.4.4.1. Detailed Description of the Process

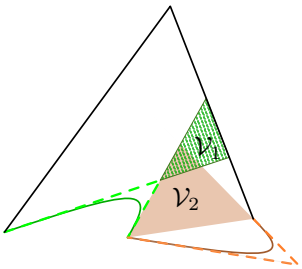
The following sections provide a detailed description of each stage of the proposed procedure. Fig. 3.15 presents an illustrative application example with accompanying explanations, which will help the reader more easily visualize the development of the algorithm.

3.4.4.1.1. Defining initial working domain

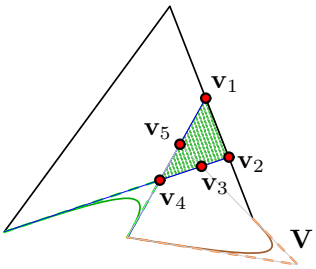
Following the notation introduced in Section 3.4.1, the working domain entering the procedure is an aggregation of invalid subdomains $\mathcal{A} = \bigcup_{i \in \mathbf{k}} \mathcal{D}_i$, such that $\overline{\mathcal{A}}$ forms a simple polygon. This implies that each initial working domain corresponds either to a surjective subdomain or to a simply connected aggregate of subdomains sharing intersected edges.



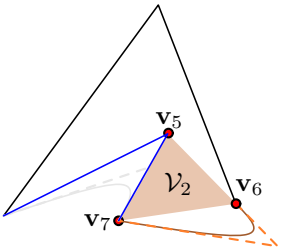
(a) *Starting from $\mathcal{D}_1$, the working domain is defined as the union $\mathcal{D}_1 \cup \mathcal{D}_2$, since $\mathcal{D}_1$ is a self-intersected subdomain connected to $\mathcal{D}_2$ through an intersected edge shown in red.*



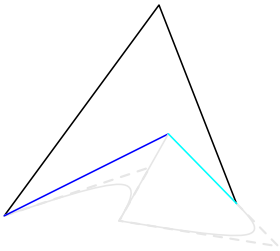
(b) *The geometry remains unchanged because both $\mathbf{C}_1$ and $\mathbf{C}_2$ have non-empty visibility polygons. Due to the smaller extent of $\mathcal{V}_1^2$, the curve $\mathbf{C}_1$ is addressed first.*



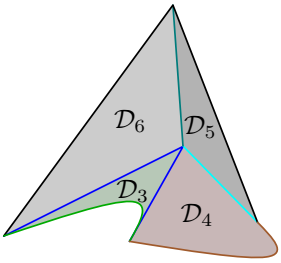
(c) *The partition of $\mathcal{V}_1^2$, considering $\partial\mathcal{V}_2^2$, yields five candidate points. Among these, $\mathbf{v}_3$, $\mathbf{v}_4$ and $\mathbf{v}_5$ are preferred, as they have visibility over both curves. Since all are interior to the working domain, their positions do not provide a basis for prioritization.*



(d) *The integration subdomain $\mathcal{D}_3$ is defined from $\mathbf{C}_1$ and $\mathbf{v}_5$ and is subtracted from the working domain, the updated boundary (in dark blue) appears. The visibility polygon vertices of the remaining curve are evaluated to close its subdomain. Although all candidates exhibit the same visibility, point $\mathbf{v}_5$ is already a vertex of the working domain.*



(e) *The integration subdomain $\mathcal{D}_4$, which connects $\mathbf{C}_2$ with $\mathbf{v}_5$, is then defined. At this point, the remaining working domain becomes a linear quadrilateral, which can be decomposed using constrained Delaunay triangulation.*



(f) *Finally, $\mathcal{D}_3$, $\mathcal{D}_4$, $\mathcal{D}_5$, and $\mathcal{D}_6$ form a decomposition compatible with NE integration, covering the domain initially spanned by the subdomains $\mathcal{D}_1$ and $\mathcal{D}_2$*

Figure 3.15: *Application example of the integration decomposition refinement procedure.*

3.4.4.1.2. Visibility-based geometric refinement

Once the working domain is identified, the mathematical model of its boundary is examined and adapted. Each curve is considered independently, and it is verified whether there exist points within the working domain from which the entire curve is visible. To this end, the polygon $\mathcal{A}^\star$ associated with the working domain is used to compute the visibility polygons for each control point of the curve. The intersection of these visibility polygons yields a region containing only but not all the points in $\overline{\mathcal{A}}$ from which the entire curve is visible.

To approach the actual visibility region $\mathcal{V}^2$ for the curve under consideration, the points of the intersection region located inside the convex hulls of the control polygons of all curves are removed. This subtraction is essential, as Property P.3 states that the curves traverse these regions but the actual path of the NURBS cannot be precisely known within them.

Fig. 3.16 illustrates a working domain $\mathcal{A}$ bounded by a fourth-degree curve. The associated polygon $\mathcal{A}^\star$ is decomposed into five subdomains: $\Omega_1^\star, \Omega_2^\star, \Omega_3^\star, \Omega_4^\star, \Omega_5^\star$. These subdomains are used to construct the visibility polygons for the curve's control points, as shown in Table 3.1. The example highlights the distinction between the intersection of individual visibility polygons and the visibility polygon of the entire curve. Notably, the latter excludes Ω_4, as it cannot be determined whether a point in this subdomain truly belongs to the working domain $\mathcal{A}$. For instance, any point lying on the line segment connecting control points $\mathbf{P}_2$ and $\mathbf{P}_3$ that pass through Ω_4 would in fact be outside the working domain.

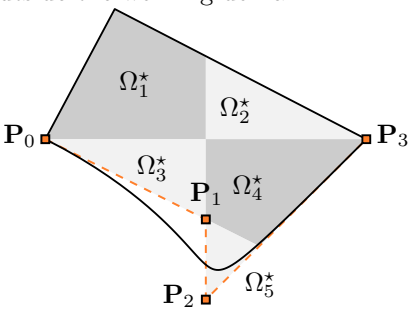


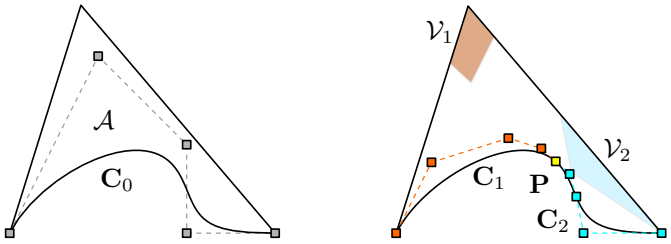
Figure 3.16: *Working domain decomposition*

Visibility polygon	Components
$\mathcal{V}_0^1$	$\Omega_1, \Omega_2^\star, \Omega_3^\star, \Omega_4^\star$
$\mathcal{V}_1^1$	$\Omega_1^\star, \Omega_2^\star, \Omega_3^\star, \Omega_4^\star, \Omega_5^\star$
$\mathcal{V}_2^1$	$\Omega_2^\star, \quad \Omega_4^\star, \Omega_5^\star$
$\mathcal{V}_3^1$	$\Omega_1^\star, \Omega_2^\star, \Omega_3^\star, \Omega_4^\star, \Omega_5^\star$
$\mathcal{V}_0^1 \cap \mathcal{V}_1^1 \cap \mathcal{V}_2^1 \cap \mathcal{V}_3^1$	$\Omega_2^\star, \quad \Omega_4^\star$
$\mathcal{V}_0^2$	$\Omega_2^\star$

Table 3.1: *Visibility polygons*

Cases may arise, such as the one shown in Fig. 3.17a, where some curves defining the boundary of the working domain have empty visibility polygons. For such configurations, it is not possible to guarantee the existence of a subdivision of the working domain that, while maintaining the boundary division, meets the requirements of NE integration. Curved segments with empty visibility polygons are recursively divided until a suitable boundary decomposition is achieved. Since all curves are smooth by assumption (see Section 3.4.1), this procedure is guaranteed to arrive to a solution. Fig. 3.17b illustrates how splitting curve $\mathbf{C}_0$ at $\mathbf{P}$, the equivalent configuration is

solvable, since the resulting curves $\mathbf{C}_1$ and $\mathbf{C}_2$ are entirely visible from $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively.



(a) *Unsolvable surjective subdomain.*

(b) *Solvable surjective subdomain.*

Figure 3.17: *Comparison of two versions of the same working domain before an after adequating its definition to the refinement stage.*

3.4.4.1.3. Solving a portion of the contour

The next stage of the algorithm consists in identifying the boundary segment that presents the greatest difficulty for inclusion in an integration subdomain. To this end, the area of the visibility polygons associated with each curve in the working domain is computed. The curve whose visibility polygon has the smallest area is selected as the target of the current iteration. Let us denote this curve by $\mathbf{C}_t$.

A triangular integration subdomain containing $\mathbf{C}_t$ can be defined by selecting any point within its visibility polygon $\mathcal{V}_t^2$. However, choosing this point arbitrarily would typically result in decompositions with a greater number of subdomains and, consequently, less efficient integration quadratures over the domain. Since the goal is to minimize the overall integration burden, the selection of this point must be performed with care. The following presents the procedure developed for this purpose.

First, the visibility polygon $\mathcal{V}_t^2$ is partitioned into a set $\{\Upsilon\}$, taking into account the visibility polygons of the remaining curves. The resulting set $\{\Upsilon\}$ consists of simple polygons, each grouping points that are simultaneously contained in the visibility polygons of the same set of curves. The candidate point $\mathbf{v}_t$, which will complete the definition of the integration subdomain, is chosen among the vertices of the polygons in the partition $\{\Upsilon\}$.

These candidate points are ranked based on the number of curves visible from each. Preference is given to those belonging to the visibility polygons of a greater number of curves. If multiple candidates remain, priority is given to points that are already vertices of the working domain. If a new vertex must be introduced, it is placed on the boundary of the working domain whenever possible. Applying these selection criteria reduces the total number of integration subdomains required by the end of the process.

Finally, a triangular integration subdomain is defined, conforming to the curved boundary $\mathbf{C}_t$ and having the selected point $\mathbf{v}_t$ as the opposite vertex.

3.4.4.1.4. Updating working domain

The working domain is updated by subtracting the newly generated subdomain from the previous one. This updated configuration defines the starting point for the next iteration.

3.4.4.1.5. End of refinement process

The iterative process terminates when the updated working domain is either empty or has a purely linear boundary. In the latter case, the decomposition can be completed using constrained Delaunay triangulation.

3.4.4.2. Algorithmic resume of the process

In this section, the previously described procedures are summarized in the form of pseudo-code. Algorithm 1 presents the complete algorithm for computing NE-integrable subdomains $\{\mathcal{D}\}_{\mathrm{NE}}$, starting from a domain $\mathcal{D}$ that belongs to the set of invalid subdomains $\{\mathcal{D}\}_{\mathrm{I}}$. This procedure is executed iteratively with different inputs until all domains in $\{\mathcal{D}\}_{\mathrm{I}}$ have been successfully decomposed. Algorithms 2 and 3 define subprocesses, encapsulated as functions, which are invoked within the main workflow.

Algorithm 2 outlines the procedure for refining the definition of the boundary of the working domain $\mathcal{A}$, to ensure the existence of a solution to the decomposition problem. Algorithm 3, in turn, presents the process for computing and selecting the most suitable points within $\mathcal{A}$ to form a valid triangular integration subdomain with a curved boundary segment corresponding to the curve $\mathbf{C}_t$.

Algorithm 1 Solve invalid subdomain

function ComputeValidDecompostion($\mathcal{D}, \{\mathcal{D}\}_{\mathrm{I}}$)
- ▷ *Define working domain* ◁
- $\mathcal{A} \leftarrow \mathcal{D}$
- **while** $\mathcal{A}$ is self-intersected **do**
  - $\mathcal{D}_i \leftarrow \mathcal{D}_i \in \{\mathcal{D}\}_{\mathrm{I}}$ meeting $\mathcal{A}$ at an intersected edge
  - $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{D}_i$
- ▷ *Solve working domain* ◁
- $\{\mathcal{D}\}_{\mathrm{NE}} \leftarrow \emptyset$
- **while** $\mathcal{A} \neq \emptyset$ **do**
  - ▷ *Compute suitable definition of the working domain contour* ◁
  - $\mathcal{A} \leftarrow$ AdaptDomainDefinition($\mathcal{A}$)
  - ▷ *Compute visibility polygons and identify target contour* ◁
  - $\{\mathcal{V}^2\} \leftarrow$ visibility polygons of $\{\mathbf{C}\}$ in $\mathcal{A}$
  - $\mathcal{V}_t^2 \leftarrow \arg\inf_{\mathcal{V}_i^2 \in \{\mathcal{V}^2\}} |\mathcal{V}_i^2|$
  - ▷ *Compute the best location for the vertex opposite to target contour* ◁
  - $\mathbf{v}_t \leftarrow$ ComputeBestVertex($\mathcal{V}_t^2, \{\mathcal{V}^2\}, \mathcal{A}$)
  - ▷ *Define NE integrable subdomain over the target contour* ◁
  - $\mathcal{D}_t \leftarrow$ curved side subdomain $(\mathbf{C}_t\mathbf{v}_t)^{\triangle}$
  - $\{\mathcal{D}\}_{\mathrm{NE}} \leftarrow \{\mathcal{D}\}_{\mathrm{NE}} \cup \mathcal{D}_t$
  - ▷ *Update working domain* ◁
  - $\mathcal{A} \leftarrow \mathcal{A} \setminus \mathcal{D}_t$
  - ▷ *Check if the updated working domain is divisible into simplices* ◁
  - **if** $\mathcal{A}$ is a polygon **then**
    - $\{\mathcal{D}\}_{\mathrm{DT}} \leftarrow$ constrained Delaunay triangulation on $\mathcal{A}$
    - $\{\mathcal{D}\}_{\mathrm{NE}} \leftarrow \{\mathcal{D}\}_{\mathrm{NE}} \cup \{\mathcal{D}\}_{\mathrm{DT}}$
    - $\mathcal{A} \leftarrow \mathcal{A} \setminus \{\mathcal{D}\}_{\mathrm{DT}}$
- **return** $\{\mathcal{D}\}_{\mathrm{NE}}$

Algorithm 2 Compute appropriate definition of the contour

function AdaptDomainDefinition($\mathcal{A}$)
- ▷ *Recursively split NURBS curves bounding $\mathcal{A}$ until achieve visibility therein* ◁
- **repeat**
  - $\{\mathbf{C}\} \leftarrow$ set of curved sides in $\partial\mathcal{A}$
  - **for** $\mathbf{C}_i \in \{\mathbf{C}\}$ such that $\mathcal{V}_i^2 = \emptyset$ **do**
    - $\{\mathbf{C}_a, \mathbf{C}_b\} \leftarrow$ divide $\mathbf{C}_i$
    - $\partial\mathcal{A} \leftarrow \{\partial\mathcal{A} \setminus \mathbf{C}_i\} \cup \{\mathbf{C}_a, \mathbf{C}_b\}$
- **until** $\mathcal{V}_i^2 \neq \emptyset \; \forall \; \mathbf{C}_i \in \{\mathbf{C}\}$
- **return** $\mathcal{A}$

Algorithm 3 Compute best vertex opposite to selected curved side

function ComputeBestVertex($\mathcal{V}_t^2, \{\mathcal{V}^2\}, \mathcal{A}$)
- ▷ *Compute all vertex candidates* ◁
- $\{\Upsilon\} \leftarrow \mathcal{V}_t^2$
- **for** $\mathcal{V}_i^2 \in \{\mathcal{V}^2\} \setminus \mathcal{V}_t^2$ **do**
  - $\{\Upsilon\}^0 \leftarrow \{\Upsilon\}$
  - **for** $\Upsilon_j \in \{\Upsilon\}^0$ **do**
    - $\{v\} \leftarrow$ divide Υ_j through $\partial\mathcal{V}_i^2$
    - $\{\Upsilon\} \leftarrow (\{\Upsilon\} \setminus \Upsilon_j) \cup \{v\}$
- $\{\mathbf{v}\} \leftarrow$ set of $\{\Upsilon\}$'s vertices
- ▷ *Select best vertex candidates based on visibility of the curves* ◁
- $S \leftarrow 0$
- **for** $\mathbf{v}_i \in \{\mathbf{v}\}$ **do**
  - $s_i \leftarrow |\{j\colon \mathbf{v}_i \in \mathcal{V}_j^2 \text{ and } \mathcal{V}_j^2 \in \{\mathcal{V}^2\}\}|$
  - **if** $S < s_i$ **then**
    - $S \leftarrow s_i$
    - $\{\mathbf{v}\}^* \leftarrow \mathbf{v}_i$
  - **else if** $S = s_i$ **then**
    - $\{\mathbf{v}\}^* \leftarrow \{\mathbf{v}\}^* \cup \mathbf{v}_i$
- ▷ *Select the best vertex candidate based on its location w.r.t. $\mathcal{A}$* ◁
- **if** $\mathbf{v}_v \in \{\mathbf{v}\}^*$ and $\mathbf{v}_v$ is a $\mathcal{A}$ vertex **then**
  - $\mathbf{v}_t \leftarrow \mathbf{v}_v$
- **else if** $\mathbf{v}_e \in \{\mathbf{v}\}^*$ and $\mathbf{v}_e \in \partial\mathcal{A}$ **then**
  - $\mathbf{v}_t \leftarrow \mathbf{v}_e$
- **else**
  - $\mathbf{v}_t \leftarrow \mathbf{v}_n \in \{\mathbf{v}\}^*$
- **return** $\mathbf{v}_t$

3.5. Stabilized scheme

As previously mentioned in Section 1.2, the computational advantage exhibited by the DG formulation is closely related to the use of explicit integrators to advance in time. The stabilization strategy proposed in this section seeks to minimize the negative effect that the presence of cut elements in the mesh has on the maximum stable time step of the explicit time integrator.

3.5.1. Stability indicator

Our goal is to identify and address the stabilization of unstable degrees of freedom (DOFs) at the earliest stages of the computational workflow, thereby avoiding

redundant computations. The global stability of an explicit time integrator is governed by the Courant–Friedrichs–Lewy (CFL) condition [15], which depends on both the polynomial degree and the size of the support domain associated with the most restrictive element. Since the polynomial degree is not affected by the mesh-geometry interactions in cgDG, it is reasonable to conclude that the limiting factor regarding time integrator stability is the size of the support domain at intersected elements. We propose the use of the relative area α_e^r as a stability indicator to predict which intersected elements are likely to severely constrain the maximum stable time step. Following the notation introduced in the previous section, this parameter represents the ratio between the area of region r and the total area of element e, and is defined as:

$$\alpha_e^r = \frac{|\Omega_e^r|}{|K_e|}, \tag{3.36}$$

In Section 3.6.1.2 we show the convenience of this coefficient to predict unstable behaviours. The study of how its reduction affects the maximum step size shows a monotonic relationship between these magnitudes which justifies its use. Furthermore, it is a purely geometric parameter that is easy to evaluate in the initial stages of the algorithm. Finally, a set of DOFs will be considered unstable if the relative area of its support region is less than a certain value specified by the user α_{th}. At the end of the day, the value of the maximum stable time step that may be expected from the resulting discretization will be greater than or equal to that associated with an element with a relative area of α_{th}. Results in Section 3.6.1.2 support this hypothesis and confirm that the stabilization strategy we are about to describe effectively limits the maximum stable time step and eliminate its dependence on the cut elements.

Remark. Instability evaluation is performed at the level of isolated regions. For the sake of facilitating the comprehension of the algorithm, in the following we assume that each element contains one isolated region at most, $n_{ir} = 1$. Therefore, the DOFs linked to a region always correspond to those associated with the element that encompasses it, namely $\mathfrak{N}_e = \mathfrak{N}_e^1$. With that in mind, we simplify the notation using α_e and Ω_e instead of α_e^1 and Ω_e^1.

3.5.2. Key features of the stabilization strategy

The stabilization strategy proposed here is based on modifying the support domains of the interpolation functions. Once an unstable region is identified, its domain is reassigned to the polynomial approximation defined on a neighboring element considered to be stable. The resulting element is referred to as an *extended element*, as the weak form integrals associated with it extend beyond the boundary of the element used for stabilization.

The shape, nodal distribution and shape functions of the extended elements remain the same before and after the reassignment procedure. Therefore, as far as implementation is concerned, they can be treated as standard intersected elements

and do not entail any additional computational burden. Since the unstable regions are small with respect to the size of the extended element, problems related to polynomial instability due to extrapolation have not been detected. Once the reassignment has been completed, the DOFs originally associated with the unstable region have zero support and can be eliminated from the analysis mesh.

Fig. 3.18 compares two meshes: on the left, a mesh that does not account for stabilization; on the right, a stabilized version. For illustrative purposes, a one-dimensional domain Ω is considered. The conventional cgFEM discretization consists of three elements: K_1, K_2, and K_3, which cover the subdomains Ω_1, Ω_2, and Ω_3 of Ω, respectively. The shape functions $\mathbf{N}_3$ associated with element K_3 have limited support, restricted solely to Ω_3, leading to unstable behavior of the corresponding DOFs. In the stabilized configuration, element K_3 is removed, and the domain Ω_3 is reassigned to element K_2. Consequently, the shape functions $\mathbf{N}_2$ now have extended support over the enlarged region Ω_2', which includes both the original domain Ω_2 and its extrapolation onto Ω_3.

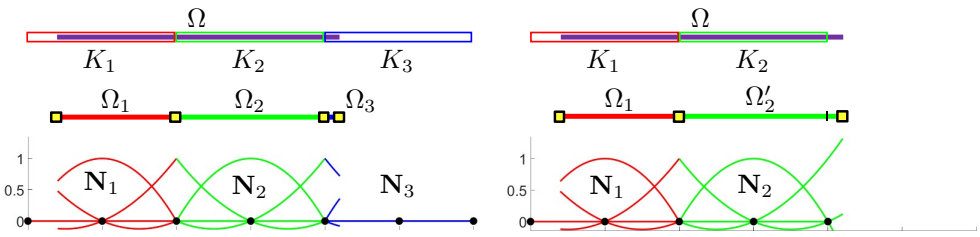


Figure 3.18: *Standard vs. Stabilized Cartesian discretization.*

To address the potential for polynomial instabilities caused by extrapolation, an alternative strategy for defining the approximate solution on extended elements was also investigated. In this alternative approach, the reference element was mapped to a rectangle that contained all the regions associated with each element at hand. As the support of the shape functions is completely within the boundary of the nodal distribution, no extrapolation is applied. In comparison with the approach adopted here, this alternative did not provide any major advantage in terms of the overall stability of the method while the departure from the cgFEM data structure of the mesh reduced the efficiency of the resulting scheme.

Removing the nodes associated to unstable elements solve the instability issues but also reduces the richness of the approximation basis. To minimize the effect of stabilization on the discretization error, we propose to add new DOFs to those elements that have been extended when necessary. Even beyond the 1D setting, we can easily do this thanks to the special features of the cgDG. On the one hand, DG formulation lets us locally enrich the spatial discretization regardless of the discretization used in the surrounding elements. On the other hand, the Cartesian nature of the elements in cgFEM invites to define nodal distributions as the tensor product of 1D nodal distributions. Following this approach, the polynomial degree of the approx-

imation in each direction becomes independent and it is possible to opt for denser nodal distributions in the extended direction alone. Thorough studies on the performance of this strategy are carried in Sections 3.6.1.3 and 3.6.2.3. We must carefully decide when to add nodes, since increasing the degree again reduces the stability of the discretization. Therefore, it will be necessary to achieve a balance between the quality of the spatial discretization and the size of the time step.

There is a difficulty that remains hidden in the example illustrated in Fig. 3.18. In the 1D case, there is only one possible element to extend since the neighborhood of a boundary element is restricted to a single element. However, as the number of dimensions grows, the size of the neighborhood may also increase. Since several candidates to extrapolate its solution may stand, a strategy to rank the neighboring elements and select the best one among them is required. We adopt a simple even effective criterion mostly based on global geometric characteristics of the subdomains being merged. In the following section, we provide the base definition of this suitability indicator and some variants to better guide the selection process.

To end up this section, it is key to highlight that the proposed stabilization strategy only affects the spatial discretization of the problem and is carried out in the preprocessing stage. The DG formulation of the problem is not affected by the changes in the polynomial approximation of the solution, i.e., no additional terms need to be evaluated and the burden of computing the different operators is the same as for the non stabilized scheme. Furthermore, this strategy is transparent to the temporary integrator. So the methodology we propose can be used to solve any problem using DG formulation and with any explicit temporal integrator.

3.5.3. Detailed description of the stabilization algorithm

In this section, we go through the different stages of the process leading to stable discretizations in cgDG and dig into the details of its implementation. After introducing the notation that let us describe the procedures in a general setting, we explain both the way of identifying unstable subdomains and how to handle them to arrive to a stable discretization.

3.5.3.1. Notation

Prior to going through the specific steps of the reassignment algorithm we need to generalize and extend the notation we introduce in the previous section. Let us consider two adjacent elements i and j that intersect the physical domain Ω leading to $\Omega_i = K_i \cap \Omega$ and $\Omega_j = K_j \cap \Omega$ where K_i and K_j are the domains of the whole elements. We represent as α_{ij} the relative area of the combined domain $\Omega_i \cup \Omega_j$ with respect to K_j. This corresponds to the extension of the element j over Ω_i. Note

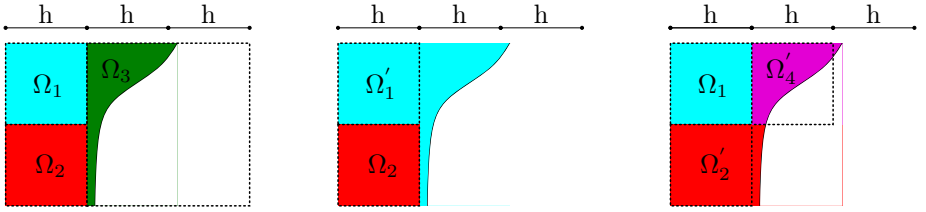
that the subscripts identify the physical subdomains to combine and the element we compare to is that of the last index.

We also need to introduce some notation regarding geometric quantities that appear in the suitability indicator. The centroids of the domains Ω_i and Ω_j are represented as $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}_j$, respectively. Finally, we identify as d_{ij} the distance between the centroids of the domains Ω_i and Ω_j, which formulates as $d_{ij} = \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|$.

3.5.3.2. Labeling unstable subdomains

The first stage in the algorithm used to reassign unstable domains addresses the selection of unstable subdomains. First, we identify the regions that limit the stability of the time integration scheme by computing the relative areas of all isolated regions within the boundary elements of the mesh and comparing them to α_{th}. Next, the mesh is checked to verify that all selected regions belong to elements that are not greater than their neighbors. The elements that do not meet this condition are refined and the resulting elements are studied to identify new sources of instability. By the end of this procedure, we identify a total of n_{ue} unstable subdomains.

The refinement criterion presented in this section prevents from extending over regions even greater than the size of the element, which may lead to polynomial instabilities associated to extrapolation. Alternatively, mesh refinement may be understood as a way of splitting the original unstable regions into smaller ones. With this in mind, the strategy lets us perform a better reassignment of the unstable region, linking each subregion to the most suitable element independently. We can even find refined substitutes of the original elements that are inherently stable without modifying the time integration scheme. In Fig. 3.19, we consider an initial unstable discretization (Fig. 3.19a) and compare the outputs of the reassignment algorithm when it is directly applied (Fig. 3.19b) and when the mesh is refined in advance (Fig. 3.19c). The figures show how this last approach, the one adopted in this thesis, leads to a more effective rearrangement of integration subdomains and to minor extrapolations. It is worth noting that the refinement criterion does not affect the stability of the base mesh since the stable time step is governed by the finest discretization.



(a) *Non-stabilized discretization (not used)*

(b) *Clumsy stabilized discretization (not used)*

(c) *Refined stabilized discretization (used)*

Figure 3.19: *Comparison of different spatial discretizations*

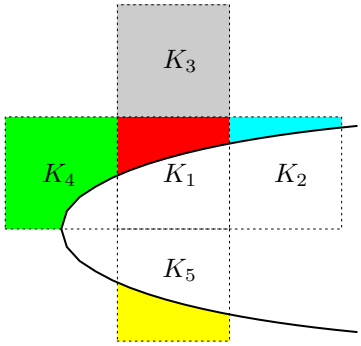
3.5.3.3. Rearranging element subdomains

Once the final mesh is obtained, we loop through the unstable regions in increasing order of relative area, namely $\alpha_e \leq \alpha_{e+1} \; \forall \, e \in \{1, 2, ..., n_{ue}\}$. Let us consider a general unstable region Ω_i. The first step is to identify the neighbouring elements, which will be those that share an edge with the element i. This operation is trivial given the Cartesian structure of the mesh in cgFEM. Next, we explore the regions within the neighbouring elements and discard those that are not adjacent to Ω_i. We obtain the preliminary set of candidates $\mathcal{N}_p$ that contains the elements that may be extended. In the following stage, the combined relative area resulting from each of the possible reallocations is computed, namely $\{\alpha_{ij} \; \forall \; j \in \mathcal{N}_p\}$. The final set of candidates $\mathcal{N}$ is computed removing from $\mathcal{N}_p$ the elements whose combined relative area does not met the condition $\alpha_{ij} \geq \alpha_{th}$. If $\mathcal{N}$ is an empty set, the iteration ends and we move on to the next subdomain without performing any change on the discretization. If some elements remain, the centroid of all regions is evaluated $\{\bar{\mathbf{x}}_k \; \forall \; k \in \{i, \mathcal{N}\}\}$ and the distance with respect to the centroid of the unstable region is computed $\{d_{ij} \; \forall \, j \in \mathcal{N}\}$. Finally, the unstable region is reassigned to the element that presents the minimum distance, $\min\left(\{d_{ij} \; \forall \, j \in \mathcal{N}\}\right)$. If the element chosen for extension was a stable element, the region is marked as resolved. If the final combination involves two initially unstable subdomains, both are removed from the set of unstable subdomains. The process continues until there are no subdomains classified as unstable.

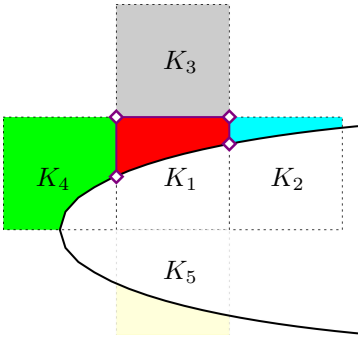
The performance of this procedure is illustrated in Fig. 3.20. The labeling stage rises Ω_1 as unstable subdomain and this example targets to solve it. As shown in Fig. 3.20a, the neighborhood of K_1 encompasses elements K_2, K_3, K_4 and K_5. However, K_5 is discarded in advance since Ω_5 does not meet Ω_1 at any point. The set of candidates $\mathcal{N}_p$ for Ω_1 is composed by the three elements represented in solid colors in Fig. 3.20b. Next, the relative area of all possible reallocations are computed in Fig. 3.20c. The support of the shape functions associated to element K_2 is too insufficient even after extending them over Ω_1. The small value of α_{12} prevents K_2 to be included in $\mathcal{N}$. The distances d_{14} and d_{13} depicted in Fig. 3.20d allow to rank the final set of candidates. The proximity criterion leads to choose K_3 for extension over Ω_1. Finally, K_1 falls from the computational mesh and Ω_1 is reallocated to K_3 meaning that the support of $\mathbf{N}_3$ becomes Ω_3'. The result of this process is shown in Fig. 3.20e.

Note that, unstable neighboring elements are included as candidates in $\mathcal{N}_p$. If, at the time of evaluating stability, the support resulting from merging the domains causes the initial discretization to become stable, it will make it likely to be extended. In this way, a smaller number of DOFs is eliminated and the size of the extensions is contained. By traversing the target domains in ascending order of relative area, we foster these types of assignments to occur.

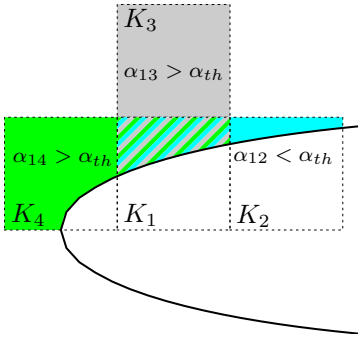
Configurations may appear for which not all unstable regions are resolved by the end of the loop. This is the case shown in Fig. 3.21a and Fig. 3.21b. When trying to reassign the domain encompassed in element K_1, the extension of the K_2 and K_3 elements is checked but in both cases, the resulting discretization is identified as



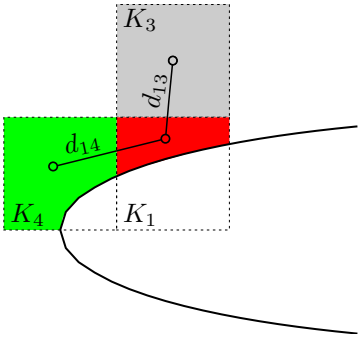
(a) *Unstable element under consideration and neighbourhood*



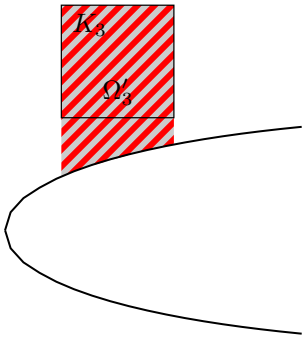
(b) *Filtering by continuity*



(c) *Filtering by expected stability*

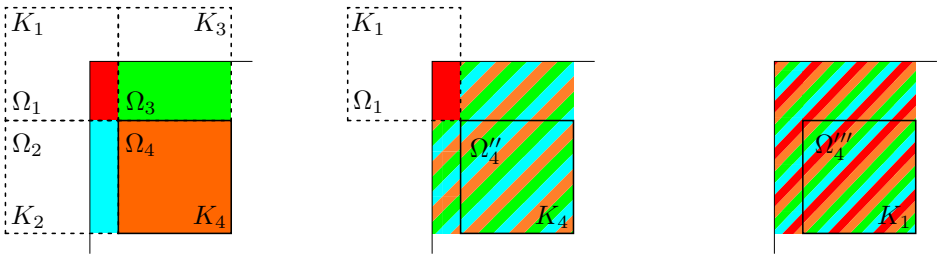


(d) *Geometric centers and distances*



(e) *Final domain assignment*

Figure 3.20: *Procedure to reassign unstable regions.*



(a) *Initial assignment* ***(b)*** *Assignment after first loop* ***(c)*** *Assignment after second loop*

Figure 3.21: *Example of two passes through loop to solve an unstable region.*

unstable since $\alpha_{1j} < \alpha_{th}$ with $j = 2, 3$. If we study elements K_2 and K_3 at the end of the loop, they will both have been removed from the mesh and their domains have been rearranged and now belong to element K_4. If we start over the loop taking into account the reallocation, a new study for the physical domain in K_1 leads to a valid reassignment. The domain could be reallocated to the element K_4, since it fulfils both the continuity conditions and the stability condition $\alpha_{1234} \geq \alpha_{th}$. Therefore, the domain reassignment process is performed iteratively until all target regions are solved.

3.5.3.3.1. Tuning suitability indicator

When selecting the element to be extended, we may be interested in considering other characteristics beyond the proximity between domains. As an example, level of stability, orientation or complexity of the assembled domain may be important features we must control. These aspects can be implicitly taken into account for the selection as a series of factors f_c that multiply the distance d_{ij} obtaining:

$$\bar{d}_{ij} = \left(\prod_{k=1}^{n_f} f_k \right) d_{ij} \tag{3.37}$$

where n_f refers to the total number of modifiers taken into account.

In the current implementation we consider the following factors:

- f_1: penalizes those elements with a relative area close to the threshold value. It gives an advantage to those elements with better characteristics form the stability point of view. This factor fits to the expression:

$$f_1 = 2 \left(\tanh \left(A_1 \ (\alpha_j - \alpha_{th}) \right) + 1 \right)^{-1} \tag{3.38}$$

  being A_1 the parameter that controls the range of values that are penalised. We recommend to set $A_1 = 10$.

- f_2: penalizes those elements with an unfavourable distribution of the physical domain. It reduces the appearance of very slender integration domains. It is formulated as:

$$f_2 = \prod_{k=\{x,y\}} 2\left(\tanh\left(A_2\ \left(\frac{12}{h^4} I_{ij}^k - \left(\alpha_{th}\right)^3\right)\right) + 1\right)^{-1} \tag{3.39}$$

  where I_{ij}^k represents the moment of inertia of the merged region with respect to the k-axis that pass through its centroid and A_2 is a number that limits the range of values that are penalised. We recommend to set $A_2 = 10$.

- f_3: penalizes those elements that have already been extended. It indirectly reduces the possibility of integration domains with internal corners. It is computed as:

$$f_3 = \sum_{s=0}^{n_t} 2^{-s} \tag{3.40}$$

  where n_t represents number of times an element has been previously selected for extension.

3.6. Numerical results

3.6.1. Verification

This first results section presents the numerical studies conducted to understand the general behavior of cgDG and to evaluate the performance of the proposed stabilization strategy. The test problems are not tied to specific applications but have been carefully designed to isolate and investigate particular aspects of the methodology.

First, Section 3.6.1.1 presents convergence analyses performed on the method without stabilization. Next, Section 3.6.1.2 quantifies the impact of the subdomain reassignment technique introduced in Section 3.5 on the stability of the time integration scheme. Finally, Section 3.6.1.3 investigates the effects of eliminating unstable degrees of freedom on solution accuracy and examines how p-refinement techniques can be employed to mitigate this impact.

3.6.1.1. Convergence analysis

In [149], Peterson proved that the original DG method for linear hyperbolic problems converges to the exact solution at a rate of $\mathcal{O}(h^{p+1/2})$, where h denotes the

characteristic length of the discretization and p is the degree of the polynomial approximation. Under certain regularity and compatibility assumptions, the convergence rate improves to the optimal $\mathcal{O}(h^{p+1})$ as shown by LeSaint and Raviart [150] for Cartesian grids and by Richter [151] for some structured non-Cartesian grids. iven the Cartesian structure of the mesh used in cgDG, a convergence rate of $\mathcal{O}(h^{p+1})$ is therefore expected.

The setup of the numerical test corresponds to a square domain $[-\lambda, +\lambda]\times[-\lambda, +\lambda]$ with its sides oriented in the same direction as the Cartesian axes. We impose boundary conditions, initial field and source term in such a way that the analytical solution of the problem is given by:

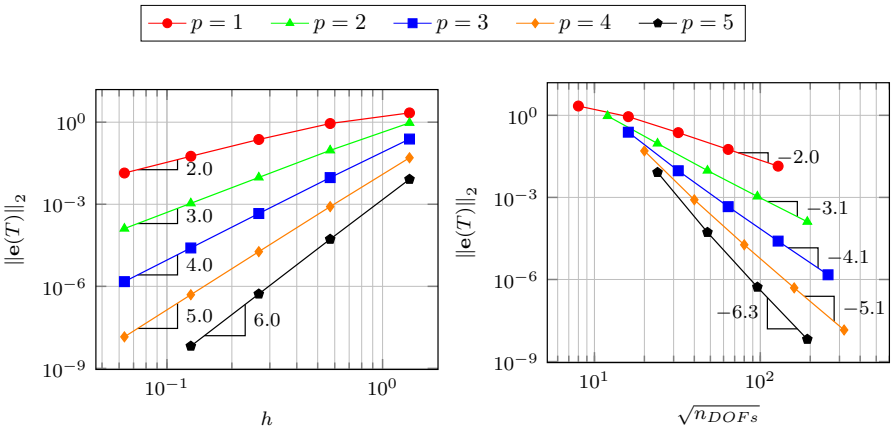
$$\mathbf{U} = \begin{Bmatrix} -\varepsilon \sin(\pi t)\cos(\pi x)\sin(\pi y) \\ \varepsilon \sin(\pi t)\sin(\pi x)\cos(\pi y) \\ -\mu \cos(\pi t)\cos(\pi x)\cos(\pi y) \end{Bmatrix} \tag{3.41}$$

A set of increasingly refined meshes is defined so that the intersection points of the boundary elements with the geometry are at the midpoint of the edge. The aim of this geometric constraint is to get the unfitted character of the cgDG method while avoiding the emergence of stability issues, which will be addressed in a separate study. Starting with the coarsest mesh, each finer mesh has twice as many elements per direction as the immediate coarser one. Regarding the temporal integration, we use a small enough time step so that the spatial discretization error dominate over any other.

Fig. 3.22a shows the errors of the cgDG solution field in the L_2-norm computed at time $t = T$, $\|\mathbf{e}(T)\|_2$, with respect to the element size, h. The slope of the curves in the asymptotic region equals to $p+1$ which correspond to the optimal convergence rate of DG when using Cartesian grids. This result puts the proposed method in advantage with respect to standard FE methods, that exhibit lower convergence rates. Fig. 3.22b shows the same results versus the number of DOFs. It is worth noting that, to obtain a certain level of error, interpolations of a higher degree require a lower number of DOFs and, for this problem, a lower computational cost, due to the smoothness of the analytical solution. Furthermore, even for the same number of DOFs, using high-degree polynomials is preferable as it leads to a reduction in the number of elements and, therefore, the amount of numerical fluxes that need to be evaluated. These results support the use of high-degree polynomial approximations to analyse problems involving wave propagation.

3.6.1.2. Impact of extended elements on the method's accuracy

Next analysis pursues two goals: quantifying the effect of the intersected elements on the stability of the method and evaluating the effectiveness of the domain extension strategy as a stabilization procedure. For this test, we consider a TE^z electromagnetic



(a) *Error with respect to the element size* ***(b)*** *Error with respect to the number of DOFs*

Figure 3.22: *Convergence of the method for an analytical problem solved in a square domain*

plane wave that propagates in the positive direction of the x-axis through an infinite domain. The electric components of the wave are computed as:

$$\mathbf{E} = \mathbf{p}\cos\left(\omega t - \boldsymbol{\kappa}\mathbf{x}\right), \quad \boldsymbol{\kappa} = \omega \begin{Bmatrix} \cos\phi \\ \sin\phi \\ 0 \end{Bmatrix}, \quad \mathbf{p} = E_\phi \begin{Bmatrix} -\sin\phi \\ \cos\phi \\ 0 \end{Bmatrix} \tag{3.42}$$

with:

$$E_\phi = 1 \quad \phi = 0 \tag{3.43}$$

The infinite domain is modelled as a rectangular domain with periodic boundary conditions on the boundary. Given an initial distribution that satisfies (3.42), the system is advanced in time. With this configuration, we want to evaluate the stability of the method regardless of the influence of the boundary conditions. In the test that we propose, the fluxes through the boundary only depend on the numerical solution computed in the previous step. We run the same problem iteratively following a bisection method until we get the maximum time step for which the L_2-norm of the solution does not grow in time, hereinafter Δt_{st}. This result is an approximation of the time step that makes the spectral radius of the method equal to one.

Fig. 3.23 gathers the set of base grids and computational domains considered in this test. All meshes are Cartesian grids made up of 8×8 elements and the x_1-dimension of all computational domains equals the wavelength. However, the element sizes and the x_2-dimension of the computational domains vary so that $\alpha = \{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 0.99\}$ for the outer layer of elements on the left and on the right.

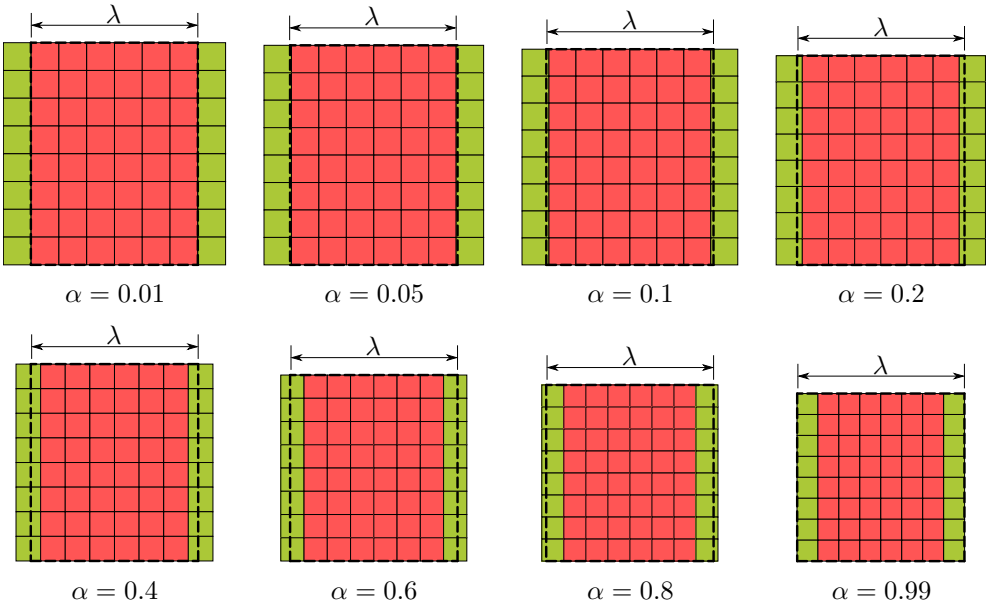
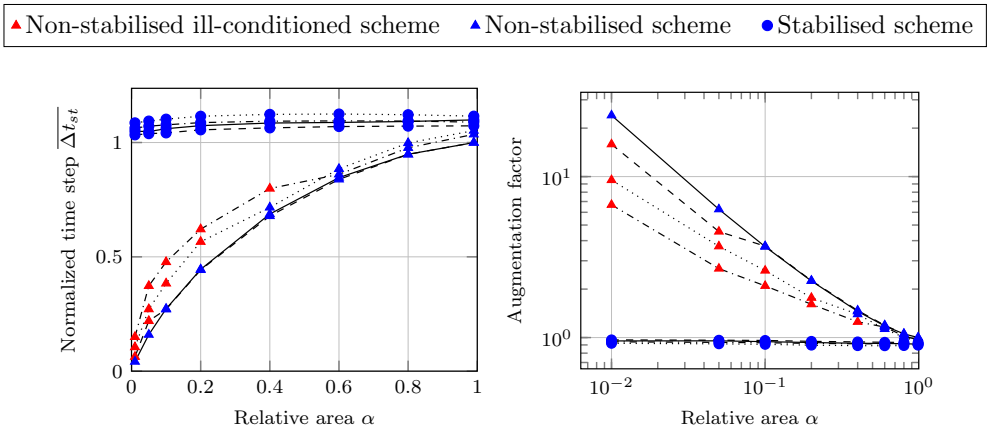


Figure 3.23: *Set of grids and computational domains used to test the stability of the method. The dashed lines represent the boundary of the computational domains. Elements in green are intersected elements while elements in red are internal elements.*

For each discretization, we considered two different scenarios. Firstly, all intersected and internal elements are active in the analysis mesh. Secondly, we apply the stabilization strategy before starting the integration loop. This second scenario uses a value of α_{th} big enough so that all intersected elements will be classified as unstable and the stabilization procedure will be applied to them. In the final discretization all intersected elements of the initial mesh disappear since their domains are reassigned to the internal neighbours and their DOFs are also removed from the system of equations. We consider the interpolation degrees $p = \{2, 4, 6, 8\}$. In order to facilitate the understanding of the results, Δt_{st} is normalized with respect to the maximum stable time step calculated for an equivalent discretization without cut elements, namely $\overline{\Delta t}_{st}$.

Fig. 3.24a shows the evolution of $\overline{\Delta t}_{st}$ with respect to α. For the non-stabilized scheme we observe that $\overline{\Delta t}_{st}$ drops with α, tending to 0 as α approaches this value. The slope of the curves is more pronounced as we move to smaller values of α. If we focus on the stabilized discretization, the stable time step remains almost constant around one. The results show that the stability of the stabilized method is governed by the base mesh and is independent of the cut patterns that appear in the intersected elements.

The results also show a common problem of the FDM: when the analysis mesh contains any element that posses very little support, the system becomes ill-conditioned.



(a) *Evolution of the stable time step* ***(b)*** *Evolution of the augmentation factor*

Figure 3.24: *Dependence of stable step size for the marching algorithm with respect to the relative ratio for both the standard and that stabilised discretizations. Each line represents a different interpolation degree p.*

As the degree of interpolation increases, the conditioning worsens and the domain necessary to ensure a certain precision of the solution increases. In Fig. 3.24a, the results obtained for discretizations that exhibit this behavior have been represented in red. In these cases, the vector of unknowns has been calculated using the Moore-Penrose pseudoinverse [152]. It is important to point out that these situations only appear when non-stabilized discretizations are used. The stabilization strategy eliminates the polynomial approximations with little support whose DOFs suffer form dependency problems. Therefore, the proposed stabilization strategy not only improves the behavior of the method with regard to time integration, but also lets us compute FE models that, even using very small time steps, could not be calculated accurately due to ill-conditioning.

In Fig. 3.24b, the inverse of Fig. 3.24a is drawn. On this occasion, the y-axis represents the factor by which the number of necessary iterations of the time loop is multiplied in order to reach the end of the simulation. Given that in most practical examples the calculation time corresponding to the time loop represents mostly the entire analysis time and that the computational cost of each iteration is independent of the step, this factor also applies to the overall analysis length. We see that the negative effects of α from the point of view of time integration are significant for $\alpha \leq 0.4$. As an example, for a polynomial approximation of order $p = 3$, if the mesh contains an intersected element with $\alpha = 0.2$, this supposes that the simulation prolongs 2.26 times. Contrary, the stabilized scheme is not affected by the parameter α and, therefore, the simulation times will be the same regardless of how the geometry intersects the mesh. In this latter case, the simulation time will only depend on the target level of accuracy, which will define the size of the elements and the approximation degree.

The same test has been carried out for a range of different element sizes. In all cases, the method behaves as in case described above.

3.6.1.3. The impact of extended elements on method accuracy

Extending the support of the interpolation polynomials implies the reduction in the richness of the solution since the number of DOFs per wavelength reduces. To counteract the expected loss of accuracy, in Section 3.5 we proposed to increase the degree of the polynomial approximation used on the extended elements. The p-refinement must be done in such a way that the stabilized scheme keeps the magnitude of the discretization error. To calculate the expected errors, a characteristic length of the interpolation support domain is considered. We define the approximation degree on element i based on the characteristic length l_i^d, which is the size of the projection on the Cartesian axes of the integration domain associated to element i. The degree is assigned according to the expression:

$$p_i = \left\lceil (p_r + 1)\frac{\ln(h_r)}{\ln(l_i)} - 1 \right\rceil \tag{3.44}$$

where p_r and h_r are two parameters defined by the user that correspond to the degree and element size of a reference discretization, respectively. The previous expression is evaluated independently in each direction. Once the appropriate polynomial degrees for each direction are computed, we may use all this information to define the final discretization, leading to a directional p-refinement; or we may only consider the worst case and use the same degree of approximation in both directions.

The following test seeks to quantify the effect of the stabilization strategy on the global error and the performance of the proposed p-refinement strategies. We use a numerical test that consists of a squared computational domain through which a wave is propagated. The distribution of such wave corresponds to equation (3.41). The dimensions of the domain are modified to sample the space of α parameter. Two different meshes that differ in orientation are studied. The border of the bounding box used to obtain mesh number 1 is aligned with the Cartesian axes, while the bounding box of mesh number 2 is rotated 45°. The base meshes remain unchanged throughout the test and we interpolate nodal values using 4th-degree polynomials. Fig. 3.25 shows mesh number 1 and mesh number 2 for $\alpha = 0.25$.

In Fig. 3.26, the red line represents the error in L_2-norm of the solution for time $t = T$ for the non-refined, stabilized scheme while the blue line shows evolution of the same magnitude for the non-stabilized one. First of all, it should be noted that the higher error obtained for mesh number 2 is due to its elements being of a larger size. For both examples, we observe that the error for the stabilized scenario is always greater than that of the non-stabilized one. The difference increases as the relative area grows. The gap reaches a size of one order of magnitude. Furthermore, we see that the slope of the curve for the stabilized results is steeper for mesh 2 than for mesh 1. The different size of the extension from the point of view of linear distance

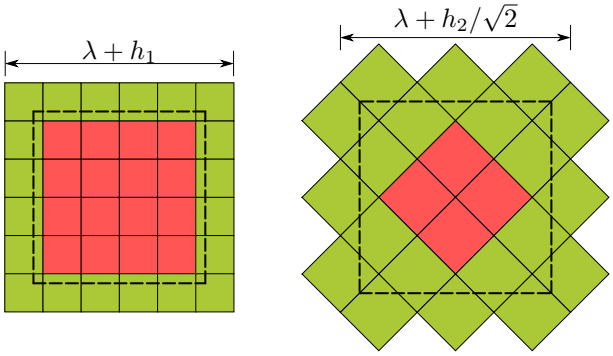
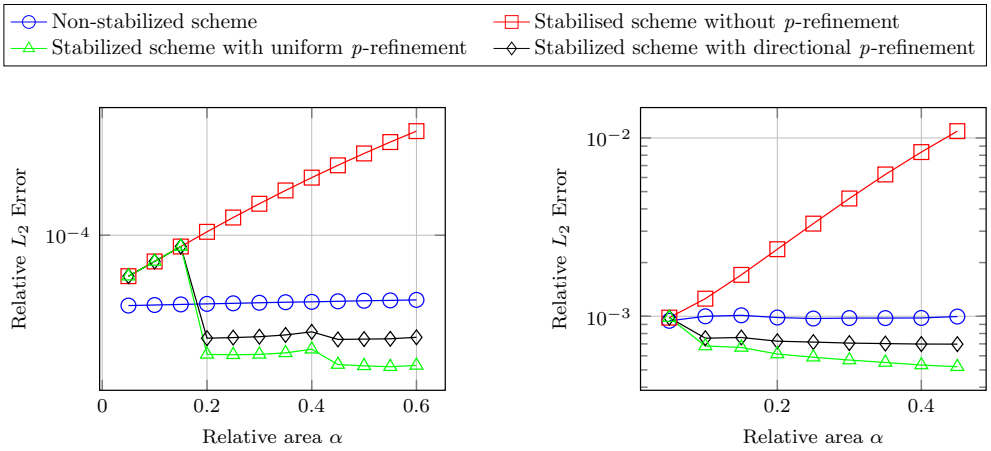


Figure 3.25: *Analysis meshes used to quantify the effect of the stabilization strategy on the discretization error. Mesh number 1 and 2 on the left and on the right, respectively.*

explains this different behavior. Although the area of the unstable domain is the same, if we focus on a single direction of space, the extensions for mesh number 2 are greater than for mesh number 1. This yields a reduction of the nodal density in this direction which leads to an increase in the global error.



(a) *Mesh number 1*

(b) *Mesh number 2*

Figure 3.26: *Effect of the extension strategy on error*

In contrast, we realize that using any p-refinement strategy driven by (3.44) we can keep the global error in the same order of magnitude as for the non-stabilized case, which shows that the refinement criterion is appropriate. As the extent of the extrapolation increases, the refinement criterion detects that the error distances from the target value and locally enriches the polynomial approximation to compensate this deviation. In Fig. 3.26, each degree increment translates into a drop in the curves going from left to right. Looking closely, we realize that mesh number 1 experienced

two increases in the degree of interpolation during the test while this number goes up to five for mesh number 2. The higher interpolation degrees used in the computations are 6 and 9, respectively.

Comparing the results for both p-refinement strategies, the drops in the error are more pronounced when the p-refined discretization is computed regardless of the extension direction. Thus, this strategy leads to a lower global error. However, we may not forget that the error target we pursue is that provided by the non-stabilized scheme. From this perspective, the directional p-refinement strategy reaches the desired level of accuracy. In addition, this second alternative brings two main advantages. First, it lets us save some computational effort since the number of DOFs of the final discretization is smaller. Second, the stability of the model is hardly affected since the polynomial order only raises in the direction that can be balanced with a larger support. For these reasons, we highly recommend this last strategy.

As previously mentioned, the stability depends on the degree of the polynomial approximation as well as the extent of its support. Increasing the degree of the extended elements to preserve a certain level of accuracy negatively affects the maximum stable time step. A balance has to be found between the target error and the number of steps of the time integrator. In practice, this dependence does not entail an important limitation since the worst cases from the point of view of the time integration do have little influence on the error and vice versa. Values of α_{th} around 0.3 have proven to be a good compromise solution for the set of problems studied so far.

3.6.2. Scattering by a PEC circular cylinder

The second problem to be studied is an infinite cylinder with the properties of a PEC which reflects a linearly polarized plane wave with either **H** or **E** vector fields aligned to the longitudinal direction of the cylinder. This is a reference problem in the field of scattering as an analytical expression is available for the exact radar cross section (RCS) or, more precisely, the scattered width (SW), the counterpart of the RCS in 2D. See [107] for further details. The 2D model considers a slice of the problem perpendicular to the longitudinal direction of the cylinder.

First, Section 3.6.2.1 evaluates the capability of cgDG to reproduce the analytical solution of the bistatic SW for this configuration. Next, Section 3.6.2.2 analyzes the influence of the wave propagation direction relative to the Cartesian mesh axes on the solution accuracy. Finally, Section 3.6.2.3 assesses the performance of the error compensation strategies discussed in Section 3.6.1.3, now applied in a context involving greater shape variability and increased solution complexity.

3.6.2.1. Overall performance of the method

In the first analysis, a cylinder with a diameter equal to 10λ is considered and solved both, with the TE^z and TM^z formulations, for an incident wave that propagates

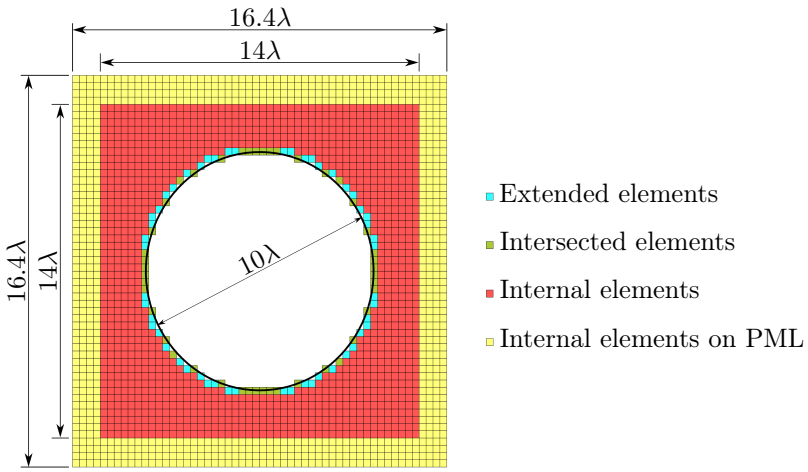


Figure 3.27: *Mesh and element locations*

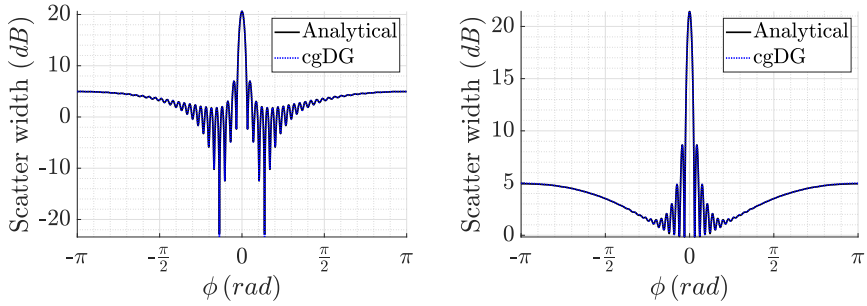
in the positive direction of the x-axis. The infinite domain is truncated to an extension of 2λ and the infinite boundary condition is imposed by using a Berenguer PML with an ABC on the boundary [110]. It is worth noting that the PML has been set following the advantageous procedure presented in Section 2.3.2 that leverages the Cartesian structure of the mesh.

The discretization consists of a uniform mesh of elements of 4th degree and a nodal density of 13 nodes per wavelength. Fig. 3.27 depicts the analysis mesh once the stabilization strategy has been applied.

Fig. 3.28 gathers the results obtained for the bi-static scattered width computed in both the TE^z and the TM^z modes. In these graphs ϕ represents the bi-static angle. In both cases, we consider that the time harmonic steady state is reached when the L_2-norm of the SW varies in less than 10^{-4} per unit in two consecutive cycles. Together with the numerical solution, the analytical solution is plotted. We see that the curves overlap for any ϕ. Numerically, the relative error in L_2-norm of the SW is 7.7×10^{-3} for the TE^z case and 4.53×10^{-3} for the TM^z case. These results confirm the accuracy of the proposed method.

3.6.2.2. The influence of propagation direction on method accuracy

In the following test the influence of the direction φ in which the incident wave propagates on the overall solution is studied. The objective is to evaluate the influence of the apparent length of the element with respect to the wavelength on the accuracy of the simulation.



(a) *TE^z mode* ***(b)*** *TM^z mode*

Figure 3.28: *Bistatic scattered width for a* 10λ *PEC cylinder.*

$\varphi(^\circ)$	0	9	18	27	36	45
TE^z Error	77.0	62.8	51.9	39.1	35.1	35.1
TM^z Error	45.3	45.1	38.9	32.8	27.7	25.4

Table 3.2: *Relative error* $\left(\times 10^{-4}\right)$ *in* L_2*-norm for the scattered width*

Changing the incidence direction leads to an equivalent change in the scattered field. We use this to evaluate if there is any propagation direction in the Cartesian mesh that drives to higher or lower accuracy. We start from the configuration of the previous case and carry out 5 analyses in which the angle of incidence is increased at a rate of $\Delta\varphi = 9^\circ$. Table Table 3.2 gathers the relative errors of the SW calculated according to both the TE^z and the TM^z mode for each of the angles in the L_2-norm. The values show a certain dependence to the orientation of the incident wave. However, the error of the SW always remains in the same order of magnitude, so we deduce that the method is not sensitive to the direction of propagation. In other words, the information equally propagates in all directions regardless of the Cartesian structure of the mesh. The small variations that we observe are probably caused by the influence of the terms of higher degree presented in the nodal interpolation.

3.6.2.3. Comparison of *p*-refinement strategies for accuracy on extended elements

The results shown so far in this subsection rely on a uniform discretization both in terms of element size and approximation degree. The final test of this set focuses on the effects of the stabilization strategy in conjunction with the p-refinement strategy. Unlike the squared geometry studied in Section 3.6.1, the circular boundary generates a great variety of cut patterns with different shapes and relative areas. In this example we reduce the domain of propagation up to 1.5λ and simulate considering: uniform

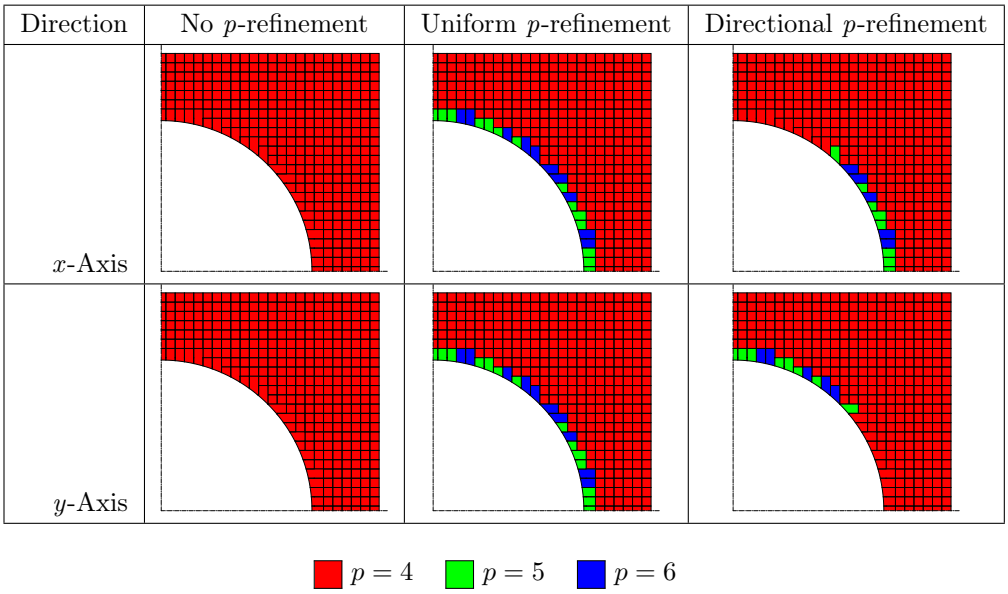


Table 3.3: *Comparison of the spatial discretizations for the PEC cylinder problem depending on the p-refinement strategy used.*

p, uniform p-refined and directional p-refined meshes; and the stability thresholds $\alpha_{th} = \{0.2, 0.4, 0.6, 0.8\}$. Table 3.3 shows a quarter of the mesh and the elemental interpolation degrees in each direction for $\alpha_{th} = 0.8$. The relative errors of the SW in the L_2-norm are gathered in Table Table 3.4. The results confirm that increasing the stability threshold has a negative effect on the accuracy as far as no compensation is considered. Regarding the performance of the p-refinement strategies, we do not find any difference in terms of error reduction while the directional p-refinement strategy lets save some computational cost. Therefore, we recommend the use of this approach.

α_{th}	0.2	0.4	0.6	0.8
No p-refinement	13.2	86.6	114.5	241.3
Uniform p-refinement	7.6	8.9	11.5	13.6
Directional p-refinement	7.6	9.0	11.0	13.8

Table 3.4: *Relative error* $(\times 10^{-4})$ *in* L_2*-norm for the SW.*

3.6.3. Scattering by a PEC NACA0012 aerofoil

In this section the scattering of a plane electromagnetic wave that falls out a NACA0012 profile modelled as a PEC is studied. The NACA0012 profile is a symmetrical shape that presents a singular point at the trailing edge. Special attention needs to be paid to the spatial discretization around this kind of points in order to accurately capture the distribution of the electromagnetic field.

The first analysis focuses on understanding the behavior of cgDG when applied to problems exhibiting singularities. The results presented in Section 3.6.3.1 illustrate how the location of the singular point within the mesh influences the accuracy of the solution. Furthermore, it is demonstrated that *hp*-refinement techniques effectively eliminate this dependency.

The second analysis addresses the case of a very high-frequency incident wave. Conventional computational methods often struggle in such scenarios as small geometric features —typically omitted or poorly captured in this schemes— have a significant impact on the solution. In contrast, the results reported in Section 3.6.3.2 highlight the outstanding capabilities of cgDG in accurately solving high-frequency problems.

3.6.3.1. The influence of geometry-mesh relative position on method accuracy

In the first place, the effect of the discretization around the singular point on the calculation of the SW will be studied. We compute the bistatic SW produced by a profile with a chord length equal to 10λ. The incident field corresponds to a plane TE^z wave whose trajectory forms an angle of $10°$ with respect to the chord line. The outer boundary of the computational domain is a square whose side length is equal to 13λ. The bounding box of the profile is centred inside this domain. We run a collection of 7 simulations which differ each other by the geometry orientation. The angle of the x-axis respect to the chord line φ_i varies from $\varphi_0 = 0°$ to $\varphi_6 = 45°$, with increments of $\Delta\varphi = 7.5°$. For each configuration, two analyses are performed: the first one employs a uniform mesh while the second one makes use of a *hp*-refined mesh. This last discretization uses top-level elements (smaller h) in the surroundings of the leading and trailing edges. In all cases, the interpolation degree is assigned to each element according to equation (3.44). Such approach means that refined elements are of lower order. Fig. 3.29 shows examples of both meshes for φ_6 as well as a detail of the elements around the singularity. This last representation makes it clear that multi-level jumps between adjacent elements are allowed. The setup completes deactivating the p-refinement strategy design to counteract of the extension and setting the stability limit to $\alpha_{th} = 0.4$.

Fig. 3.30 shows the SW obtained with the uniform mesh Fig. 3.30a and with the *hp*-refined mesh Fig. 3.30b. Both cases are compared with a reference solution computed on a verified FE software using a very fine high-order spatial discretization.

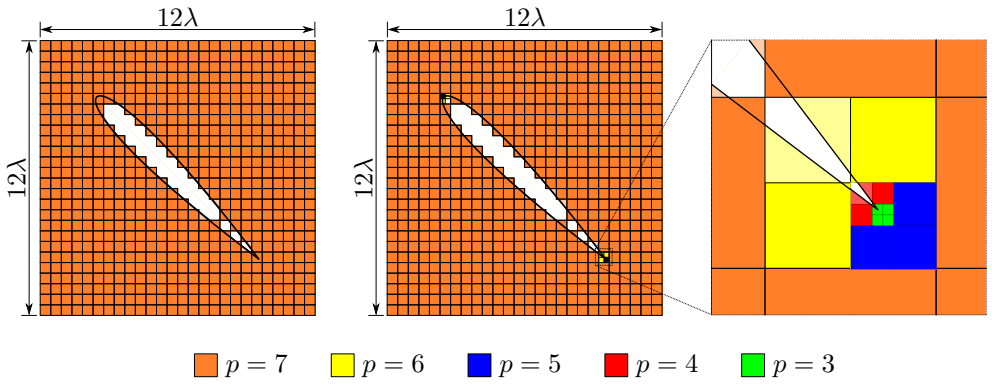
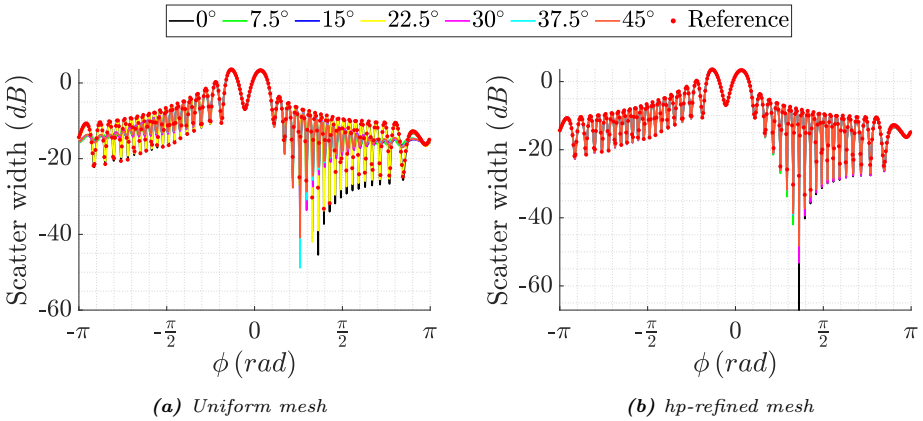


Figure 3.29: *Comparison of the meshes used to discretize the PEC NACA0012 profile: uniform mesh (left), hp-refined mesh (center) and detail of the refinement around the trailing edge (right).*



(a) *Uniform mesh* ***(b)*** *hp-refined mesh*

Figure 3.30: *Influence of the hp-refinement procedure based on geometry on the scattered width.*

We observe that the SW computed with the uniform mesh is much more sensitive to the position of the geometry than the one derived from the *hp*-adapted mesh.

In Fig. 3.30b all cgDG curves overlap independently of the bistatic angle. We only observe small discrepancies at those angles for which the reflected intensity is lower. Moreover, a very good agreement with the reference values is observed which highlights the higher accuracy of the *hp*-adapted scheme.

Contrary, Fig. 3.30a shows a far less consistent behavior. We may differentiate two set of curves within this analysis: $\{\varphi_1, \varphi_2, \varphi_4, \varphi_5, \varphi_6\}$ and $\{\varphi_0, \varphi_3\}$. The curves in the first set get away form the reference as the absolute value of the bistatic angle increases. This mismatch suggests the that these discretizations are insufficient to accurately approximate the solution. Furthermore, the discrepancies among the simulations are also remarkable specially for the greatest bistatic angles. This result

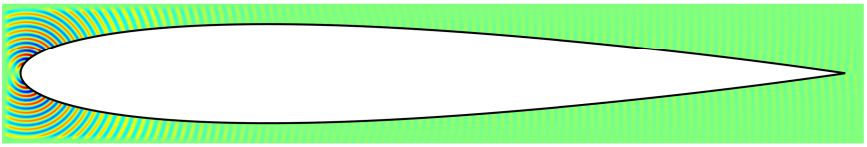
shows that cgDG is sensitive to geometry location with respect to the mesh meaning that how the complexity of the electromagnetic field distributes among the elements affects global performance of the method. The second set of curves achieves far better agreement with the reference solution and the simulations therein feature the trailing edge of the profile lying on a element edge. This result reveals that certain configurations may outperform the expected accuracy of the method since the resulting discretization is more likely to capture the solution singularities.

In conclusion, cgDG requires appropriate refinement strategies to consistently achieve accurate results. The effect of singularities induced by the domain shape can be counteracted by using geometry-based h-refining strategies and subsequently using a degree assignment strategy to result in an hp-adapted discretization. Chapter 5 addresses more sophisticated hp-refinement strategies based on error indicator that continuously, automatically adapt the entire discretization to the solution characteristics, including singularities. In addition, these last results confirm the conclusion reached in the previous example: the precision of the method is independent of how information propagates.

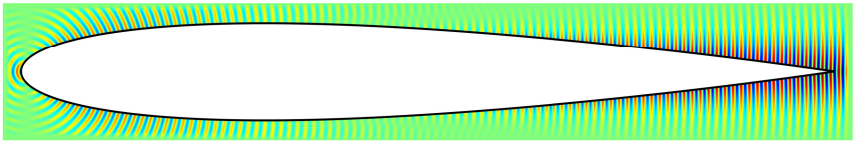
3.6.3.2. Method capabilities for high-frequency simulations

The following example analyses the same geometry but subject to a high frequency incident wave. This simulation supposes a great challenge as most numerical approaches suffer form numerical dispersion and/or dissipation which limits their scope of application. The wavelength is shortened so that the chord line becomes 100λ. The direction of incidence is also modified, and set to coincide with the direction of the chord. The geometry is embedded in a domain that extends 1λ in each direction beyond the geometry bounding box and is surrounded by a 1λ thick PML. Regarding discretization, we consider a nodal density of 17 nodes per wavelength and a interpolation degree equal to 7 for the elements of the uniform mesh. We apply a h-refinement strategy that split the elements of the uniform mesh up to two times. Regarding criterion (3.44), the resulting mesh contains elements of degree $p = \{7, 6, 5\}$. This brings the total of 363,602 nodes and 5,687 elements. These numbers are quite contained for a problem of these characteristics.

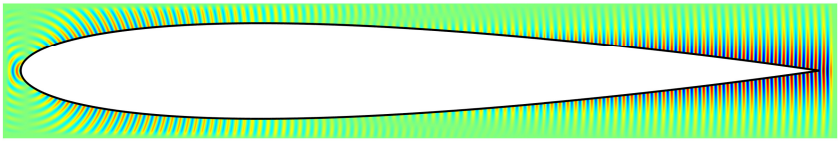
Fig. 3.31 shows the spatial distribution of the non-zero components of the scattered electromagnetic field once it is fully developed. Fig. 3.32 compares the SW computed with the cgDG implementation and a reference solution. The great match between these curves demonstrates the great performance of the proposed method when applied to high-frequency simulations.



(a) *Scattered E_1-field*



(b) *Scattered E_2-field*



(c) *Scattered H_3-field*

Figure 3.31: *Scattered electromagnetic field for the* 100λ *NACA profile*

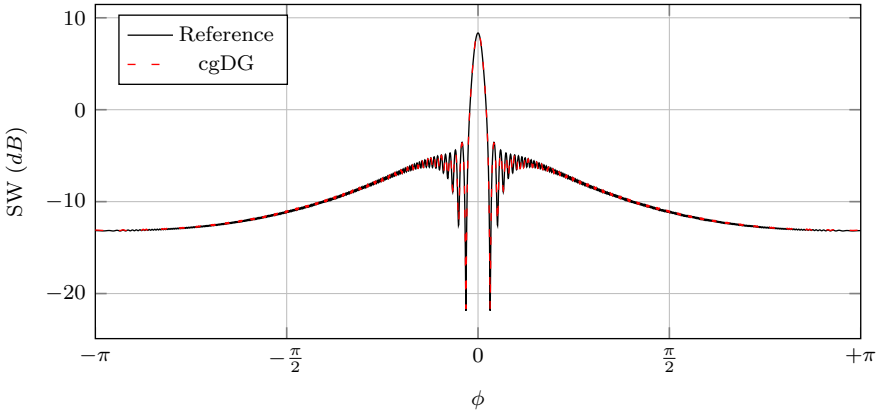


Figure 3.32: *Scattered width for the* 100λ *NACA profile*

3.6.4. Scattering by complex geometries

The simulations presented in this section aim to demonstrate the capabilities of cgDG for solving complex problems in the broadest sense. Two analyses with similar characteristics are conducted: in both cases, the scattering of high-frequency waves by bodies with intricate contours and small geometric features is investigated. The results reported in Sections 3.6.4.1 and 3.6.4.2 show the potential of cgDG in addressing problems of this nature, which are representative of real-world application scenarios.

3.6.4.1. Satellite section

The first problem concerns the scattering caused by the middle section of a PEC satellite. The maximum dimension of the satellite bounding box is $40\lambda/3$ and the region of interest extends 1.5λ beyond this bounding box. The computational domain is completed by a 1.5λ-thick PML that surrounds that region. In this case, the problem is formulated in TM^z mode and the plane incident wave travels in the direction of the x-axis. The spatial discretization used for this analysis corresponds to a uniform mesh formed by 1,582 elements of degree 7, which makes a total of 303,744 DOFs. It should be noted that, despite the fact that the satellite geometry presents details that are small compared to the element size, the cgFEM technology allows this boundary to be taken into account without the need to refine the elements over the boundary until their sizes match those of the geometric features. The spatial discretization is governed by the complexity of the solution but, unlike most of the fitted methods, it is independent of how the geometry is defined. In Fig. 3.33, the third component of the electric field is presented at the moment the steady state is reached. The figure shows how the method is able to capture the complexity of the field in the antenna cavity. In Fig. 3.34 the SW obtained with cgDG is compared with that computed by a contrasted FE software using an over-refined discretization. We see that, although we are using quite a coarse spatial discretization, the results fit the reference solution [9] quite well. Furthermore, it is worth mentioning that the SW obtained with the proposed method is perfectly symmetric. This results from the fact that the spatial discretization presents symmetry about the x-axis, just like the physical problem (domain and boundary conditions). Obtaining this sort of symmetries for the spatial discretization in cgDG is straightforward since the mesh only depends on the definition of the bounding box and the splitting process, both of which present symmetry about the Cartesian axes. In contrast, most mesh generation techniques used in fitted methods evolves to non-symmetric discretization regardless of the starting point.

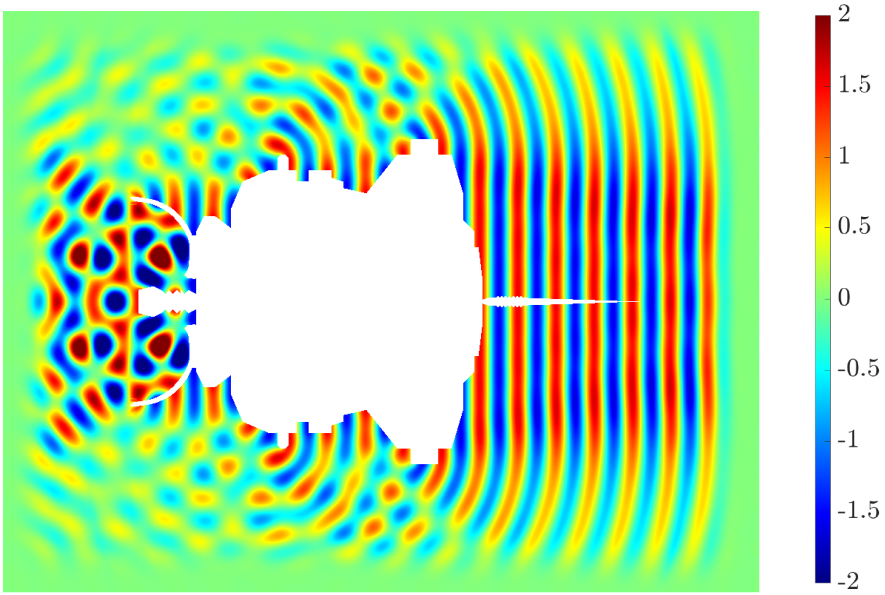


Figure 3.33: *H_3 scattered field computed for the satellite profile (click on the figure to show the animation, available when using Adobe Acrobat).*

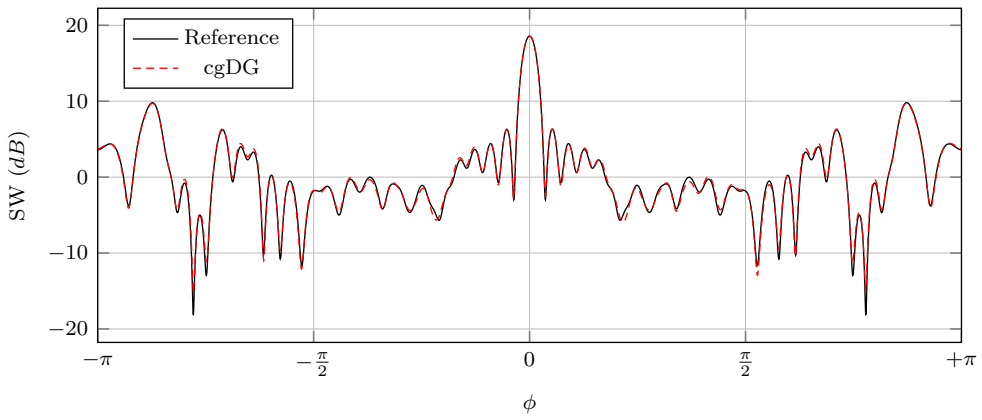


Figure 3.34: *Bistatic scattered width obtained for the satellite profile compared with a reference solution [9].*

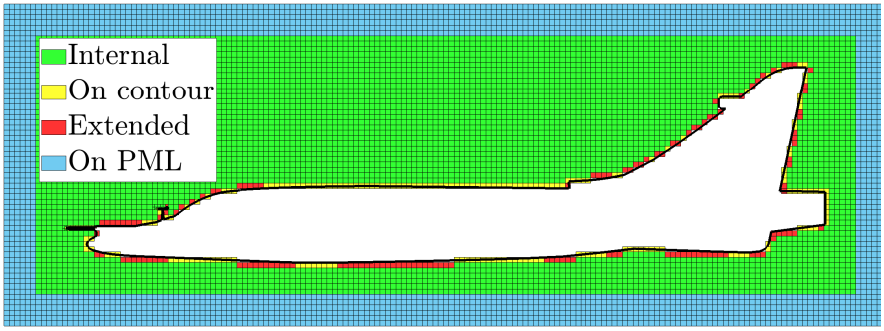
3.6.4.2. Airplane section

Next, the scattering problem for the middle longitudinal section of an airplane is considered. This boundary is particularly challenging from the point of view of the geometry. Small features compared to the global size the profile, such as the front antenna, originate resonance regions that have a great influence on the overall distribution of the electromagnetic field. In order to capture these local complexities of the solution we use an *hp*-adapted mesh with smaller elements in the environment of this geometric feature. Additionally, as these structures are slender, they give rise to elements with several isolated regions within the physical domain. The dissociation strategy is used on these elements.

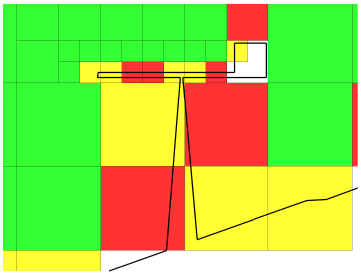
In the following, the setup of the airplane problem is depicted. The body is modelled as a PEC whose length is 50λ. The physical domain extends 2λ beyond the geometry bounding box in both directions with a 2λ-thick PML surrounding it. We consider the TM^z version of the problem and study the scattering a plane wave that propagates form left to right. As aforementioned, this analysis uses an *hp*-adapted mesh with smaller elements around the most relevant geometric details and a base discretization of 20 nodes per wavelength. Fig. 3.35 shows the complete mesh as well as a zoom of the antenna surroundings.

Note the presence of elements in the mesh with more than one disconnected regions. It is of interest to analyze how such elements evolve during the execution of the program. Consider an element K^i; at the initial stage, two disjoint regions K_1^i and K_2^i are identified within it, each of which is assigned a different set of nodes k_1^i and k_2^i, respectively. When evaluating the stability, it is detected that the discretization of the K_2^i region is unstable and the support domain is reassigned to the element K_1^{i+1} giving rise to K_2^{i+1}. Finally, the DOFs linked to k_2^i are removed from the set of DOFs and the discretization k_1^{i+1} is enriched through p-refinement to obtain k_2^{i+1} .

The third component of the electric field is shown in Fig. 3.36. Fig. 3.37 shows some details of the solution in the regions where it exhibits a higher complexity. At the overall level, we see how the method is able to capture the influence of geometric details. We observed an increase in the amplitude of the scattered field, especially in the lower part of the refuelling needle, the front part of the antenna placed in front of the cockpit and around in the electronic countermeasure (ECM) placed on the vertical stabilizer. However, when we go into detail, the polynomial approximation used is not able to correctly represent the very local complexity of the solution caused by the complexity of the boundary. This is clearly seen in the elements that contain the end of thin details of the geometry. This limitation does not affect the general performance of the method, as these are errors that affect locally and, thanks to the mesh refinement, do not pollute the global solution. Fig. 3.38 confirms this assertion. The SW obtained with the proposed method fits the reference solution, computed with a much finer fitted NEFEM mesh [9] and an in-house high order DG solver, really well, taken into account the total amount of 523,828 nodes considered in this simulation.



(a) *Overview*



(b) *Detail around the front antenna*

Figure 3.35: *Analysis mesh for the aircraft section.*

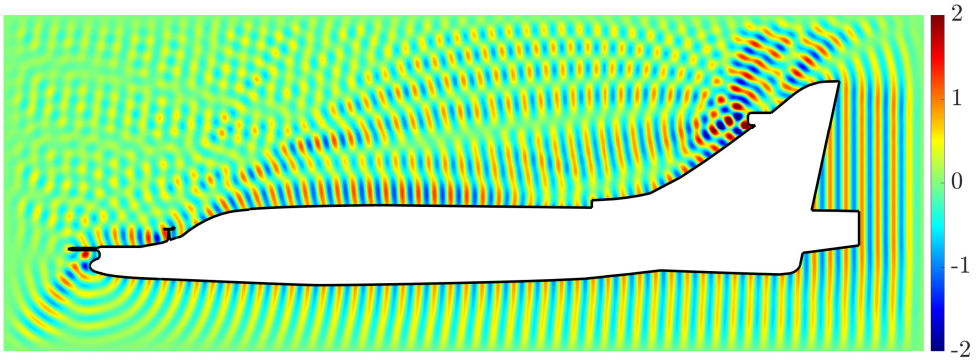
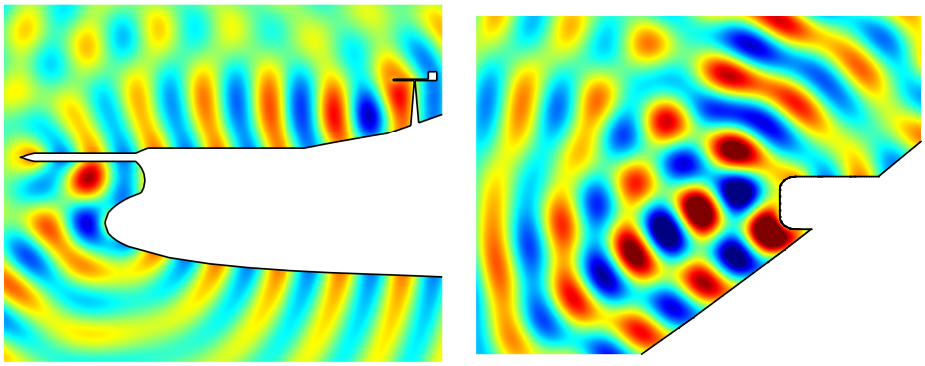


Figure 3.36: *H_3 scattered field computed for the aircraft section.*



(a) *Front part* ***(b)*** *ECM*

Figure 3.37: *Zoom of the H_3 field around geometrical details.*

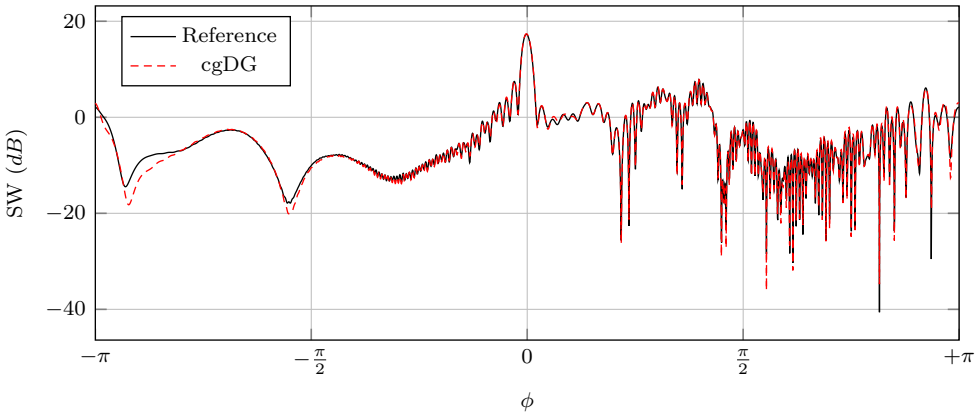


Figure 3.38: *Bistatic scattered width obtained for the aircraft profile.*

Chapter 4

The cgDG in the 3D case

Extending a simulation strategy from the 2D case to the 3D one is not trivial. Introducing the third dimension increases the complexity of the calculations and greatly expands the number of scenarios that need to be considered. This means that methodologies and algorithms that were sufficient for the two-dimensional case must be refined, modified, or even replaced by others that approach the same problem from a completely different perspective. This chapter presents the strategies we have developed so far to extend the cgDG methodology to the three-dimensional space.

The contributions presented address three essential aspects within the simulation process: the adaptation of the geometric model without losing information for subsequent use within the program, the robust, accurate, and efficient evaluation of volume and boundary integrals in intersected elements, and the stabilization of the time integration scheme considering the presence of degenerate intersected elements. These topics are discussed in Sections 4.1, 4.2 and 4.3, respectively. More specifically, Section 4.1 outlines the preprocessing strategy that will be applied to raw geometric models provided by CAD tools, which will represent the computational domain of the subsequent simulation. Section 4.2 presents three alternatives for evaluating integrals in intersected elements, highlighting their main differences. Section 4.3 introduces two variations of the stabilization methodology presented in Section 3.5 to adapt it to the three-dimensional case. In Sections 4.2 and 4.3, special attention is given to how the characteristics of the Cartesian mesh can be leveraged to effectively implement the different strategies. The chapter concludes with Section 4.4, which presents a series of numerical tests comparing the performance of the proposed strategies, as well as a final example in which the method is validated by solving a more general scattering problem.

4.1. Domain definition and geometry pre-processing

CAD codes have become the preferred design tool in the industry. A critical phase in the new product development cycle involves the designer progressively refining the product through iterative design. When dealing with parts that must fulfill an engineering function, the modifications introduced are aimed at optimizing the characteristics of the part for the function it must perform. CAE programs are the tools that allow analysts to predict the behavior of these prototypes quickly and economically. Traditionally, the analyst must preprocess the geometry provided by the designer before setting the parameters of the simulation that will evaluate the performance of new designs. This is a laborious process that results in a new definition of the geometry adapted to the specifications of the CAE program and the properties of the numerical method to be used. Typically, this task involves changing the format used to encode the model, redefining some of its parts to meet certain desired properties from the simulation's point of view, and simplifying areas of the model by eliminating geometric features that, in the analyst's judgment, will not have a significant influence on the simulation results but may pose a problem when constructing the numerical model. Efforts have been made to automate these **defeaturing** tasks. However, communication between designers and analysts is much more efficient when both can work on the same files.

The implementation of cgFEM in the 3D context, which we will refer to as FEAVox, has been approached considering that the model geometry will be provided by the output file of CAD software (the exchange format used is the one defined in the IGES standard **IGES** due to its wide acceptance and versatility). The goal has always been to maintain a definition of the geometry as close as possible to the original design. We are able to do this because, in the cgFEM methodology, the independence of the computational mesh from the geometry eliminates many of the limitations that required preprocessing CAD models. These complications are transferred to the more flexible integration phase.

For cgDG to be robust, we must ensure that we are able to accurately capture the relative position of the computational domain with respect to the elements of the Cartesian mesh. Additionally, it is desirable that each element can be studied in isolation. This would make it possible to parallelize calculations to reduce computation times and obtain redundant information that allows detecting anomalies within the workflow. This last consideration represents a significant paradigm shift compared to previous versions of FEAVox. In those versions, a global approach was used that imposed restrictions both when calculating intersections (a uniform mesh had to be considered with a cell size equal to the smallest element used in the discretization) and when locating the elements (information from all intersections had to be gathered

before classifying the elements). The shift to the local approach will only be possible if the geometric model meets the hypotheses detailed below.

In this section, we will present the methodologies we have developed to address the most common difficulties we have encountered when working with CAD geometry models in FEAVox. The tasks that have been addressed are:

1. determining the relative position of each element's entities (faces, edges, and vertices) with respect to the three-dimensional solid,
2. resolving ambiguous intersections that appear when intersecting the Cartesian mesh with periodic curves, periodic surfaces, and collapsed surfaces,
3. inferring a structure that reflects the continuity of the boundary defining the solid from a set of disconnected faces,
4. avoiding the erroneous classification of intersected elements as internal/external.

It is important to emphasize that the geometry remains unchanged at all times. The modifications applied only affect the mathematical model used to represent it.

4.1.0.1. Locating Cartesian elements with respect to the computational domain

In the field of computer-aided design, various strategies exist for mathematically representing 3D models. The most widely used techniques in commercial software are Constructive Solid Geometry, B-Representation, features-based modeling, parametric modeling, and implicit representation. Among them, the most commonly used system is B-Representation. This strategy consists of defining a solid through its surface. This is possible because the boundary of a closed and bounded set uniquely defines the set. The representation can be extended regardless of the cardinality of the system. Thus, the geometry of a solid is fully defined by a structure in which each level contains elements of a lower dimension than the elements of the higher level.

In Fig. 4.1, the structure that would allow a tetrahedron to be defined is shown. At the top level of the structure is the solid (Solid), which points to a series of shells (ShX), each of which has an associated boolean (ShBX) that determines whether the outward normal vector to the shell points in the outward direction of the solid. When both vectors coincide, the boolean will be true. Each shell is composed of a set of faces (FX) that are paired with booleans (FBX) to ensure that the normals of the faces forming the shell are consistent with each other. This consistency means that a point in space views all the faces from the same side: either with the outward direction toward it or in the opposite direction. A face is the result of trimming a surface (SX) with a set of closed loops (LX), where the first is the outer contour and the subsequent ones are holes. Each of these loops is formed by an ordered set of curves (CX), each paired with a boolean (CBX) indicating whether the start and end points of the curve are such that the loop is traversed in the correct direction. A false value would indicate that the curve should be traversed in the opposite direction.

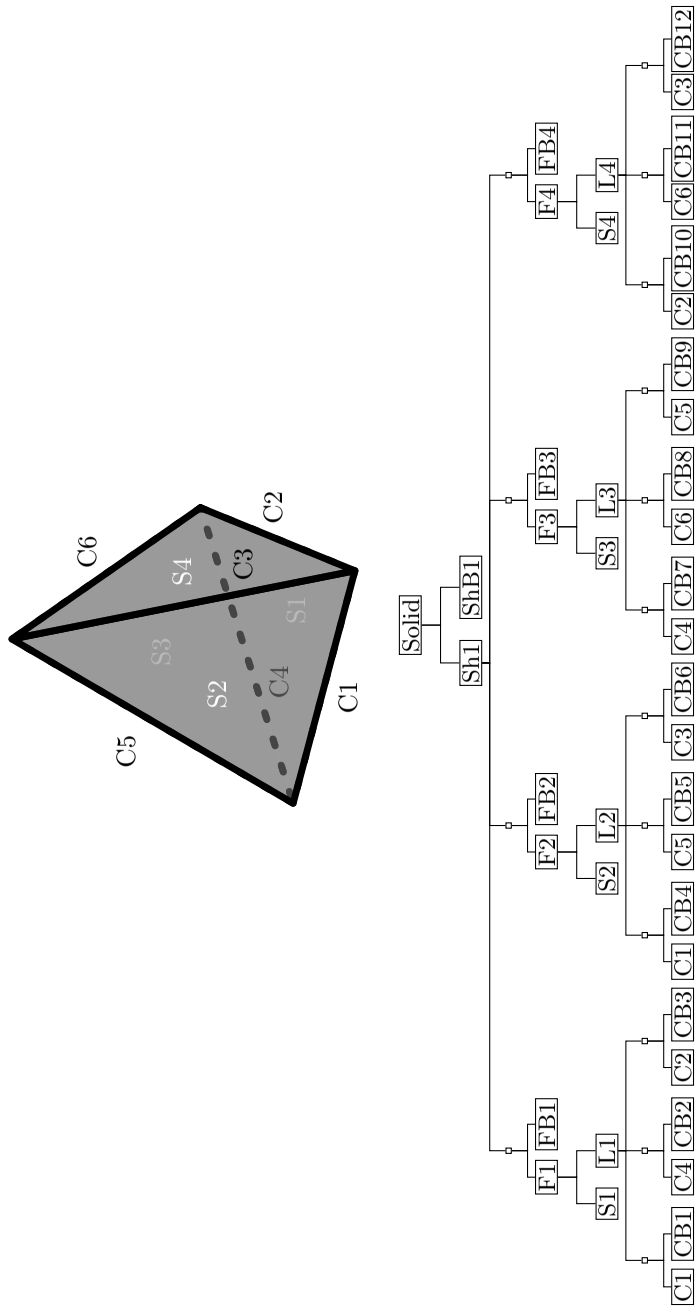


Figure 4.1: *B-Representation of a tetrahedron*

The solid representation is completed by associating a mapping with each geometric entity (surfaces, curves, and points) in the hierarchical structure, allowing for the transition from a parametric space in R^{d-l} to R^d. The dimension of the parametric space $d - l$ corresponds to the number of characteristic dimensions of each entity, which means that l will range from $l = 0$ for volumes to $l = 3$ for points.

For the representation to be valid, the set of the hierarchical structure and the parametric definitions must satisfy the following constraints:

R1 All points must be distinct.

R2 Two edges must intersect at a vertex or be disconnected.

R3 An edge must belong to an even number of faces.

R4 The intersection between two faces must be an edge. Otherwise, the faces will be disconnected.

In FEAVox, we will adopt a solid representation similar to B-Representation. The main advantage we obtain is the ability to locally evaluate the relative position with respect to the solid. With the parametric definition of the surfaces and traversing the solid's tree from bottom to top, we can uniquely determine the outward and inward directions to the solid at any point. This allows us, given the intersection of an element's edge with the solid's boundary, to determine the section that falls within the domain of interest by making a simple projection and evaluating the sign of the result. Using connectivity and hierarchical relationships, classification can be extended to non-intersected edges and the nodes and subdomains within the element. In Fig. 4.2, the element intersected by a spherical domain is shown, where the intersections and outward directions of the solid are depicted in magenta, and the parts of the edges and the nodes inside and outside the solid are shown in green and red, respectively.

There are cases where the normal information may lead to contradictory classifications, as seen in Fig. 4.3. These situations occur when multiple intersections coincide at the same point. The solution we will adopt is to slightly displace the edge so that the multiple intersection splits into several intersections, allowing us to uniquely classify each section of the edge. The original case, which will be the limit where the two intersections converge at the same point, will retain the same classification as the auxiliary problem with the displaced edge. In Fig. 4.3, the relative position of the element (solid line) is ambiguous if only the intersection information is used. By displacing the intersected edges, a new configuration is obtained (dotted line) where classification is feasible. The same issue arises when the intersection point is a tangency point. The strategy we proposed can also resolve this situation.

The constraints required by the B-Representation of a solid can be used at various stages of the geometry preprocessing phase to facilitate or validate calculations. As an example, we can use constraint R3 to detect open volumes: if face F is the only one containing edge E, we can be sure that no set of faces S containing F will be closed since S will have at least one opening at E.

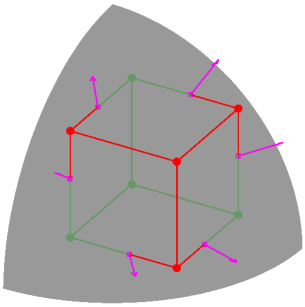


Figure 4.2: *Isolated element intersected by a sphere*

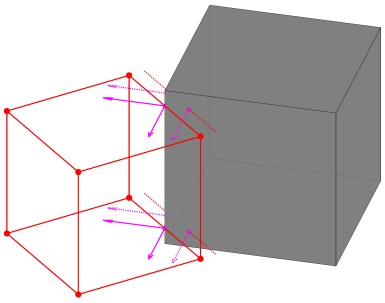


Figure 4.3: *Isolated element intersected by a corner leading to uncertain classification*

4.1.0.2. Ambiguous intersection points in curve/surface parametric spaces

In CAD modeling, it is common to encounter geometric entities whose mapping from physical space to parametric space is not injective. This is the case for closed curves, cylinders, spheres, etc. All these entities are characterized by having periodicity. The mapping from parametric to physical space curves the entity in such a way that the representation of at least one of the boundaries of the parametric space ends up coinciding in the physical space. This poses a problem when determining the portion of the geometric entity contained within an element.

Let's present an example for the case of curves. We start with the hypothesis that the curves will never be tangent to the boundary of the element. This implies that a curve $\mathbf{C}(t)$ will intersect the boundary of an element $\mathcal{K}$ a finite number of times. Additionally, we will be able to determine the relative position of the ends of the curve with respect to the domain of the element. Let us consider the union of the set of intersections $\mathcal{P}_1 = \mathbf{C}(t) \cap \partial\mathcal{K}$ and the set of curve ends that are not external to the element $\mathcal{P}_2 = \partial\mathbf{C}(t) \cap \mathcal{K}$, we will call $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ this set. Due to the continuity of $\mathbf{C}(t)$, the cardinality of $\mathcal{P}$ will always be even, and we will identify it as $2n$. If the mapping $\mathbf{C}(t)$ is bijective, the images of $\mathcal{P}$ in the parametric space of $\mathbf{C}(t)$ form a set $\mathcal{P}^{\text{st}} = \{t_i : t_i = \mathbf{C}^{-1}(\mathbf{x}_i) \; \forall \; \mathbf{x}_i \in \mathcal{P}\}$ that preserves the cardinality. If we order the elements of $\mathcal{P}^{\text{st}}$ to form the sequence $\{t_i : t_i < t_{i+1}\}$, we can obtain the segments of the curve contained within the element by considering that $\mathbf{C}(t) \in \mathcal{K} \leftrightarrow t \in \cup_{j=1}^{n} [t_{2j-1}, t_{2j}]$. The same result would be obtained if we define the sequence $\{t_i : t_i > t_{i+1}\}$ and the condition $\mathbf{C}(t) \in \mathcal{K} \leftrightarrow t \in \cup_{j=1}^{n} [t_{2j}, t_{2j-1}]$.

Remark. When the intersection is a tangency point, we can follow the same procedure if we include in $\mathcal{P}_1$ only those intersections $\mathbf{C}(t_i)$ that satisfy $\{\mathbf{C}(t_i^-), \mathbf{C}(t_i^+)\} \in \mathcal{K}$, where $t_i^{\pm} = \lim_{\delta \to 0} t_i \pm \delta$.

In Fig. 4.4, we observe a straight segment (curve 1), a periodic curve that describes a circumference (curve 2), and their intersections with the dihedral representing two

contiguous faces of an element and the edge joining them. Curve 1 is the general case. We encounter two intersections $P_{1,1}$ and $P_{1,2}$ that have a unique representation in parametric space. However, when studying curve 2, we see that $P_{2,2}$ has two possible images in parametric space. The periodicity of the curve implies that the mapping from physical to parametric space is not injective. If we proceed to determine which portion of the curve is contained within the element following the same procedure as for curve 1, we realize that there are two possibilities depending on the ordering: the range of parameter values $Sp_{2,1}$ or $Sp_{2,2}$. A priori, we cannot discern which segment is the internal one since both are equally valid.

From an implementation perspective, when calculating the intersection using an approximate resolution method like Newton-Raphson, we will only obtain one of the values, and the convergence to one parameter or the other will depend solely on the initial guess. In this scenario, we will define the portion of the curve inside the element unambiguously, but there are no guarantees that it is the correct range. In the case presented in Fig. 4.4, we could obtain a parameter value equal to 1 for $P_{2,2}$. Applying the procedure described above, we would arrive at the incorrect conclusion that $Sp_{2,2}$ is the range of parameter values that fall within the element.

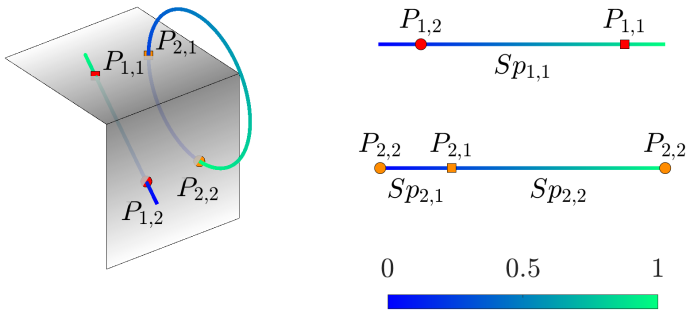


Figure 4.4: *Intersection of two curves with the faces of an element. On the right, the parametric space of both curves is represented, as well as the location of the intersection points in both spaces.*

To solve this problem, the geometry will be preprocessed: the periodic entities of the geometric model will be divided in such a way that all mappings are bijective. To do this, the direction in parametric space where the overlap occurs will be identified, and two independent entities will be generated, each of which will cover half of the original parametric space. In Fig. 4.5, the intersection of a cylindrical surface with a Cartesian system is shown. The original surface is periodic, and as a consequence, the intersection point $P_{0,1}$ can have two representations in parametric space. If we follow the procedure outlined above, the first step is to detect the boundaries of the parametric space where the overlap occurs. The representation in physical and parametric space has been marked in blue and corresponds to lines where the first parametric coordinate s_1 is constant. If we divide the original surface S_0 using a line with coordinate $s_1 = \bar{s}_1$, where $\bar{s}_1$ is a value within the definition interval of s_1, we will obtain two child surfaces S_1 and S_2. The mappings associated with these

two surfaces will be bijective, as the points in parametric space that shared the same projection in physical space will now belong to different surfaces.

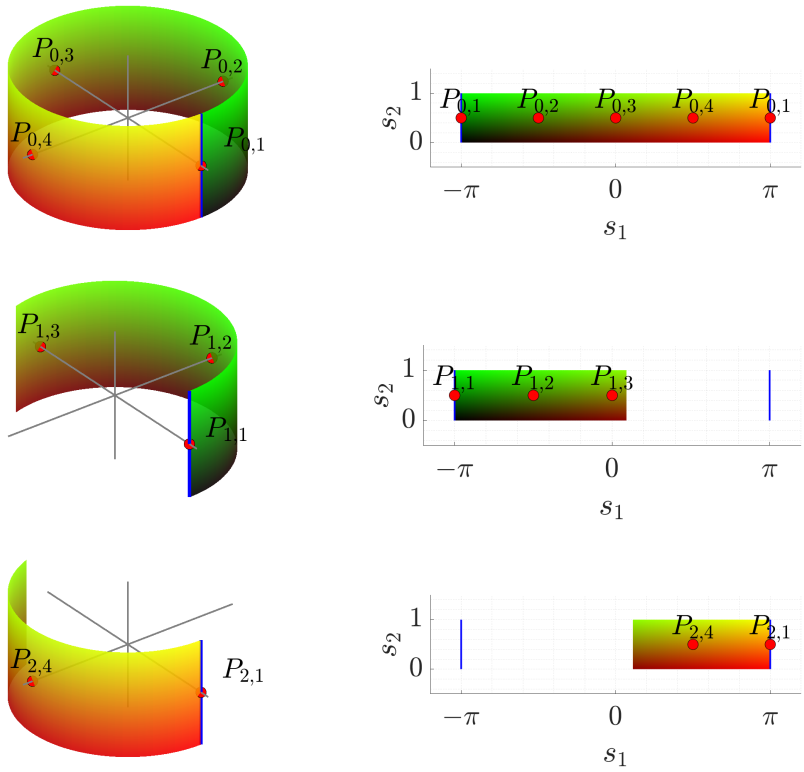


Figure 4.5: *Cartesian axes intersecting periodic surface and children of periodic surface*

It is important to highlight that preprocessing will be done within the workflow of the analysis software. If we try to preprocess the geometry within the CAD tool, it is very common that, even if we divide the periodic surfaces using some tool they provide, the entities describing the geometry will not be modified. Cutting tools usually modify the trimming loops associated with the faces or the endpoints of the edges, but not the actual definition of the surface and curves, respectively. Therefore, we would still encounter the same problem when performing the calculations. As an example, a semicircle can be a circle where only a portion of the parametric space has representation in physical space. From an implementation perspective, this modification offers no advantage since the intersection calculation is performed considering the basic geometric entities. Trimming is taken into account at a later stage to determine whether these intersections have a real image in the physical domain.

4.1.0.2.1. Collapsed points

A particular case of periodicity is the collapsed points. These are characterized by a region of the parametric space being projected onto a single point in the physical space. In these cases, the subdivision of the geometric entity is ineffective since the child entities would present the same problem as the predecessor. Instead, the intersections at collapsed points are post-processed, and each of them will split into two or more intersections with different coordinates in the parametric space but the same representation in the physical space.

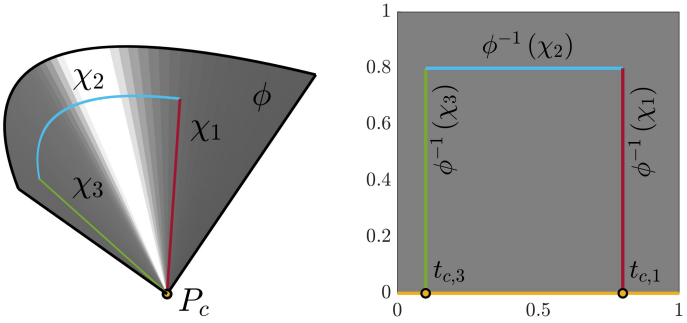
Our method for handling collapsed points is based on the hypothesis that these points will always correspond to a portion of the surface boundary. This is a reasonable hypothesis that is satisfied in all the cases we have studied. We have only found collapsed points in geometries with surfaces of revolution or sweeps, where a boundary of the parametric space becomes the vertex of the geometry. According to the B-Rep representation, for the collapsed point to have representation in the physical space, it must be contained within one of the trimming loops associated with the faces that use the surface. The loops that contain the point must necessarily pass through it, as it is the only way to delimit a region of the surface without leaving it and using a continuous function. Therefore, we will have curves that have their endpoint or origin at the collapsed point. Knowing this, we can decompose the edge-face intersection into a set of plane-curve intersections, all of them with the same spatial coordinates but to which we will assign different coordinates in the surface parametric space. The number of parametric representations for a collapsed point will depend on the number of edges that converge at it.

Let us consider the image in the physical space P_c of a collapsed point on the surface defined by the mapping $\boldsymbol{\phi} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. For a curve with an associated transformation $\boldsymbol{\chi}_i : \mathbb{R} \rightarrow \mathbb{R}^3$ that projects the parameter t_c into P_c, a version of the intersection with coordinates in the parametric space of the surface will be generated:

$$\mathbf{s}_{c,i} = \lim_{t \to t_c} \boldsymbol{\phi}^{-1} \left(\boldsymbol{\chi}_i \left(t \right) \right) \tag{4.1}$$

In conclusion, and as can be seen in Fig. 4.6, we are imposing that the projection of the trimming curve in the parametric space of the surface is continuous. When obtaining the portion of the face contained within an element, we must calculate the position of the intersection, taking into account the planes of the Cartesian mesh that converge at the element's edge. This case is resolved by applying the same principle. The difference lies in the fact that now the curve is implicitly defined using the fixed coordinate of the Cartesian plane x_{cp} and the surface definition:

$$\boldsymbol{\chi}_i \left(t \right) = \boldsymbol{\phi} \left(\boldsymbol{s}_{cp} \left(t \right) \right) \quad s.t. \quad \phi_k \left(\boldsymbol{s}_{cp} \left(t \right) \right) = x_{k,cp} \tag{4.2}$$



(a) *Surface containing a collapsed point and a trimming loop*

(b) *Projection of trimming curves and collapsed points in parametric space*

Figure 4.6: *Example of geometry with a collapsed point*

4.1.0.3. Detection of intersected elements with uniform vertex classification

One of the critical points to ensure the robust operation of the finite element solver is to correctly detect all the elements intersected by the geometry. The correct detection of intersected elements and classification of their vertices is crucial to obtain an appropriate domain decomposition and discretization of the solution. On one hand, the information from the intersections is used to identify the integration domain associated with each element. On the other hand, the classification of the vertices associated with intersected elements allows the classification of the rest of the elements in the Cartesian mesh.

We will argue the need for a preprocessing stage of the geometry using the classification problem. In Fig. 4.7, we present the reasoning stages that we will describe. We begin by joining all the intersected elements in order to identify a region of space (Σ_c) that contains the boundary of the computational domain. This union can be used as an approximation of the boundary. Additionally, the relative position with respect to the domain, which was previously encoded in the surface normal ($\mathbf{n}$), is now transferred to the boundary of the union of intersected elements. Based on the intersections and the solid's B-Rep representation, we can identify two surfaces: an internal surface ($\partial\Sigma_{c,i}$) with respect to the computational domain, and an external surface ($\partial\Sigma_{c,e}$). If we group the remaining mesh elements so that neighboring elements belong to the same group, we obtain different sets (Σ_1, Σ_2), each of which will have a uniform classification. Knowing the location of any point within a set is sufficient to know the position of any other point in the set. This information is available since, as previously mentioned, we know the classification of their boundaries ($\partial\Sigma_{c,i}$ and $\partial\Sigma_{c,e}$ determine the classification of Σ_i and Σ_e, respectively). This reasoning is what we have followed to determine the position of the non-intersected elements with respect to the geometry.

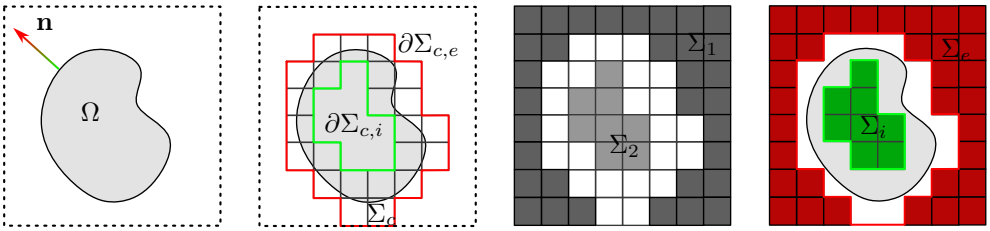
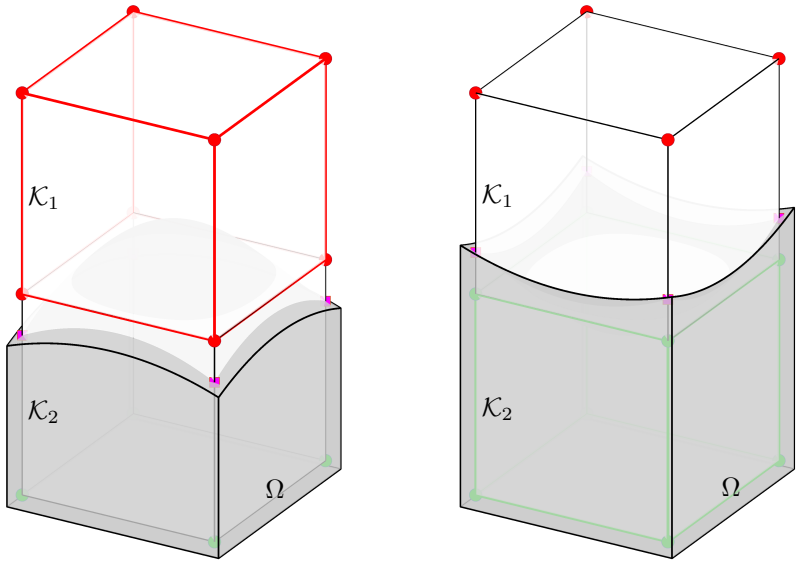


Figure 4.7: *Stages in the classification process of non-intersected elements.*

At the implementation level, the most direct way to propagate the information is by using vertex connectivity. This strategy is robust for internal and external elements. However, the relative position of the vertices is not enough to determine if an element lies on the domain boundary. An element with all its vertices classified as internal may be an intersected element, the same if the vertices were classified as external.

This situation arises when a face of the geometry penetrates the intersected element through one of its faces, without surface-edge or curve-face intersections. Such is the case of the configuration shown in Fig. 4.8a, where element $\mathcal{K}_1$ would be classified as external and would not be included in the computational mesh. At the computational level, the part of the domain of the problem Ω contained in $\mathcal{K}_1$ would be considered as belonging to $\mathcal{K}_2$, which would result in an extrapolation of the solution beyond the boundary of the elements. In cases where the extension of $\Omega \cap \mathcal{K}_1$ is small, the extrapolation of the solution in $\mathcal{K}_2$ will not have a significant effect on the accuracy of the method. However, it is not possible to quantify a priori the extension of $\Omega \cap \mathcal{K}_1$.

A more problematic situation is shown in Fig. 4.8b. This time, the vertical edges of element $\mathcal{K}_1$ are detected as intersecting the geometry's boundary and it will be classified as intersected. Its four lower vertices will be located inside the geometry and, by propagation, element $\mathcal{K}_2$ will be evaluated as internal. In this case, the computational mesh will be appropriate, but the domain of the problem will not be correctly captured. The integration domain of element $\mathcal{K}_2$ will extend vertically beyond the concave surface of the geometry, up to the boundary of the element itself. Having been classified as an internal element, it is processed as a standard finite element. If the solution in $\mathcal{K}_1$ and $\mathcal{K}_2$ were the same, the integration error in $\mathcal{K}_2$ would be compensated by the integration of $\mathcal{K}_1$, which would provide a negative contribution to the system matrices associated with the domain $\mathcal{K}_2 \setminus \Omega$. However, since the solutions in these two elements are independent, this compensation does not occur, leading to an error when evaluating the weak form of the problem. Moreover, the boundary conditions applied on the surface would only be applied to element $\mathcal{K}_1$ and, since the solution on $\mathcal{K}_2 \cap \partial\Omega$ will be defined on $\mathcal{K}_2$, there will be points on the surface that will not satisfy the problem's constraints.



(a) Proper domain integration *(b) Wrong domain integration*

Figure 4.8: *Intersected elements with misleading vertex classification*

To avoid these situations, a second stage of geometry pre-processing is implemented. This time, for each of the surfaces, a relative maximum or minimum in one of the three spatial directions will be sought. The points we are looking for must have an image located in the internal region of the parametric space of the surface; those located on the boundary will be discarded. If one of these points is found, the initial surface is divided into four daughter surfaces using as cutting entities the constant parameter lines that pass through the point. The procedure is applied recursively until a generation is reached in which all maxima and minima of the geometry are located on the boundaries of the surfaces.

Taking the processed geometry into account, there will never be elements on the boundary without intersections with the geometric model. The elements with misleading vertex classification were those where a surface S_i of the geometry entered the element through one of its faces, say F_j. Since the element faces are parallel to the Cartesian planes, this means that S_i must have at least one relative extremum in the Cartesian direction perpendicular to F_j. By dividing S_i by the extreme points, new curves $\{C_{i,k} : C_{i,k} = S_{i,m} \cap S_{i,n}, m \neq n\}$ will appear, connecting the surfaces $\{S_{i,l} : \cup S_{i,l} = S_i\}$ which must necessarily intersect with F_j. The introduction of these new curves allows the detection of both the boundary elements that contain the extreme points and those neighboring elements that have been crossed by the surface.

An representative example of the kind of output produced by the proposed technique is shown in Fig. 4.10.

4.1.0.4. Implementation issues: recovering restrictions required for B-Representation and tolerance handling

The modification of geometric entities (curves and surfaces) that form the model must be carried out considering that the final result must comply with the restrictions of a B-Representation as presented in Section 4.1.0.1. The division of curves does not pose a significant problem since it is only necessary to identify the face loops that contain the original curve and place the new curves in their place. The model obtained after the substitution is equivalent to the original and will respect the representation's restrictions to the same extent as the original model.

The division of surfaces involves deeper changes within the solid's structure. Typically, the smaller extent of the new surfaces requires decomposing each face defined on the original surface into a set of new faces. For each of the child faces, trimming loops must be defined so that the union of the child faces is equivalent to the original face. This requires modifying the curves contained in the trimming loop of the original face and adding curves that had not previously appeared in the model. Furthermore, the divided faces must be replaced by the child faces within the solid's boundary definition.

Below is a general explanation of the strategy we follow to handle the division of a surface S_i into a set of surfaces $\{S_{i,l} : \cup S_{i,l} = S_i\}$. First, the faces $\mathcal{F}(S_i)$ defined on it are identified. The trimming curves $\mathcal{C} \circ \mathcal{F}(S_i)$ associated with the set of faces $\mathcal{F}(S_i)$ are intersected with the contours of each of the new surfaces, namely $\{C_{i,l}\}$. The result is a cloud of points used to split the curves in $\mathcal{C} \circ \mathcal{F}(S_i)$ and $\{C_{i,l}\}$, yielding the sets $\mathbf{C}^I$ and $\mathbf{C}^N$, respectively. In this way, two sets of curves on $\{S_{i,l}\}$ are obtained, such that any curve from $\mathcal{C} \circ \mathcal{F}(S_i)$ or $\{C_{i,l}\}$ can be obtained by joining elements of $\mathbf{C}^I$ or $\mathbf{C}^N$, respectively, and any two elements of $\mathbf{C}^I \cup \mathbf{C}^N$ intersect at their endpoints or are disconnected.

The next step is to calculate the trimming loops that define the boundaries of the new faces. The most appropriate approach we have found to address this problem is to independently process each element $\mathcal{F}(S_i)_k$ in $\mathcal{F}(S_i)$ and each surface $S_{i,l}$. First, the trimming loops of $\mathcal{F}(S_i)_k$ are redefined using the curves of $\mathbf{C}^M$ and the parent-child mapping. Then, the trimming loops of the faces $\mathcal{F}(S_{i,l})$ derived from $\mathcal{F}(S_i)_k$ are calculated considering that they are sequences of curves such that:

- their intersection with $S_{i,l}$ is non-null
- at least one curve is contained in the loops of $\mathcal{F}(S_i)_k$
- the curves that were part of the loops of $\mathcal{F}(S_i)_k$ are traversed in the direction marked by these loops
- no curve connected to the loop lies inside the trimming resulting from applying the loop.

Once the trimming loops are defined, they are grouped to form the child faces, and the pointers in the solid's structure that pointed to those in $\mathcal{F}(S_i)$ are modified to reference their descendants.

In Fig. 4.9, the surface S is divided into the four surfaces S_1, S_2, S_3, S_4, which share a common vertex at the center of S. The face F in Fig. 4.9a is defined on S and has two trimming loops, including the curves a and b, respectively, which are traversed following the arrows drawn in the figure. If we follow the trimming loops in the indicated direction, we see that the portion of the surface contained in the face is to the right according to the direction of traversal.

Fig. 4.9b shows the result of intersecting the trimming loops with the contours of the new surfaces. The circles represent both the intersections between pairs of curves and the endpoints of the curves that make up the set $\mathbf{C}^I \cup \mathbf{C}^N$ described above. As an example, we can see that curve a will be divided into two curves a_1 and a_2, while new curves, such as m and n, will appear.

The definition of the trimming loops is exemplified for surface S_1 in Fig. 4.9c. It can be verified that the only loop that meets the imposed restrictions is $[b_3, f, j, a_1, h]$. First, the curves represented in dashed lines are disregarded as they lie outside the parametric space of S_1. Loops like $[c, e, f, j, i, h, g, d]$ are eliminated for not containing any curve from $\mathbf{C}^I$, represented in black. The loop $[b_3, g, d, c, e]$ would also be invalid for traversing b_3 in the opposite direction to that established in the definition of F. The same loops but traversed in the reverse direction, namely $[b_3, e, c, d, g]$, would also be discarded because the curves h and f are connected to the loop and remain on the material side of the trimming loop (right side according to the traversal direction).

Finally, Fig. 4.9d shows the four faces that appear from F as a result of the division of S.

The strategy we have presented is applied independently to each surface and affects the faces defined on it, but it does not impact the rest of the model. One advantage is that all surfaces can be processed in parallel, but as a result, it leads to a violation of the B-representation restrictions. To recover the properties of the B-Representation, overlaps between curves in the processed model are evaluated. When curves partially coincide, they are split into their overlapping and independent segments. Finally, the trimming loop pointers are unified to point to the latest generation of curves, which will contain a single curve for each overlap case. It is important to note that the procedure we have described not only allows recovering the B-Representation properties lost due to surface splitting but also repairs models that originally violated the B-Representation restrictions.

Correct handling of tolerances is crucial when applying the algorithms described in this section, so it seems appropriate to conclude it with some comments on this matter. In the best-case scenario, CAD models provide a geometric tolerance that indicates the distance in physical space below which two points are considered indistinguishable. However, it has been observed that this information is not sufficient. To obtain a more robust procedure, we have had to evaluate tolerances at the curve-surface pair level, considering the solid's structure. Otherwise, we encountered models with trimming curves that were at a distance greater than the tolerance of the surfaces

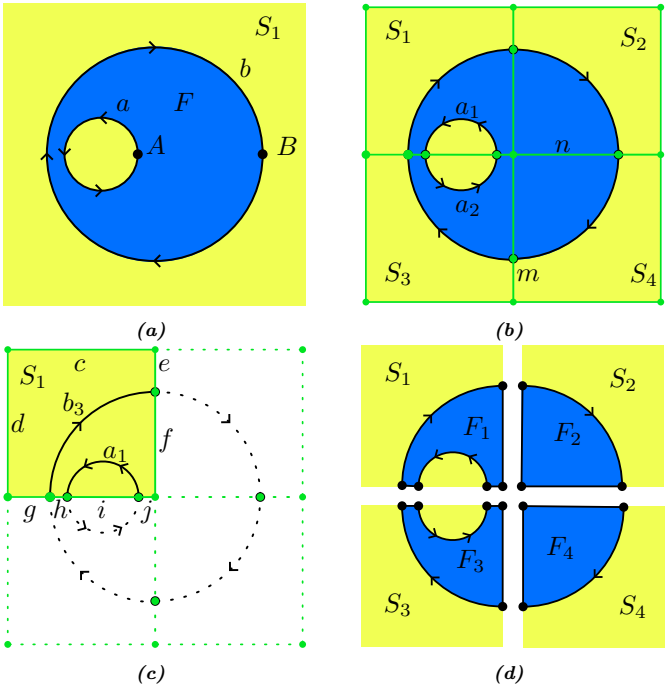


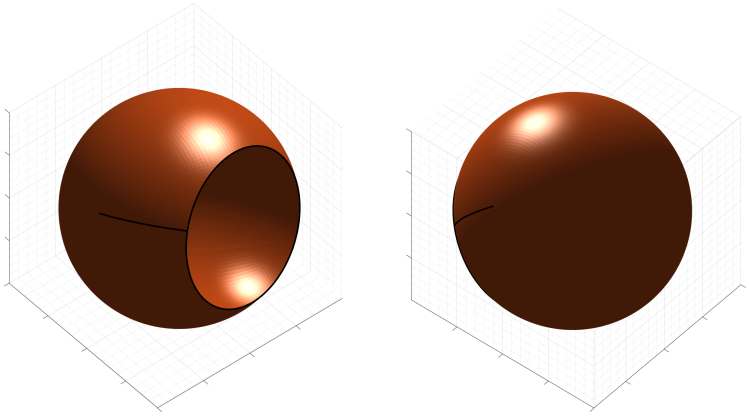
Figure 4.9: *Main stages of the algorithm followed to define new faces after surface splitting.*

they were trimming. The evaluation of overlaps must be done on a scale close to the geometric tolerance to avoid inconsistencies between entities of different scales. Beyond evaluating coincidences between entities in physical space, the model's geometric tolerance is also considered when defining convergence criteria for nonlinear problems involved in these strategies, such as the projection of points onto NURBS entities or the calculation of intersections. Convergence criteria are evaluated in physical space, and tolerances are always imposed at least one order below the model's geometric tolerance.

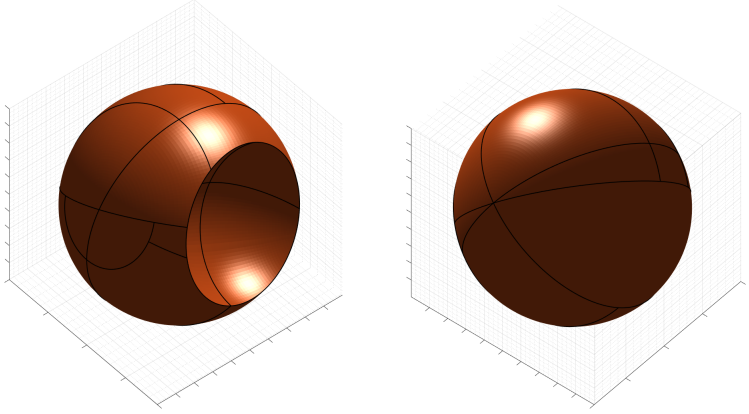
4.1.0.5. Output of geometry pre-processor

In Fig. 4.10, a geometry directly imported from a CAD program and its result after applying the treatments described in the previous sections are shown. The original geometry shown in Fig. 4.10a consists of 2 curves: a circumference and a semicircumference, and 2 surfaces: one concave and the other convex. These entities combine to form 2 faces: one convex face bounded by 4 curve segments and one concave face bounded by 1 curve, which constitute the boundary of the solid. This is a representation with very few elements that, although containing all the necessary information to define the solid, is not suitable for analysis with cgDG. The version of

the same geometry after processing is shown in Fig. 4.10b. The new representation, while still defining the same solid, consists of many more elements: 18 surfaces, 44 curves, and 18 faces. Although this new representation may seem more complex due to the number of constituents, it greatly facilitates obtaining of consistent discretizations within the cgDG environment.



(a) *CAD representation of the geometry*



(b) *Processed representation of the geometry*

Figure 4.10: *Geometry transformation to enhance the robustness of finite analysis*

The proposed methodology has been applied to relatively complex geometries, such as the hook shown in Fig. 4.11. It is worth noting that the original CAD model used as the starting point for this example did not meet the expected restrictions of a B-Representation. Once the strategies described in the previous sections are applied, we are able to obtain a model that satisfies those restrictions and is also optimal for analysis with cgDG.

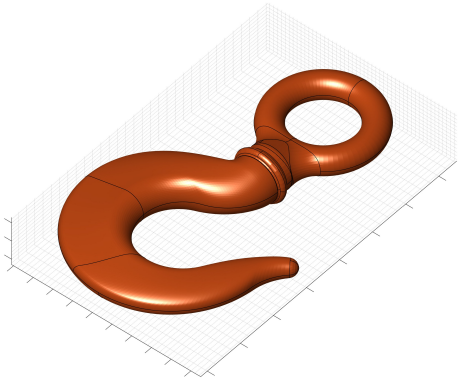


Figure 4.11: *Processed representation of a complex geometry that meets B-Representation's restrictions.*

4.2. Integration

The evaluation of the weak form operators in the internal elements of the mesh is carried out in the same way as in the case of a standard finite element solver. However, as we outlines in Section 3.1 the Cartesian structure of the mesh offers advantages from the point of view of operator evaluation. The integration of the intersected elements must be aware of the material discontinuity within the element and perform integration only in the portion of the element that lies in the physical domain. Following a similar approach to the one adopted for the 2D case we will divide the integration domains into subdomains with simpler geometries that can be obtained through the transformation of polyhedra. Inserting the definition of the geometry in the transformation, the integration can be perform on the exact domain which avoid to commit any geometrical error. The higher complexity of the 3D setting requires new integration strategies. Making a parallelism with FE mesh refinement, we could say that this new strategies compute the integration decomposition in the line of p-refinement methods; in contrast with the h-refinement approach we adopt in the 2D implementation, see Section 3.4. The shape of high order integration subdomains results from deforming a linear decomposition according to the laws of continuum mechanics.

4.2.1. Defining the topology of the integration mesh

In [63, 153], the characteristics of a technique for obtaining integration quadratures in FEAVox were presented. It proposes a strategy for calculating integration subdomains in intersected elements, which originates from the algorithm proposed by

W. Lorensen in [146], the Marching Cubes (MC) algorithm. The basic unit of the MC algorithm is a hexahedral cell with its vertices classified in a binary manner. Based on the assumption that there will never be more than one intersection per edge, an approximate reconstruction of the classification surface can be made. An immediate application is found in the field of computational graphics, where, starting from a Cartesian mesh with its vertices classified as internal or external based on their position relative to the volume, the MC algorithm is used to approximate the contour of the initial geometry.

However, the potential of the technique goes beyond that. Each MC pattern can be associated with additional properties such as connectivity degrees between vertices, magnitudes associated with fields (mean values, gradients, etc.), or surface triangulation topologies. Moreover, the information can be significantly compressed since, by applying symmetries, the 256 possible configurations are reduced to the 14 reference patterns shown in Fig. 4.12. With these characteristics, the MC algorithm presents itself as a simple and efficient procedure for assigning properties to cells within a Cartesian mesh. Returning to the application example in computational graphics, if a triangulation of the classification surface is associated with each of the reference patterns, the MC algorithm will allow us to obtain a linear representation of the solid's contour. We simply need to traverse the cells of the mesh and, for each of them: project the vertex classification into the reference pattern space, retrieve the characteristics associated with the pattern, and undo the transformation. This way, cell by cell, a triangulation of the contour is constructed.

This capability of the MC algorithm to infer the basic structure of the domain contained within a cell of the Cartesian mesh from the classification of its vertices is used in cgFEM to obtain the integration subdomains. The result of the intersection process of the Cartesian mesh with the geometry is a cloud of intersection points of planes and axes with the geometry and the classification of the mesh vertices as internal and external, the natural starting point for the MC algorithm. In the original implementation, a tetrahedralization of the domain and a triangulation of the surface are associated with each pattern. These simplices are the subdomains onto which the integration quadratures are projected. It is worth noting that this decomposition into subdomains is very efficient since the topology of the decompositions is precomputed, ensuring it conforms to the geometry. The approximation to the original model is improved using the intersection information as well as the surface definition provided by CAD software. Finally, by applying tensorial integration quadratures similar to those used in [136], the exact geometry of the domain is captured.

Remark. To simplify the implementation without losing generality, we will restrict ourselves to patterns 1, 2, 5, 8, 9, 11, and 14. These are the patterns that give rise to a single surface within the cell. Additionally, patterns 11 and 14 can be studied together as they are symmetric with respect to the YZ plane. However, this transformation is not considered because it increases the complexity of implementing the algorithm. As in [63, 153], we will use a total of 7 patterns.

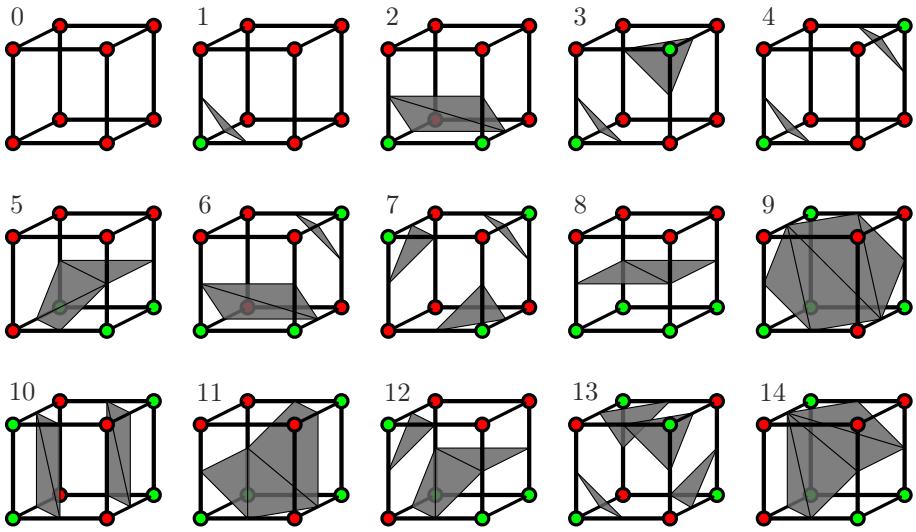
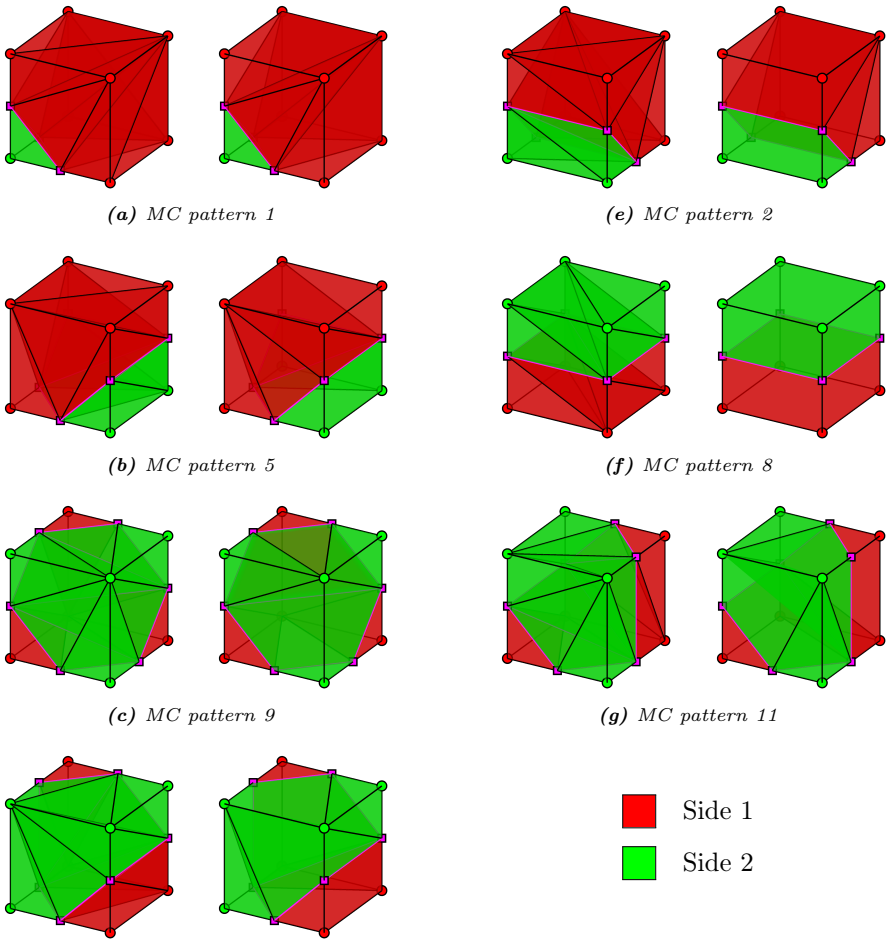


Figure 4.12: *Reference patterns in the Marching Cubes algorithm. Red and green dots represent the two different states in a binary classification.*

In this work, the catalog of shapes used to define the domain decomposition will be expanded. The volume division will consist not only of tetrahedra (T), but also triangular-based prisms (Pr), square-based pyramids (Py), and hexahedra (H). The decomposition into subdomains is constructed a priori, seeking to use the minimum number of subdomains possible. To ensure continuity between subdomains after the transformation, it is imposed that two contiguous subdomains must always share the same face. The new topologies allow for encompassing regions within a single subdomain that would be divided using only tetrahedra. The goal of this change is to reduce the cost of evaluating integrals in intersected elements by using quadratures with fewer points.

In Fig. 4.13, the decompositions defined according to the original strategy based on simplices are compared with those constructed using the catalog with all shapes. For each pattern, the decompositions on both sides of the classification surface are shown, identifying each of the regions into which the element is divided with a different color. Table 4.1 quantifies the number of subdomains composing each decomposition and their types. It can be observed that with the new strategy, a significant reduction in the total number of integration subdomains is achieved.

When quickly obtaining integration quadratures for multidimensional spaces, the most direct option is to resort to the tensor product of 1D integration quadratures. The simplest geometries can be obtained by deforming the resulting domains. For the 3D case, a Pr can be integrated as an H with one face collapsed onto a line, a Py as an H with one face collapsed onto a point, and a T as a Pr with one triangular face collapsed onto a point. However, these quadratures exhibit an accumulation of



(a) *MC pattern 1*

(e) *MC pattern 2*

(b) *MC pattern 5*

(f) *MC pattern 8*

(c) *MC pattern 9*

(g) *MC pattern 11*

(d) *MC pattern 14*

Figure 4.13: *Comparison of the topologies for each MC pattern comprising simplices (on the left) and all shapes (on the right).*

unnecessary evaluation points in the collapse areas. In recent years, much more efficient quadratures for the integration of polynomials over these domains have been presented [154, 155, 156]. Until the recent work of J. Jaskowiec, there were no quadratures precise enough to be used within a high-order finite element solver.

Taking into account the rules provided in [155, 156] (valid for polynomials up to degree 20), Table 4.1 shows the number of integration points that would be necessary to evaluate an integral with polynomial accuracy of order 7, and compares the results obtained using all shapes versus the use of simplices exclusively. Reductions in the

number of points of up to 84% are achieved, which translates into significant savings in computation times.

Apart from the volume decomposition, each pattern will also be associated with a decomposition into subdomains of both the surface and the faces of the elements in order to have the necessary information to compute fluxes on the boundaries. These divisions are made such that the 2D integration domains coincide with the facets of the 3D subdomains. The integration quadratures will be constructed taking into account the high-order rules available in the literature for quadrilaterals and triangles [154, 157].

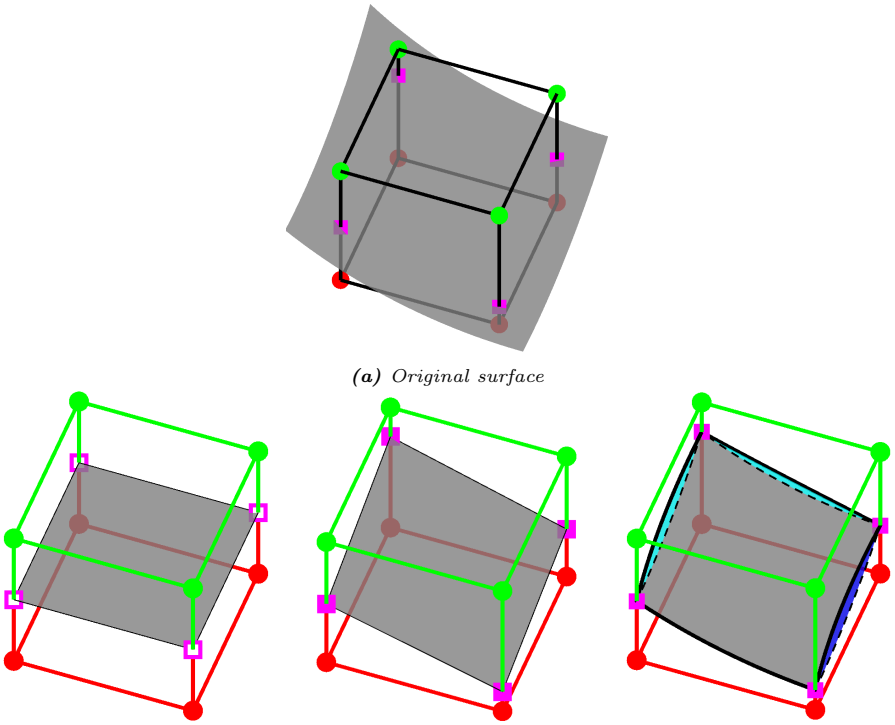
4.2.2. Approximating integration domains

The MC algorithm provides the topology of the decomposition, which must be combined with geometric information to obtain a sufficiently accurate representation of the computational domain. First, the points located on the edges are displaced from the central point and positioned at the intersections of the Cartesian mesh axes with the geometry. This corresponds to moving from a representation like the one shown in Fig. 4.14b to the one shown in Fig. 4.14c. The linear approximation of the

MC Pattern		Simplices		All shapes					Saving
ID	Side	T	npi	T	Pr	Py	H	npi	-
1	1	10	350	4	0	3	0	233	0.33
1	2	1	35	1	0	0	0	35	0.00
2	1	9	315	0	0	3	0	93	0.70
2	2	3	105	0	0	1	0	31	0.70
5	1	8	280	2	0	3	0	163	0.42
5	2	5	175	0	1	1	0	66	0.62
8	1	6	210	0	0	0	1	34	0.84
8	2	6	210	0	0	0	1	34	0.84
9	1	7	245	2	1	1	0	136	0.44
9	2	7	245	2	1	1	0	136	0.44
11	1	7	245	1	2	0	0	105	0.57
11	2	8	280	2	2	0	0	140	0.50
14	1	7	245	1	2	0	0	105	0.57
14	2	8	280	2	2	0	0	140	0.50

Table 4.1: *Number of cells and integration points per MC pattern when integrating a polynomial of order 7. We differentiate among 4 cell types: tetrahedrons tetrahedrons (T), triangular based prisms (Pr), squared based pyramids (Py) and hexahedrons (H).* npi *stands from the numbers of integration points.*

domain resulting from applying this strategy is only appropriate for simulations with low-order approximations and little sensitivity to boundary definition [33].



(a) *Original surface*

(b) *MC surface*

(c) *MC surface at intersections*

(d) *Linear decomposition of parametric space*

Figure 4.14: *Approximate representations of a surface*

By using integrations that account for the exact definition of the geometry, it is possible to expand the range of applicability of this methodology. To achieve this, the facet topology on the geometry of the MC is taken, and starting from the position of the intersection points in the parametric space of the surface, a decomposition of this space is obtained on which the integration quadrature is constructed. The vertices of the subdomains on the surface will coincide with the vertices of the volume decomposition, but the position of any other point in the surface decomposition may differ significantly from its equivalent in the linear facet decomposition. This results in the physical representation of the surface decomposition often having regions that fall outside the boundary of the element, while other regions of the surface inside the boundary are not considered for integration. In Fig. 4.14d, these discrepancies are represented in light blue and dark blue, respectively. The extent of these regions will increase as the curvature of the surfaces increases. For the simulation, the result is an incorrect assignment of boundary conditions on the surface.

The approximation of the intersection can be improved by using curved decompositions. However, introducing a curved edge in a linear decomposition can result in it intersecting with some of the edges that form the boundary of the subdomains that contain it. When calculating the domain transformation from the reference for integration to the curved subdomain, we find that the projection is not injective. It is well known that the finite element method produces erroneous solutions when the Jacobian of the transformations is not strictly positive [158, 159, 160].

The FEAVox implementation, which was taken as a starting point, included some of the improvements mentioned [63, 153]. The portion of the surface contained within an element was approximated using polynomial curves in the parametric space of the surface. Specifically, all intersections associated with axes contained within a single Cartesian plane were identified, and a B-Spline type curve was fitted. The curve represents an approximation of the intersection of the plane with the surface. By applying this strategy for all plane-surface pairs, a decomposition of the surface into curved subdomains is achieved. Each of these subdomains is associated with an element.

Let's identify some limitations of this methodology. First, it is a strategy applied globally, which can only be applied once all intersections are available. Additionally, the degree of the curves cannot be freely adjusted, as it is tied to the number of available intersections. When the element size is large relative to the surface, very little information will be available, and the approximations will be imprecise. Finally, the strategy fits one curve per surface-plane pair, leading to erroneous results when the plane divides the surface into three or more regions.

In FEAVox, the subdomains for surface integration are obtained through triangulation in the parametric space of the curved subdomains. Meanwhile, the volume integration subdomains are constructed from the surface triangles and the element nodes, resulting in a tetrahedralization of the volume. The elements added at this stage are linear (edges in the surface decomposition; edges and faces in the volume decomposition) and do not account for the shape of the boundary. The strategy fails when the elements on the boundary become highly nonlinear, leading to situations with highly distorted subdomains.

Geometry is integrated exactly using NURBS-enhanced integration. However, implementations based on the projection of tensorial quadratures are associated with high computational cost, partly due to the large number of sampling points. Transforming a tensorial quadrature into a simplex involves collapses that result in an accumulation of sampling points in certain areas. These types of quadratures are inefficient because much of the information provided by the integrand evaluations is redundant.

The following sections present strategies aimed at eliminating the limitations of the integration strategy in the original FEAVox implementation. In a scenario with high-order elements, the mesh elements will be larger, and the geometric complexity of the domains contained within the intersected elements will increase. First, a new approach for calculating the surface-plane intersection will be addressed. Subsequently, three strategies for calculating surface and volume decompositions into curved subdomains

will be presented. The strategies are ordered according to the degree of nonlinearity of the boundaries they can robustly handle.

Local approximation of surface-plane intersections

Starting with the plane-surface intersection, this work proposes to approximate the curve using a polynomial function with an image in the parametric space of the surface. To achieve this, the position in the parametric space of the nodes of a one-dimensional element of degree p will be calculated in such a way that the following functional is minimized:

$$f(\mathbf{u}^h) = \left\| S_i(\mathbf{u}(\mathbf{u}^h, t)) - X_i \right\|^2 + \alpha \frac{\left|\mathbf{S}(\bar{\mathbf{u}}(t))\right|^2}{\left|\bar{\mathbf{u}}(t)\right|^4} \det\left(\mathrm{Cov}\left[\partial_t u_1(\mathbf{u}^h, t), \partial_t u_2(\mathbf{u}^h, t)\right]\right) \quad (4.3)$$

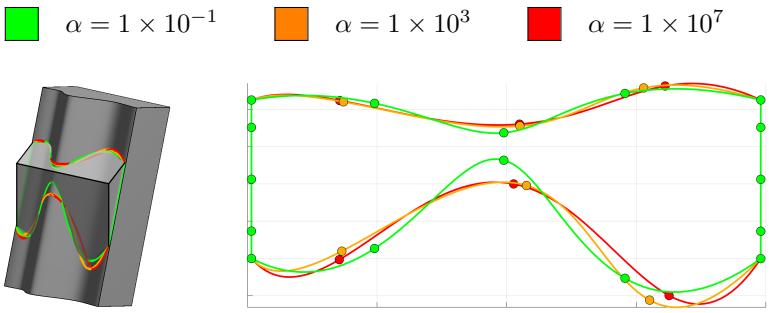
where $\mathbf{u}^h$ represents the coordinates of the internal nodes, i is the index of the coordinate perpendicular to the plane, $\mathbf{S}(\mathbf{u})$ is the surface equation, X_i is the invariant coordinate of the Cartesian plane, α is a weighting coefficient, and $\left|\mathbf{S}(\bar{\mathbf{u}}(t))\right|, \left|\bar{\mathbf{u}}(t)\right|$ are the lengths of the curves in the physical and parametric space when a linear interpolation is used.

The first term of the functional quantifies the distance between the image in the physical space of the curve that approximates the intersection and its projection in the cutting Cartesian plane. This term ensures that the function we are calculating approximates, as closely as possible, the image in the parametric space of the surface-plane intersection. The second term is added to restrict the freedom in adjusting the intersection and to prevent the resulting curve from extending beyond the boundary of the element. In Fig. 4.15, the effect of variations in the coefficient value on the final result can be seen. As it increases, the curve in the parametric space reduces its curvature and increasingly resembles the linear interpolation of the surface intersection points with the axes. In this work, the parameter value α is set to 0.1. Regarding the discretization used for the adjustment, one-dimensional elements with Fekete-type nodal distributions are considered.

Remark. It is important to emphasize that this procedure is carried out at the element level, which has significant advantages. The accuracy in approximating the intersection can be adjusted by modifying the order of the curve. The order can be adjusted locally, devoting greater computational efforts only to those intersections that, due to the complexity of the geometry, require it. Moreover, the independence between elements allows for parallel computations. Continuity between the decompositions calculated for two adjacent elements will be ensured by the symmetry between the calculations performed for each element.

4.2.2.1. Transfinite representation of integration subdomains

The curved approximation of the intersections between the geometry and the Cartesian grid can be transferred to the integration mesh using different strategies.



(a) *Representation in physical space*

(b) *Representation in parametric space*

Figure 4.15: *Approximation of surface-plane intersections*

As a first approach, the transfinite mapping proposed in [161] is used. This strategy allows for an exact representation of the geometries and provides the flexibility to independently define each of the subdomain boundaries. In our implementation, we will only consider the transformation terms due to boundary entities (vertices and edges for two-dimensional subdomains; vertices, edges, and facets for three-dimensional subdomains). In this way, the image of an interior point will be a weighting of the images of the point projected onto each of the elements that constitute the subdomain boundary.

First, the decomposition on the parametric surface needs to be adapted. The internal edges will be curved, replacing them with Bezier curves. In a first approach, curves of the same degree as the intersection approximation will be used. However, the procedure is general, and the degree of interpolation could be adapted to the specific needs of each decomposition. In Fig. 4.16, a transformation case can be seen in which the edges E1, E2, and E3 correspond to polynomials of degree 1, 3, and 2, respectively. The transformation of the entire subdomain is defined based on the location of the points drawn in green, which represent the control points of the three curves.

The geometry of the internal edges is established by solving an optimization problem. Taking as a starting point the determinant of the matrix for each of the subdomains over the geometry $J_i\left(\mathbf{u}^B\right) = \det\left(\mathbf{J}_{\boldsymbol{\tau}\xrightarrow{i}\mathbf{x}}\left(\mathbf{u}^B\right)\right)$, the objective is to find the position of the points $\mathbf{u}^B$ that form the control polygons of the internal edges in such a way that the following functional is maximized:

$$f\left(\mathbf{u}^B\right) = \min\left(\inf\left(J_i\left(\mathbf{u}^B\right)\right)\right) \tag{4.4}$$

This procedure results in the best mapping in the parametric space of the surface that can be obtained with the available degrees of freedom. We consider the best mapping to be the one that avoids excessively distorted regions. To satisfy the condition of bijective transformation, it would be sufficient to find a set of degrees

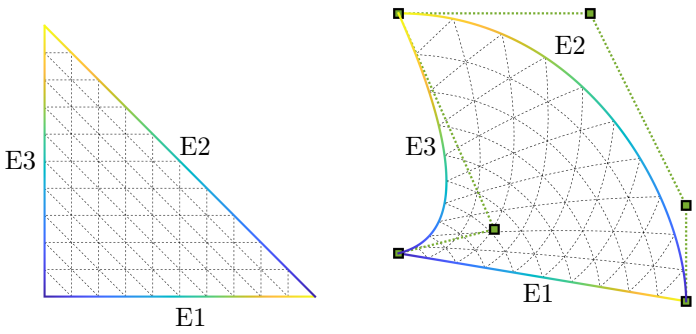


Figure 4.16: *Transfinite mapping of a triangular subdomain*

of freedom for which the functional is positive. Therefore, the optimization problem will be solved, but the tolerance for the solution will be relaxed once the functional surpasses a zero value. The relaxation of the convergence condition is associated with a considerable reduction in computation times. For each iteration of the problem, as many optimization subproblems as there are subdomains composing the decomposition will need to be solved. Each of the nested problems must also be solved iteratively and will provide the minimum value of the Jacobian of the transformation for the subdomain.

Once the decomposition on the surface is obtained, the volume decomposition is achieved using linear edges. As with the NURBS Enhanced integration, this strategy can result in twisted domains. In some situations, this can be avoided by deforming the internal edges of the mesh. In a first approximation, a weighting of the displacements $\mathbf{v}$ experienced by a set of points $\mathbf{x}_i$ located on the surface and the element boundaries will be used:

$$\mathbf{v}(\mathbf{x}) = \sum \mathbf{v}(\mathbf{x}_i)\lambda_i(\mathbf{x}) \tag{4.5}$$

where the weighting factor λ_i is a function of the distance from the evaluation point to the sampling points $d_i(\mathbf{x})$, and, denoting α as the coefficient of the maximum-to-minimum distance, follows the expression:

$$\lambda_i(\mathbf{x}) = \frac{\alpha d_i(\mathbf{x})^{-3} + (1-\alpha) d_i(\mathbf{x})^{-5}}{\sum \alpha d_j(\mathbf{x})^{-3} + (1-\alpha) d_j(\mathbf{x})^{-5}} \tag{4.6}$$

This strategy is simple and provides a rational displacement field inside the element, which is sufficient to obtain valid decompositions in those elements where the boundary curvature is not too large. Based on the ideas applied for surface decomposition, a strategy could be devised that involves solving an optimization problem that takes into account the Jacobian of the transformation for each of the subdomains. The implementation of this strategy is beyond the scope of this work.

4.2.2.2. Polynomial representation of integration subdomains

Another way to define the decompositions of integration domains within boundary elements is to use a high-order element mesh. The position of the nodes in the auxiliary mesh will be defined by drawing a parallel with a structural problem. We will start from a configuration where the nodes are distributed by applying affine transformations of the reference elements based on the coordinates of the vertices. Subsequently, the mesh will be deformed according to the laws of continuum mechanics until the mesh boundary coincides with the boundary of the integration domain. For a sufficiently rich discretization and a complete physics model, solving the problem ensures that the transformation from the initial linear mesh to the final deformed mesh will be bijective, as otherwise, the law of mass conservation would be violated.

In practice, hypotheses will be made to reduce the cost of solving the problem. In particular, this work proposes to replace the potentially nonlinear initial problem (displacements can be significant relative to the size of the elements) with a sequence of linear problems where the assumptions of small displacements and elastic behavior will be accepted as valid. The number of linear problems to be solved will depend on the maximum distance between a node of the linear mesh and its deformed image. To ensure that the procedure is consistent with the physics of the problem, after each calculation, the system stiffness matrix is updated to account for the new configuration. The final result will be an approximation of that obtained by solving the elastic problem with large displacements.

We first consider the decomposition into subdomains of the boundary of the solid. The starting point will be the decomposition of the parametric space of the surface into elements with straight sides. The mesh will consist of elements of the same degree used to approximate the intersections of the surface with the Cartesian planes. The elements considered are Lagrangian quadrilaterals with their nodes positioned at Fekete points, and triangles with their edge nodes also positioned at these points, following the strategy presented in [162]. Next, the mesh is deformed by applying Dirichlet conditions on nodes belonging to the boundary. The imposition of boundary conditions is trivial since the interpolation degree and node distributions are chosen so that the final position of the boundary nodes of the two-dimensional mesh matches the nodes used to define the intersections of the surface with the Cartesian planes.

Remark. In the case of a trimmed surface, the position of the nodes on the boundary will be determined by the polynomial fit of the projection of the trimming curve onto the parametric space.

The decomposition of the volume is computed similarly. First, it must be considered that the facet mesh is defined in parametric space, while the volume decomposition is defined in physical space. The geometry of the surface facets can be approximated using two-dimensional elements with the same topology as the facets in parametric space, with their nodes positioned at the images in physical space of the nodes located in parametric space. This will be the boundary geometry that we seek to recover with the three-dimensional mesh. The elements we will use for interpola-

tion will conform to the surface mesh, and the geometries are those presented when studying MC patterns. Fig. 4.17 shows, as an example, the reference elements and node distributions that would be used to solve a degree-3 mesh. Again, the curved mesh is obtained by solving the linear elastic problem with Dirichlet boundary conditions applied to the nodes. Specifically, the position of the nodes located on the boundary of the Cartesian mesh element and the geometry is restricted. Table 4.2 lists the degrees of freedom and the constraints provided by each node depending on its position.

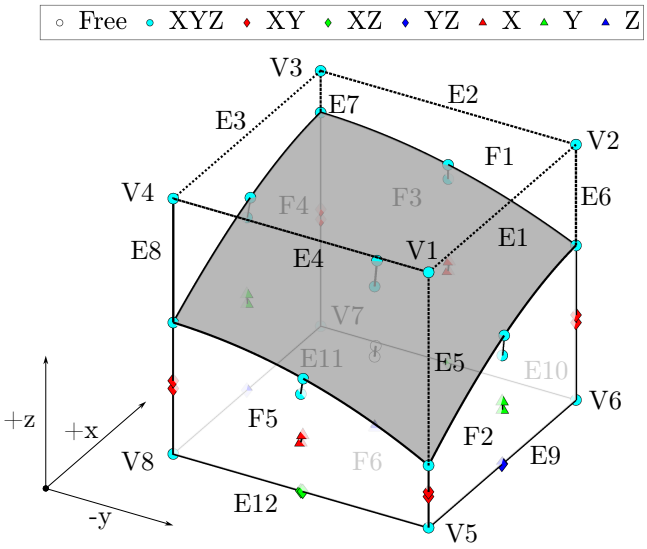


Figure 4.17: *Constrained directions in 3D elastic problems and element codification*

Location	Degrees of freedom	Constraints
On surface	-	$\{U_x, U_y, U_z\}$
{V1,V2,V3,V4,V5,V6,V7,V8}	-	$\{U_x, U_y, U_z\}$
{E1,E3,E9,E11}	$\{U_x\}$	$\{U_y, U_z\}$
{E2,E4,E10,E12}	$\{U_y\}$	$\{U_x, U_z\}$
{E5,E6,E7,E8}	$\{U_z\}$	$\{U_x, U_y\}$
{F1,F6}	$\{U_x, U_y\}$	$\{U_z\}$
{F2,F4}	$\{U_x, U_z\}$	$\{U_y\}$
{F3,F5}	$\{U_x, U_y\}$	$\{U_z\}$
Internal	$\{U_x, U_y, U_z\}$	-

Table 4.2: *Degrees of freedom and constraints in 3D elastic problems*

Once the volume division is available, the decomposition into element faces is obtained by taking the positions of the nodes of the integration mesh on the Cartesian planes and using flat elements of the same type considered for the surface mesh in parametric space.

It should be noted that the elasticity problems solved only involve domains within a specific element of the Cartesian mesh. If we look at Table 4.1, we observe that in most MC patterns, meshes with 4 elements or fewer will be solved; the mesh cardinality is only higher for configuration 1 and side 1, reaching 7 elements. In any case, these are small-dimension problems with very short computation times. The independence between elements also allows for the parallelization of computations. Due to these characteristics, the calculation of the integration meshes can be performed quickly and will not represent a significant cost compared to solving the objective problem.

On the other hand, the independent solution of neighboring elements implies that, in general, nodes located at the interface will not be conforming. This does not pose a major problem since the interface domain is the same, and only the transformations viewed from one element or another are modified. In Section 4.2.3, this situation and its effect on the calculation of fluxes between elements will be discussed in more detail.

4.2.2.3. Polynomial representation of integration subdomains with NURBS-enhancement

To improve the domain capture, it is proposed to use a modification of the initial polynomial mapping, in line with the transfinite transformation presented in [161]. The goal is to apply a correction to the polynomial transformation to recover the exact definition of the boundary. In this exposition, we will consider only two-dimensional elements with one side on the boundary and three-dimensional elements with only one face or edge on the boundary. This simplification is applied as these are the morphologies that appear in the MC patterns. The mapping sought is of the form

$$\boldsymbol{\phi} = \boldsymbol{\phi}_p(\boldsymbol{\tau}) + \Delta\boldsymbol{\phi}_c(\boldsymbol{\tau}) \tag{4.7}$$

where $\boldsymbol{\phi}_p$ represents the polynomial transformation and $\Delta\boldsymbol{\phi}_c$ is the correction introduced by considering the exact definition of the boundary c. This second term is composed of two factors: the boundary bubble function $\boldsymbol{\psi}_c$ and the interpolation function σ_c:

$$\Delta\boldsymbol{\phi}_c(\boldsymbol{\tau}) = \boldsymbol{\psi}_c(\boldsymbol{\tau})\,\sigma_c(\boldsymbol{\tau}) \tag{4.8}$$

The interpolation function σ_c is always constructed as the sum of the linear interpolation functions of the vertices located on the boundary c. As a result, σ_c has a unitary value on the edge or face over the NURBS and a null value on the rest of the nodes. This factor is essentially a weighting that reduces the effect of the correction linearly with the distance to c in the reference space of the subdomain.

The formulation of the bubble function $\boldsymbol{\psi}_c$ is more complex and depends both on the topology of the subdomain and the specific boundary under consideration. Below, we describe the philosophy we have followed to define them without delving into the details of the formulation. In Appendix A, a more detailed explanation of its derivation is provided, along with the calculation formulas.

In general terms, $\boldsymbol{\psi}_c$ for a given boundary is calculated as the difference between the parametric definition of the boundary Γ_c and the polynomial definition of the same $\Gamma_c^{\phi_p}$. In order to evaluate this difference, it is first necessary to define the projection onto c of any point in the subdomain. The transformations proposed in this work have been constructed with the following properties in mind:

- the transformations are continuous
- the Jacobians of the transformations are non-negative.
- the transformations are identical to the entity for points on the boundary of interest
- the projection of a point on an edge of the subdomain connected to the target boundary will be located on the vertex connecting them.
- the projection of a point on a face of the subdomain connected to the target boundary will be located on the edge they share, and can be obtained by knowing only the position of the point within the face containing it.

These last two characteristics are important because they ensure that points located on the subdomain boundaries other than the target boundary will not experience changes in their position once the transfinite correction is applied. Bubble functions are characterized by being null functions on the boundary of the support domain, which is precisely where those points will be projected. The mappings proposed in [161] do not respect these restrictions on their own, and the desired behavior is achieved by acting on the interpolation functions. Another advantage of our approach is that, thanks to these last two restrictions, the transformations between contiguous subdomains will be continuous.

Remark. Note that, although it is not used in this work due to the characteristics of the decomposition, the proposed interpolation functions and projections define a generic transfinite mapping. It would only be necessary to modify $\Delta\boldsymbol{\phi}_c$ so that the correction is cumulative, and the bubble functions of each component take into account the corrections introduced by the lower-dimensional components.

4.2.3. Implementation issues

4.2.3.1. Mismatch of integration meshes at boundary element interfaces

As highlighted in Section 4.2.2, the integration mesh is generated for each element independently. Therefore, it is not guaranteed that the topology of the integration subdomains on a face will be the same when viewed from the two elements that share that face. This situation did not occur in the two-dimensional case, where the decomposition of the interface between elements had a unique solution. In the three-dimensional case, we will need to select one of the boundary decompositions and generate the integration quadrature based on it. The choice of decomposition will be arbitrary as long as it is an interface between elements of the same level (the faces are coincident). In the case of an adapted computational mesh, the decomposition of the face obtained for the element of a higher mesh level must be considered. The decomposition of the face for the larger element (lower level) would be obtained as the union of the decompositions established for its smaller neighboring elements (higher level).

The evaluation of the operators in the volume can also be affected by this individualized treatment of the elements. The boundary of the decomposition does not exactly match the boundary of the element due to the approximation of the intersection between the surfaces and the Cartesian planes. Depending on the topology of the decomposition and the characteristics of the geometry within each element, the integration subdomains will be modified in a specific way, and it may happen that integration subdomains of neighboring elements intersect each other or that there are gaps between neighboring decompositions. This situation arises as a consequence of the deformation of the initial linear mesh.

The surface decomposition for the integration of boundary conditions does not exhibit this undesirable behavior. The reason is that the treatment given to the boundary is symmetric when viewed from one element or the other. A similar treatment could be applied to the decomposition of faces to ensure that the integration meshes for neighboring elements are conforming. Two modifications would need to be made with respect to the process outlined in this work. First, a constraint would be added to the volume decomposition, enforcing that the topologies on intersecting faces are always the same for a given 2D MC pattern. Additionally, the treatment of facets on the element faces would need to be identical to the treatment of facets on the geometry. The 2D problem would be solved, and the calculated nodal displacements would be used as boundary conditions for the volume problem. Note that different decompositions of non-intersecting faces will never pose a problem since all the nodes of the integration facets will be located on the Cartesian plane. With these modifications, the treatment of a boundary viewed from two neighboring elements would be identical, and the continuity of the integration mesh would be restored.

The price to pay for the continuity constraint on element faces will be an increase in the number of integration subdomains for both volume and boundary. By using decompositions with more subdomains, the number of integration points will increase, and with it, the cost of evaluating the operators. As we will verify in Section 4.4, no artifacts caused by the lack of continuity in the integration mesh have been detected; therefore, it was not deemed necessary to modify the topologies of the decompositions, and the implementation has not been addressed.

4.2.3.2. Codification of internal interfaces

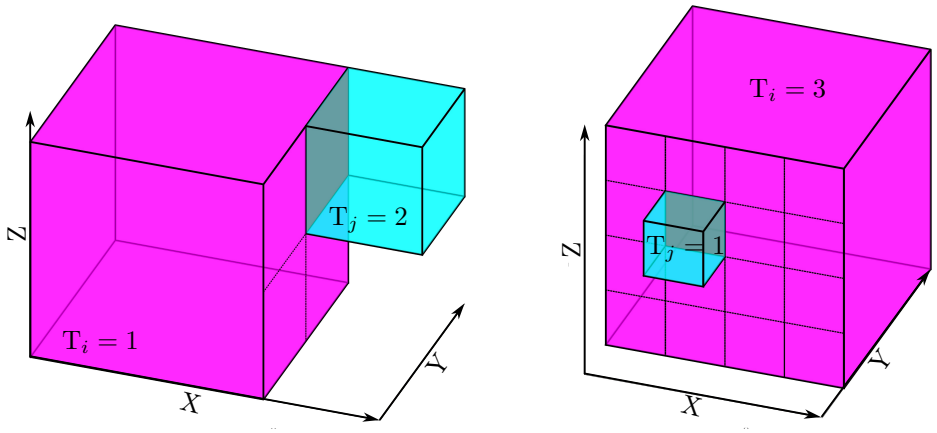
As in the planar case, the integrals for internal interfaces can be calculated as the difference between the results of multiplying a mass matrix by the incoming flux evaluated at the nodes of the face for the element and its neighbor. The mass matrix resulting from the integration depends solely on the interpolation degrees on both sides of the interface and the region of the face involved. The hierarchical structure of the mesh implies that there will be a finite number of these combinations, which will depend on the maximum allowed jump in the degree and level of neighboring elements. For a discretization without jump restrictions, elements at mesh levels $l_i \in [l_l, l_u]$ and tensorial interpolation bases of degrees $p_i \in [p_l, p_u]$, the total number of possible combinations will be:

$$n_{\text{comb}} = \big((p_u - p_l + 1)^2\big)! \sum_{i=l}^{u} 2^{2\,(l_u - l_i)}$$

For uniform discretizations or with small ranges in the parameters, n_{comb} remains at low values, and precomputing all possible combinations is not problematic. However, as more flexibility in the approximation is allowed, n_{comb} grows rapidly, and calculating all possible combinations becomes a significant workload. In most cases, a substantial portion of the combinations will not be present in the mesh; for example, sharp changes in node density between neighboring elements will be very rare. For this reason, the interface configurations present in the computational mesh will be identified, and only the face mass matrices for these combinations will be evaluated.

The interface between elements E_i and E_j can be uniquely identified using a codification of 6 numbers, namely $\{\mathrm{T}_i, \mathrm{T}_j, C_{i,j}, \Delta l_{i,j}, \xi_{i,j}, \eta_{i,j}\}$. The first two figures, T_i and T_j, identify the polynomial bases used over the domains of E_i and E_j, respectively, while $C_{i,j}$ will identify the Cartesian direction perpendicular to the interface. These 3 numbers are sufficient to determine the interpolation base on the interface as seen from each of the elements. The three remaining parameters characterize the integration domain. It is assumed that elements E_i and E_j belonging to mesh levels l_i and l_j, respectively, satisfy that $l_i \leq l_j$. The first value of this second block will be the level difference between the elements related by the interface, $\Delta l_{i,j} = l_j - l_i$. The last two values represent the relative position of the smaller element with respect to the larger one. If we obtain a Cartesian mesh of level $\Delta l_{i,j}$ over the face of E_i, $\{\xi_{i,j}, \eta_{i,j}\}$ represents the indices of the element whose domain coincides with the

interface between E_i and E_j. The indices are defined in the interval $[1, 2^{\Delta l_{i,j}}]$ and increase with the Cartesian directions. Fig. 4.18 shows two examples of interfaces with their corresponding codification.



(a) *Interface code:* $\{1, 2, 1, 1, 2, 2\}$ ***(b)*** *Interface code:* $\{3, 1, 2, 2, 2, 3\}$

Figure 4.18: *Two examples of internal interfaces and their extended codifications*

The codification used in the three-dimensional implementation presents certain differences with respect to that used for the planar case. The latter characterized each interface based on the degree of the elements and the portion of the face involved in each case. For the 3D codification, it is assumed that the elements are always arranged so that the first element is the larger one. This consideration simplifies the codification by eliminating the entries that identified the portion of the face integrated for the second element, which will always be integrated over its entire extent. Conversely, the 3D codification introduces an index to specify the direction, whereas the 2D codification was invariant with respect to rotations. This index was introduced for practical purposes as it greatly facilitated the implementation, and it is not expected to have a significant impact on computational cost. In fact, it was initially proposed to use a compact codification that assigned a value to each combination of interpolation degrees per element of the type $C_i^p = (p_u - p_l)(p_{i,s} - p_l) + (p_{i,t} - p_l)$, where $p_{i,s}$ represents the interpolation degree of element E_i in the Cartesian direction s, and $s < t$. Using C_i^p reduces the number of figures required to codify an interface and restores invariance with respect to rotations. In the case of a uniform mesh with isotropic elements, the 6-digits codification will identify 3 types of interfaces, one for each direction, while the compact codification would only detect one.

4.3. Stabilisation strategies

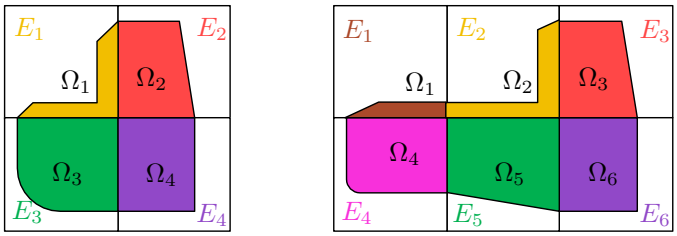
In the development for the 2D case, it became clear that the most critical point when implementing the discontinuous Galerkin formulation in the context of a Fictitious Domain method was achieving a stable discretization. In Section 3.5, the suitability of reassigning integration domains of elements with a low domain ratio to neighboring elements and extrapolating the solution of these beyond their boundary was demonstrated. This strategy allowed for discretizations with a stable time step comparable to the discretization without intersected elements. However, the choice of the neighboring element to extend is somewhat arbitrary and, although criteria were imposed to improve the selection, it is likely that cases may arise where the choice is not optimal. In the 3D context, the additional dimension considerably expands the number of possible cases, making it much more difficult to establish a set of rules to guide the process toward identifying the most suitable element. For this reason, stabilization of the discretization in the three-dimensional case will follow a different approach, aiming to develop alternative stabilization strategies that do not require the stage of selecting optimal support elements. Two stabilization techniques have been developed: multiple reassignment and weighted extrapolation.

4.3.1. Multiple reassignment

The alternative stabilization is based on the same ideas as the original solution, but this time, instead of selecting a single element to extend its solution over an unstable neighboring region, all stable elements neighboring an unstable region will extend their solution over it. To facilitate visualization and understanding of this strategy, the explanation will be done using a two-dimensional case without losing the generality of the method. Validation will be performed using three-dimensional problems.

Let's consider how the discretization presented in Fig. 4.19a, composed of four elements where element E_1 is considered unstable, would be handled. The computational domain is decomposed into subdomains according to $\Omega_i = \Omega \cup E_i$, and, due to the characteristics of the base mesh, the decomposition satisfies $\{\Omega_i : \Omega_i \cap \Omega_j = \emptyset \; \forall \; i \neq j, \; \cup_{i=1}^{4} \Omega_i = \Omega\}$. Note that, due to the symmetry of Ω_1 and the geometry of the neighboring domains, it does not seem reasonable to establish any kind of priority regarding which element will assimilate the domain.

The proposed strategy is to resolve the instability using a discretization formed by the elements $\{E_2, E_3, E_4\}$ where the domains associated with E_2 and E_3 will be $\Omega_2^{\text{st}} = \Omega_1 \cup \Omega_2$ and $\Omega_3^{\text{st}} = \Omega_1 \cup \Omega_3$, respectively. Since $\Omega_2^{\text{st}} \cap \Omega_3^{\text{st}} = \Omega_1$, the new decomposition of the computational domain does not meet the first property of the base decomposition. Thanks to the use of the DG formulation, this type of



(a) *Configuration A* ***(b)*** *Configuration B*

Figure 4.19: *Two configurations where it is unclear which element is preferable to extend over the unstable region*

discretization does not pose a problem, and it will only be necessary to correctly apply the fluxes on the boundaries of each subdomain.

We will study the fluxes on the boundary of Ω_1, particularly the one that appears at the interface with Ω_2. First, we will consider the solution in Ω_1 as an extension of the solution in E_2. In this case, the field inside Ω_1 will be $\mathbf{U}^- = \mathbf{U}_2$, while on the other side of the interface, it will be $\mathbf{U}^+ = \mathbf{U}_2$. Since the flux function is linear with respect to the field, the continuity of the field across the interface means that the flux jump will be zero in these connections between subdomains. For the implementation, these fluxes will never be calculated. If we now consider that the solution in Ω_1 corresponds to that calculated from E_3, the situation changes. The external field remains the same $\mathbf{U}^+ = \mathbf{U}_2$, but now the internal field is $\mathbf{U}^- = \mathbf{U}_3$. The DG formulation assumes that fields $\mathbf{U}_2$ and $\mathbf{U}_3$ can present a jump at the interface. Therefore, when calculating the solution in E_2, the flux jump at the interface between Ω_1 and Ω_2 must be taken into account. An identical study can be made for the interface connecting Ω_1 and Ω_3, while the relationship between Ω_4 and its neighboring domains remains unchanged.

Once the connections between domains are established and the solution jumps viewed from the different elements are identified, the solution update within the time loop can follow the usual procedure. It should be noted that the proposed solution is applied at the discretization level and is independent of the problem being solved. The weak form will continue to be solved at the element level, and the only effect of the stabilized discretization will be a slight increase in the number of fluxes that need to be calculated. The multiplicity of solutions in the stabilized domains requires calculating as many flux values per interface as solutions have been extrapolated, whereas in the base discretization, the interface fluxes are unique. For each unstable element, the calculation will shift from n interfaces in the base mesh to, at worst, $(n-2)n_s$ in the stabilized mesh, where n_s is the number of stable neighbors, each contributing $n-2$ interfaces from the unstable element (the flux is known to be zero at the interface between the unstable and extrapolated element). Generally, the calculation of the new fluxes will not have a significant impact on computational cost since the set of unstable elements usually constitutes a small fraction of the total elements in the mesh, and due to their condition as intersected elements, the

number of stable neighboring elements is typically low. In fact, in the configuration of Fig. 4.19a, the opposite effect occurs: the calculation goes from 12 fluxes ($n = 3$ with 2 internal interfaces and 1 boundary interface for each element) to 11 fluxes (3 fluxes per stable element plus the 2 contributed by the unstable element). In any case, the slight increase in the resolution cost is more than justified when considering the effect of stabilization on the time step, see Section 4.4.

The described strategy is valid for configurations where the unstable domains are surrounded solely by stable domains. In a configuration like the one presented in Fig. 4.19b, there will be an indeterminacy when calculating the fluxes. When formulating the conservation equations for element E_4, we see that the external field at the interface between Ω_1 and Ω_2 is not unique, as both $\mathbf{U}_3$ and $\mathbf{U}_5$ have support in Ω_2. Therefore, it is not possible to calculate the fluxes on the boundary of Ω_4^{st}.

4.3.1.1. Unifying solutions on unstable domains

To solve this situation, it is proposed to post-process the solution of the stabilized discretization and obtain an approximation on the base mesh structure in which there is no overlap between the supports of the interpolation functions. If we assume that we have a field $\mathbf{U}^*$ with these characteristics, the evaluation of the fluxes at the interfaces would be trivial since we could consider that the external field will always result from evaluating $\mathbf{U}^*$, while the internal field would be evaluated as usual. To apply this strategy, it would only be necessary to add a stage within the temporal integration loop where the solution in unstable elements, as mentioned in the previous paragraph, would be post-processed.

A wide variety of strategies can be considered when reconstructing the solution. Below, two simple methodologies with reduced computational cost are presented. In any case, it should be noted that the finite element calculation precedes the reconstruction stage and is independent of the chosen strategy.

4.3.1.1.1. Average solution

A first approach is to identify the solutions with support in each domain and define the post-processed solution as the mean value of those:

$$U_i^*(x) = \frac{\sum_{j \in \mathcal{N}_i} U_j(x)}{|\mathcal{N}_i|} \quad s.t. \quad \mathcal{N}_i = \{j : \Omega_j^{\text{st}} \cap \Omega_i = \Omega_i\} \tag{4.9}$$

This strategy facilitates the evaluation of the post-processed field since it can be operated on as if it were the solution of the finite element method. Given that the mean field will be of an order equal to or lower than that of the averaged fields, we know that the post-processed solution in an unstable element E_i can be adjusted to a polynomial of degree $\hat{p}$, where $\hat{p}$ is the maximum degree of interpolation of the neighboring elements. Such a polynomial distribution can be represented using the same polynomial basis that would be used in the finite element calculation. Since we

are working with a nodal approximation, the averaged field will be evaluated at the nodes $\{x_{i,s} : s \in \{1, 2, ..., \mathrm{n}_p\}\}$, and we will use the shape functions $\{N_{p,s}(x) : s \in \{1, 2, ..., \mathrm{n}_p\}\}$ to interpolate and evaluate the solution at any point $x \in \Omega_i$:

$$U_i^*(x) = \sum_{s=1}^{\mathrm{n}_p} N_{p,s}(x)\bar{U}_i(x_{i,s}) = \sum_{s=1}^{\mathrm{n}_p} N_{p,s}(x)\frac{\sum_{j\in\mathcal{N}_i} U_j(x_{i,s})}{|\mathcal{N}_i|} \tag{4.10}$$

Although it has very interesting characteristics from the point of view of computational speed, this approach seems somewhat simplistic, especially if we consider that we will be working with high-order approximations.

4.3.1.1.2. Interpolated solution

A more sophisticated approach would be to weight the solutions such that the contribution of an element is greater the closer the evaluated point is to it. It seems reasonable to think that the solution in the extrapolated region will be more accurate the closer we get to the boundary of the extended element. Most of the domain associated with the extended element will be located inside it, and this region will have more weight when evaluating the operators of the weak form. With this idea in mind, it is proposed to evaluate the reconstructed field taking into account the distances from the point to the nearest face of the neighboring element $d_{E_j}(x)$:

$$U_i^*(x) = \frac{\sum_{j\in\mathcal{N}_i} \frac{U_j(x)}{d_{E_j}(x)}}{\epsilon + \sum_{j\in\mathcal{N}_i} \frac{1}{d_{E_j}(x)}} \tag{4.11}$$

where ϵ is a small positive value added to avoid singularities. In this work, we set it based on the size of the fictitious domain used to generate the Cartesian mesh according to the expression $\epsilon = 10^{-10}L$, where L is the smaller dimension of the bounding box.

This approach results in a rational distribution that is continuous with the solution of the neighboring elements at the boundaries, while inside the element, it is a continuous weighting of the polynomial solutions based on the proximity of the point to each of the element's faces. The reconstructed solution presents discontinuities at the vertices of the element. However, these jumps do not pose any problem for the evaluation of fluxes between elements, as the recovered solution inside the interfaces is perfectly defined.

It is evident that evaluating the field obtained according to Equation (4.11) will be more complex than according to Equation (4.10). As we recall, the purpose of these fields was to have a unique expression of the solution to evaluate the fluxes over certain interfaces of extended domains. Due to their characteristics, the number of these interfaces will be small, and the integrals will have to be evaluated independently. Therefore, it is possible to precompute many of the data required for the integration of the fluxes, making the evaluation within the time integration loop have little impact.

In any case, the evaluation for the averaged field will always be more efficient since it can be expressed as if it were an intersected interface between elements of the base mesh, reducing it to the multiplication of a mass matrix by a vector. Using the rational reconstruction will necessarily involve evaluating the numerical integral for each iteration of the time loop.

4.3.1.2. Additional comments

At times, it is convenient to use a stabilization strategy more closely aligned with that presented for the planar case. There will be situations where there is a clear criterion to discard one of the neighbors due to the consequences that its extrapolation would have on the accuracy of the calculation. In these cases, the approximation can be extrapolated only for those elements that are more suitable, without this affecting the stabilization process.

A convenient strategy is to extrapolate only those elements for which the expansion of the support domain has a minimal effect on their solution. We can hypothesize that changes in the solution may be directly related to changes in the geometry of the integration domain of the weak form operators. If we capture the distribution of the computational domain within an element E_j by evaluating the location of its center of mass $\hat{\mathbf{x}}_j$, we can estimate the effect of the extrapolation by calculating the displacement of its center of mass when the unstable domain $\Delta\bar{\mathbf{x}}_{j,i}$ is added. The final objective would be to obtain a stabilized discretization where, for each unstable element, the following condition holds:

$$\left| \frac{\|\Delta\bar{\mathbf{x}}_{j,i}\|}{h_j} - \frac{\|\Delta\bar{\mathbf{x}}_{k,i}\|}{h_k} \right| \leq \mathrm{K}_{st} \quad \text{with} \quad j,k \in \mathcal{N}_i \tag{4.12}$$

where $\mathrm{K}_{st} \in [0,1]$ is a parameter set by the user that limits the range of variation of the expanded domains. For $\mathrm{K}_{st} = 0$, the strategy presented for the planar problem is recovered, except in those particular cases where the unstable domain and its neighbors exhibit symmetries. The opposite case occurs for $\mathrm{K}_{st} = 1$, where the strategy that extrapolates all stable neighbors is recovered. For low values of K_{st}, stable discretizations are obtained in which the extrapolation of approximations over regions far from the element's boundary is significantly reduced.

In Section 4.4, the performance of this stabilization approach, as well as each of its variants, will be studied.

4.3.2. Composite extrapolation

Until now, stabilization strategies have been proposed in which stable approximations adjacent to unstable domains are identified and their support is modified to completely encompass the unstable domain. When there are several elements that can be extended for the same unstable region, one ends up with as many solutions in

these domains as neighboring elements, which worsens the solution in the extended elements and necessitates post-processing of the finite element solution. In this scenario, it seems more convenient to divide the unstable domains and assign each of the resulting subdomains to a neighboring element, so that the finite element solution at each point is unique and the influence on the initial approximation is minimized. It would be desirable to find weighting functions $\phi_j(\mathbf{x})$ such that the solution in Ω_i is defined as:

$$U_i(\mathbf{x}) = \sum_{j \in \mathcal{N}_i} \phi_j(\mathbf{x}) U_j(\mathbf{x}) \tag{4.13}$$

where

$$\phi_j(\mathbf{x}) = \begin{cases} 1 & if \quad \mathbf{x} \in \Omega_{i,j} \\ 0 & if \quad \mathbf{x} \notin \Omega_{i,j} \end{cases} \tag{4.14}$$

and it holds that $\cup_{j \in \mathcal{N}_i} \Omega_{i,j} = \Omega_i$ with $\Omega_{i,j} \cap \Omega_{i,k} = \emptyset$ for all $j \neq k$.

The critical point in implementing this strategy is defining a procedure for the appropriate decomposition of the unstable domains. The properties that the decomposition must satisfy to be considered appropriate are not evident, and it is very likely that the algorithm would be computationally expensive, involving the calculation of new intersections.

The alternative is to extend the support of $\phi_j(\mathbf{x})$ in such a way that it is not necessary to calculate this decomposition but the effect of the extrapolation on the solution in the extrapolated elements is minimized as much as possible. The functions $\phi_j(\mathbf{x})$ should be continuous, with no changes in definition, and a partition of unity in Ω_i. Additionally, desired behaviors of $\phi_j(\mathbf{x})$ include being smooth, taking values in the range $[0, 1]$, being equal to unity on the boundary where Ω_i meets Ω_j, and zero on the rest of the interfaces. The weighting strategy presented in Equation (4.11) exhibited these characteristics and would give weighting functions $\phi_j(\mathbf{x})$ of the type:

$$\phi_j(\mathbf{x}) = \frac{1}{\sum_{k \in \{\mathcal{N}_i \setminus j\}} \frac{1}{d_{E_k}(\mathbf{x})}} \tag{4.15}$$

Introducing Equation (4.15) into Equation (4.13) and taking into account the definition of the field based on nodal values $U_j(\mathbf{x}_{j,l})$ and interpolation functions $\{N_l(\mathbf{x}) : l \in nod_j\}$, we arrive at:

$$U_i(\mathbf{x}) = \sum_{j \in \mathcal{N}_i} \left(\frac{1}{\sum_{k \in \{\mathcal{N}_i \setminus j\}} \frac{1}{d_{E_k}(\mathbf{x})}} \sum_{l \in nod_j} N_l(\mathbf{x}) U_j(\mathbf{x}_l) \right) \tag{4.16}$$

where we can group the terms that depend on the position and redefine the interpolation functions $\{N_l^*(\mathbf{x}) : l \in nod_j\}$ of element E_j as:

$$N_l^*(\mathbf{x}) = \begin{cases} \frac{N_l(\mathbf{x})}{\sum_{k\in\{\mathcal{N}_i\setminus j\}} \frac{1}{d_{E_k}(\mathbf{x})}} & if \quad \mathbf{x}\in\Omega_i \\ N_l(\mathbf{x}) & if \quad \mathbf{x}\in\Omega_j \end{cases} \tag{4.17}$$

to obtain:

$$U_i(\mathbf{x}) = \sum_{j\in\mathcal{N}_i} \left(\sum_{l\in nod_j} N_l^*(\mathbf{x}) U_j(\mathbf{x}_l) \right). \tag{4.18}$$

Note that, unlike U_i^* in the previous section, which was the result of post-processing the finite element solution, U_i is the approximation to the solution, and $\{N_l^*(\mathbf{x})\}$ will be the approximation basis in the unstable elements. In Fig. 4.20, a 2D example of the new shape functions for a configuration in which two linear elements extend over their common neighbor can be seen.

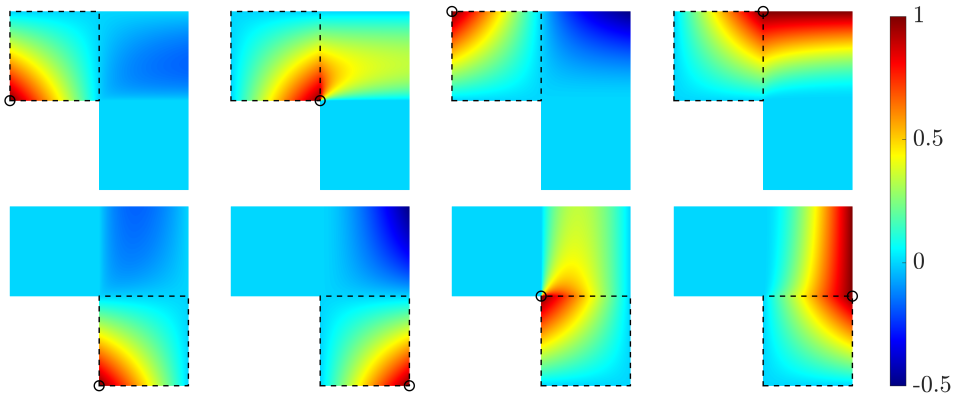


Figure 4.20: *Example of weighted shape functions associated with each node (cincumference) of the elements (dashed line) that define the approximation over a common unstable neighbor element*

4.3.2.1. Weak form for stabilized regions

Let us study the weak formulation of the electromagnetic problem for stabilized domains. The expression is the same as that derived for intersected elements, but this time the interpolation functions will be given by Equation (4.17):

$$W_i^h M_{ij}^* \frac{\partial \mathbf{U}_j^h}{\partial t} + W_i^h C_{ijk}^* \mathbf{F}_k\left(\mathbf{U}_j^h\right) + W_i^h \mathbf{\Phi}_i^* = W_i^h M_{ij}^* \mathbf{S}\left(\mathbf{U}_j^h\right) \tag{4.19}$$

where:

$$M_{ij}^{*} = \int_{\Omega_e} N_i^{*} N_j^{*} d\Omega \tag{4.20}$$

$$C_{ijk}^{*} = \int_{\Omega_e} \partial_k N_i^{*} N_j^{*} d\Omega \tag{4.21}$$

$$\mathbf{\Phi}_i^{*} = \int_{\partial\Omega_e} N_i^{*} \mathbf{F}_{\mathbf{n}}^{-} \left(\mathbf{U}_{out}\right) d\Gamma - \int_{\partial\Omega_e} N_i^{*} N_j^{*} \mathbf{F}_{\mathbf{n}}^{-} \left(\mathbf{U}_j^{h}\right) d\Gamma \tag{4.22}$$

with $i, j \in \cup_{e \in \mathcal{N}_i} nod_e$.

4.3.2.1.1. Mass and divergence matrices

The most significant difference compared to the stabilization strategies addressed so far lies in the fact that the matrix $\mathbf{M}^{*}$ couples the degrees of freedom of all neighboring elements extrapolated over the unstable domain. To some extent, the local nature of the solution is lost, as we will work with patches of elements formed by unstable elements and their stable neighbors. The actual size of the patches will grow since a stable element can extend to stabilize the solution in several neighboring domains. Each patch will consist of the set of all degrees of freedom associated with shape functions whose support overlaps. More precisely, a patch of elements $\mathcal{E}_s$ will be defined as $\{i : \exists j \in \mathcal{E}_s \setminus i \text{ s.t. } \Omega_i^{st} \cap \Omega_j^{st} \neq \emptyset\}$. The most unfavorable situation would be one in which all the degrees of freedom that define the solution at the domain boundary are coupled. In any case, the system's dimension that would need to be solved would be much smaller than if a continuous Galerkin formulation were used since the problem being solved will encompass only the boundary. Another difference we find with respect to the couplings that appear in the continuous formulation is that the stabilized mass matrices are much denser. The related degrees of freedom will not only be those located on the mesh interfaces but all those associated with extended elements. Moreover, couplings between neighboring elements will never appear; instead, there will always be an intermediate domain assembling them, the unstable domain.

The divergence matrix $\mathbf{C}_k^{*}$ will also be computed considering the couplings between degrees of freedom associated with a patch of elements $\mathcal{E}_s$. Furthermore, the evaluation of spatial derivatives must consider the factor ϕ introduced in the definition of N^{*}. To perform the calculation, it is convenient to reformulate the distance function $d_{E_i}(\mathbf{x})$ for a point $\mathbf{x}$ located in the domain associated with the unstable element E_j as:

$$d_{E_j}(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_{i,j}| = (\mathbf{x} - \mathbf{x}_{i,j}) \cdot \mathbf{v}_{i,j} \tag{4.23}$$

where $\mathbf{x}_{i,j}$ are the coordinates of the point closest to $\mathbf{x}$ on the boundary of E_i, and $\mathbf{v}_{i,j} = \text{sign}\,(\mathbf{x} - \mathbf{x}_{i,j})$, where the function used is:

$$\text{sign}\,(\mu) = \begin{cases} +1 & if \quad \mu > 0 \\ 0 & if \quad \mu = 0 \\ -1 & if \quad \mu < 0 \end{cases}$$

By introducing this new formulation of distance into Equation (4.15), we obtain the derivatives of the interpolation functions outside the element boundaries necessary for the computation of the divergence matrix:

$$\partial_k N_l^*(\mathbf{x}) \;= \phi_j(\mathbf{x})\partial_k N_l(\mathbf{x}) + \phi_j(\mathbf{x})^2 N_l(\mathbf{x}) \sum_{m\in\{\mathcal{N}_i\setminus j\}} \frac{[\mathbf{v}_{i,j}]_k}{d^2_{E_m}(\mathbf{x})} \tag{4.24}$$

4.3.2.1.2. Fluxes through the boundaries of the integration domain

Regarding the fluxes, the procedure to calculate the integrals over the physical domain boundary will be the same as the one used to evaluate these quantities in extrapolated elements. The only difference lies in the fact that the definition of the shape functions will be different. The internal interfaces require a more detailed study. Following the reasoning presented in Section 3.1, the flux through each boundary will be calculated as:

$$\mathbf{\Phi}_i^* = \int_{\Omega_1\cap\Omega_2} N_i^* \left(N_k^* \mathbf{F}_\mathbf{n}^- \left(\mathbf{U}_k^h\right) - N_j^* \mathbf{F}_\mathbf{n}^- \left(\mathbf{U}_j^h\right)\right) d\Gamma \tag{4.25}$$

where Ω_1, Ω_2 are the domains associated with the unstable element and its neighbor, respectively; and the indices $\{i, j, k\}$ will be associated with their degrees of freedom. If E_2 is a stable element, the shape functions that will be used will be those of the Lagrangian element $N_k^* = N_k$. On the other hand, if E_2 is stable, this will imply that its approximation basis extends to Ω_1 multiplied by a factor, which is the weighting function ϕ_2. The jump at the interface will be directly related to the value of ϕ_2 and the discrepancies with the rest of the elements supported in Ω_1. By definition, ϕ_2 is unitary on the boundary that connects the domains Ω_1 and Ω_2 and null for the rest of the extended elements. This means that the solution on the boundary $\Omega_1 \cap \Omega_2$ as seen from E_1 will depend solely on the approximation defined for E_2; that is, $N_i^* = N_i$ and $N_j^* = N_j$. Therefore, in these situations, the original formulation is recovered:

$$\mathbf{\Phi}_i^* = \int_{\Omega_1\cap\Omega_2} N_i \left(N_k \mathbf{F}_\mathbf{n}^- \left(\mathbf{U}_k^h\right) - N_j \mathbf{F}_\mathbf{n}^- \left(\mathbf{U}_j^h\right)\right) d\Gamma \tag{4.26}$$

Additionally, we have seen that $\{i, j, k\}$ will be the indices of degrees of freedom associated with E_2, which means that the solution on both sides of the interface is defined from the same nodal values and interpolation functions. Therefore, the second factor of the integral will be zero throughout the integration domain, and there will be no flux $\mathbf{\Phi}_i^*$ through this boundary:

$$\mathbf{\Phi}_i^* = \mathbf{0} \quad if \quad \Omega_1 \subset \Omega_2^* \tag{4.27}$$

If we follow the reasoning, we will conclude that E_1 will exhibit non-zero fluxes only at those interfaces that connect it with another element E_2 within the group of unstable elements. In this case, no simplification could be made regarding Equation (4.25), and the calculation would have to be carried out considering all the degrees of freedom of the elements extended over Ω_1 and Ω_2, respectively. We can only speed up the calculations by neglecting the contribution of the degrees of freedom that are not on the plane of the interface but belong to extended elements whose face lies on said plane. When using tensor elements, we know that the shape functions associated with those degrees of freedom will be zero for any point located on the interface.

The integrals of the weak form are evaluated using the strategies described in Section 4.2. However, the effect of the new approximation bases used in the unstable domains must be evaluated. Rational bases can compromise the method's convergence if sufficiently rich quadratures are not used. The results of the studies conducted are presented in Section 4.4.

4.3.2.2. Implementation issues

4.3.2.2.1. Solving coupled systems of equations

For the solution of the systems within the time integration loop, the matrices $\mathbf{M}^*$ and their decomposition can be precomputed to speed up computation times. It is possible to recover the dimension of the elemental problem if, for example, the system is solved using the block Gauss-Seidel method. Studies have been conducted using this strategy, taking as an initial guess the solution of the system from the previous time step to reduce the number of iterations required for convergence. Convergence is achieved for a small number of iterations (no more than 10 iterations for $p = 3$ or 50 iterations for $p = 5$). However, the computation times increase substantially, and, at least in the studies conducted so far, the dimensions of the coupled systems never reach levels that could pose problems in terms of storage or in calculating the inverse matrix. For this reason, the choice is to solve the coupled systems using a direct solver.

4.3.2.2.2. Unstable neighborhood

It must be taken into account that not all unstable elements will have a neighboring element that can provide support for stabilization. The implementation considers this and accounts for several levels of vicinity for stabilization purposes. The level 1 vicinity of an element E_i consists of all elements that share a face with it; level 2 consists of those elements that are connected to E_i through an edge; finally, the elements that only share a vertex with E_i will constitute level 3.

The aim is always to stabilize the unstable element with its stable neighbors from the lowest level of vicinity. By applying this restriction, the support for the extended approximation remains continuous. Let's consider a uniform Cartesian mesh of level 1, as shown in Fig. 4.21, with its elements coded according to their virtual numbering $E_{i,j,k}$, where, for simplicity, we omit the coordinate corresponding to the level. Let's assume that $E_{2,2,2}$ is unstable. Initially, we will try to stabilize the discretization by extending the solution of elements $E_{1,2,2}$, $E_{2,1,2}$, and $E_{2,2,1}$, which are the level 1 neighbors of $E_{2,2,2}$. If all these elements are unstable, we would extend the search to the next level of vicinity and consider extending the solution in $E_{1,1,2}$, $E_{1,2,1}$, and $E_{2,1,1}$. Suppose $E_{1,1,2}$ is indeed a stable element. In this case, the solution from $E_{1,1,2}$ would extend to $E_{2,2,2}$ as well as to $E_{1,2,2}$ and $E_{2,1,2}$, since they are unstable elements in the level 1 vicinity of $E_{1,1,2}$. It can be seen that, with this procedure, an unstable element will always share the approximation with at least one of its level 1 neighboring elements.

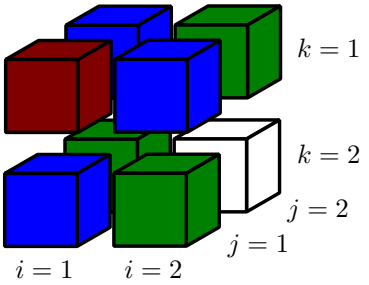


Figure 4.21: *Levels of vicinity:* level 0, level 1, level 2, level 3

The evaluation of the distance $d_{E_j}(x)$, present in Equation (4.11), will need to be modified to account for the possibility of higher-level neighbors. The general case will follow the formulation presented in Equation (4.23), where the distance is evaluated as the L1 norm of the vector connecting the point $\mathbf{x}$ with the nearest point on the boundary of E_j. Considering the alignment of the element faces with the Cartesian planes, evaluating this distance is trivial. By examining the last expression in Equation (4.23) to calculate the distances, we realize that the components of the projection that vary with $\mathbf{x}$ are the null components of $\mathbf{v}_{i,j}$. Therefore, for each pair of elements, a point $\mathbf{x}_{i,j}$ and a vector $\mathbf{v}_{i,j}$ can be defined, eliminating the need to evaluate the nearest point and reducing the calculation of distances to a vector subtraction and a dot product. For level 1 neighbors, it can be verified that the new definition of $d_{E_j}(x)$ is equivalent to the initial definition.

It is possible to encounter unstable elements that could not be resolved even after considering the highest levels of vicinity. However, this will indicate that the computational domain has slender geometric characteristics that require finer discretizations and shorter time steps. Therefore, this is not a limitation of the stabilization strategy but rather a poor choice of the approximation basis for the solution of the problem.

The user can use this indicator to adapt the computational mesh to the geometry of each particular problem.

We must not forget that we will be dealing with boundary elements, and thus, the continuity of the computational domain must be verified when determining whether one element is in the vicinity of another. Keeping this in mind, we will say that an element is in the vicinity of another when they contain regions of the computational domain that can be connected through the regions of the computational domain associated with elements of lower vicinity levels.

In Fig. 4.22, a cross-section of an extrusion geometry Ω is presented, and the vicinity of element $E_{x,1,1}$ is studied. If we only consider the mesh structure, elements $E_{x,2,1}$ and $E_{x,1,2}$ would form the level 1 vicinity of $E_{x,1,1}$. However, since $\Omega \cap E_{x,1,2}$ does not have continuity with $\Omega \cap E_{x,1,1}$, the element $E_{x,1,2}$ is not included in this level of vicinity. $E_{x,2,2}$ will be within the level 2 vicinity of $E_{x,1,1}$ because $\Omega \cap E_{x,2,2}$ and $\Omega \cap E_{x,1,1}$ can be connected through $\Omega \cap E_{x,2,1}$, and $E_{x,2,1}$ belongs to the level 1 vicinity of $E_{x,1,1}$. Following this criterion, $E_{x,1,2}$ could be included within a level 3 vicinity since $\Omega \cap E_{x,1,2}$ can be connected with $\Omega \cap E_{x,1,1}$ through the regions of Ω contained within $E_{x,2,1}$ and $E_{x,2,2}$, elements with level 1 and 2 vicinity, respectively.

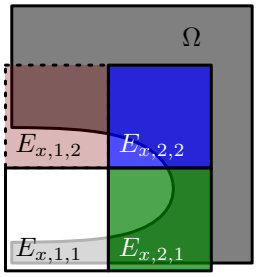


Figure 4.22: *Levels of vicinity regarding the continuity of the domain.*

4.3.2.2.3. Level restriction on unstable elements

The hypothesis has been established that unstable elements are always of a mesh level equal to or higher than their neighbors. This condition is necessary in order to apply (4.11) and the procedures described in this section. Let us imagine the case of an unstable element that shares one of its faces with four elements of a higher level. When we calculate the weighting of the solutions on said face, we will realize that we assign the same weight to the solution of the four neighbors, since the distance from the evaluation point to the boundary of the neighbors is zero in all cases. Therefore, the solution on the face, as seen from the unstable element, will be the average of the solutions of the neighboring elements. The higher-level elements will not have a unique solution on the face of the unstable element, and thus discrepancies will appear between the individual solutions and the average. These differences will be the origin

of fluxes that should be taken into account when formulating the weak formulation of the problem.

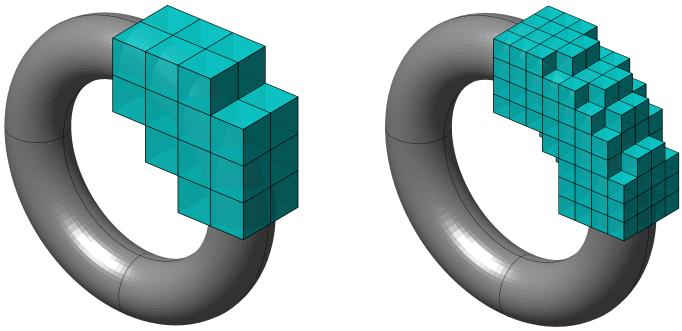
The restriction on the discretization does not imply any drawback, since the accuracy of the solution in the associated domain depends on the approximation used in the neighboring elements, and the stability sought is usually governed by the higher-level elements present in the mesh.

4.4. Numerical results

4.4.1. Numerical integration in cgDG 3D

In this first example, the performance of the integration techniques presented in this work will be compared: integration with transfinite mapping (TI), integration by decomposition into high-degree elements (pEI), and NURBS-enhanced pEI integration (NEI).

First, the differences between the integration meshes computed according to each of the proposed procedures will be evaluated. We will use 3 meshes $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ obtained from a cube with a side length of 2.55u. The geometry to be evaluated will be bounded by a toroidal surface with an inner diameter of 1.5u and an outer diameter of 2.5u. The surface is positioned so that its axis of symmetry coincides with one of the cube's edges, and its perpendicular symmetry plane intersects the edge at its center. The discretizations $\mathcal{M}_1$ and $\mathcal{M}_2$ will correspond to Cartesian meshes of levels 2 and 3, respectively, where the domain of interest is located inside the toroidal surface. Discretization $\mathcal{M}_3$ will also be a level 2 Cartesian mesh, but in this case, the domain of interest will be outside the surface. Fig. 4.23 shows the meshes to be used.



(a) *Meshes $\mathcal{M}_1$ and $\mathcal{M}_3$* ***(b)*** *Mesh $\mathcal{M}_2$*

Figure 4.23: *Problem domain and meshes*

First, we will study the necessity of using high-degree polynomial curves to approximate the intersections of the $\mathcal{M}_1$ mesh with the surfaces. Fig. 4.24 shows the approximations for different degrees of approximation. The discrepancies compared to the reference are visible for the lower degrees. In fact, for degree 2, there are difficulties in achieving convergence in the minimization of Equation (4.3). The use of large elements compared to the curvature of the geometry requires high-degree approximations. This situation will be common for high-degree discretizations such as those considered in this work.

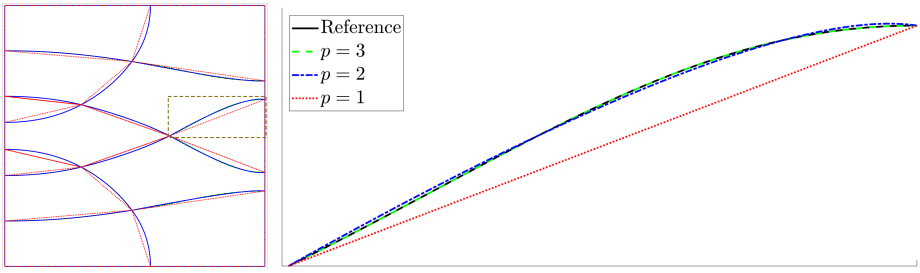


Figure 4.24: *General view and detail of intersections curves in parametric space.*

In Fig. 4.25, the subdomains on the geometry obtained for the TI and NEI procedures are compared using optimization- and elasticity-based approximations, respectively. In this representation, the $\mathcal{M}_2$ mesh is used, and the degree of the elements of the integration mesh is set to 3. We observe that there are significant differences in the results, although in both cases a decomposition without twisted subdomains is achieved. The strategy based on the elastic problem results in decompositions that are closer to the linear case, while the divisions obtained by solving the optimization problem show much greater deviations. Regarding computational cost, the structural approach is completed in 66% of the time needed to compute the optimization-based

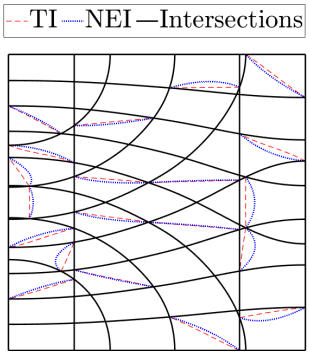


Figure 4.25: *Comparison of TI and NEI surface decomposition*

decomposition. The difference in processing speed should be taken as a preliminary result and may change as implementations are optimized.

Table 4.3 and Table 4.4 show the relative errors in the calculation of the area and volume, respectively, using the different integration strategies and approximation degrees from 1 to 3. In almost all cases, it is observed that the strategies that use the NURBS definition of the geometry improve the results obtained with the polynomial approximation. Only for volume calculation with degree 3 does pEI outperform TI. Regarding the area results, we can comment that the TI and NEI calculations are consistent. However, it is striking that accuracy decreases for the highest approximation degree. Further analysis is needed to determine the cause of this behavior but we suspect that changes in the definition within the NURBS may be causing these fluctuations. The current implementation does not consider the distribution of knots when defining the quadratures, which may be introducing an error that truncates the convergence of the integration. Although work is underway to overcome this limitation, the results we have obtained highlight the advantage of applying strategies such as TI and NEI.

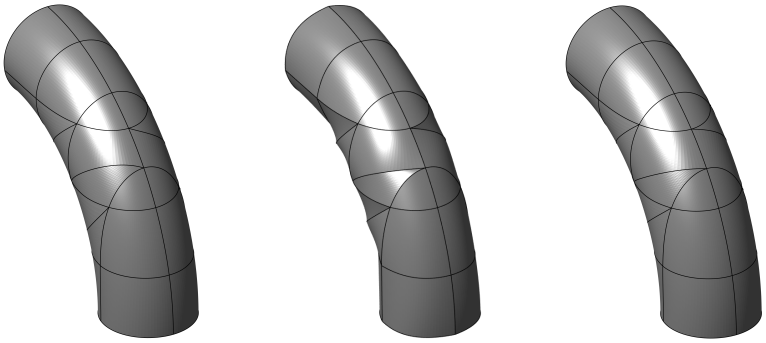
	$p=1$	$p=2$	$p=3$
TI		$3.76 \cdot 10^{-5}$	$1.10 \cdot 10^{-4}$
pEI	$8.36 \cdot 10^{-2}$	$4.45 \cdot 10^{-3}$	$3.99 \cdot 10^{-4}$
NEI	$1.10 \cdot 10^{-4}$	$3.96 \cdot 10^{-5}$	$8.67 \cdot 10^{-5}$

Table 4.3: *Relative error of area measure*

	$p=1$	$p=2$	$p=3$
TI		$9.06 \cdot 10^{-4}$	$7.49 \cdot 10^{-4}$
pEI	$3.74 \cdot 10^{-1}$	$2.38 \cdot 10^{-2}$	$1.35 \cdot 10^{-4}$
NEI	$5.53 \cdot 10^{-5}$	$9.43 \cdot 10^{-6}$	$2.07 \cdot 10^{-5}$

Table 4.4: *Relative error of volume measure*

Regarding the volume calculations, the TI and NEI results are once again very consistent, while the high-degree element-based strategies show a progressive reduction in error. This means that with TI and NEI, we are correctly capturing the problem domain regardless of the order used to construct the integration mesh. Finally, it is worth noting the improvement brought by enriching the polynomial subdomain-based integration with NURBS, as seen in the $p = 1$ case, where the error is reduced by 4 orders of magnitude. Fig. 4.26 visually demonstrates the differences in the domain representation depending on the integration strategy.

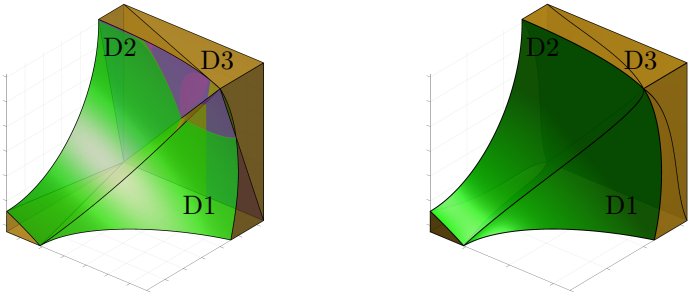


(a) TI strategy *(b) pEI strategy* *(c) NEI strategy*

Figure 4.26: *Three subdomains decompositions for integration purposes*

In light of these results, we conclude that the need to use high-order decompositions will be motivated by the need to accurately capture the interface between neighboring elements, rather than to correctly represent the domain of the problem. However, lower approximation orders will imply that the integration domains will not coincide with the internal portion of the intersected faces of the element. In certain cases, the lack of symmetry in the construction of the decomposition for neighboring elements will result in overlaps and gaps between the integration domains of neighboring elements. The boundary of the domain is accurately captured, regardless of the degree of decomposition, when using the TI and NEI techniques.

If we now study the configuration $\mathcal{M}_3$, we find that the TI strategy is not capable of achieving a configuration for the volume decomposition where there are no tangled subdomains. In Fig. 4.27, it can be seen that the TI strategy fails to resolve the decomposition, and we find that the subdomains D1 and D2 are twisted, with the green facets intersecting the violet facets. The same configuration processed with pEI gives us a valid decomposition. It can be seen how the internal facets of the domain have



(a) TI: Tangled subdomains *(b) pEI: Untangled subdomains*

Figure 4.27: *Comparison of curved subdomains starting from the same linear decomposition*

been deformed in such a way that the intersections between facets that appeared in the TI case are eliminated. A strategy needs to be developed to appropriately deform the internal facets and edges when applying the TI procedure. The current implementation behaves robustly in surface decomposition and provides valid decompositions for convex domains. However, it has been found to be insufficient when dealing with concave domains with high curvature. As a general rule, pEI or NE integrations will be used. The TI strategy will only be considered in cases where the analyzed domain has a smooth boundary.

Next, a study is carried out on the number of integration points needed to preserve the convergence of the method. To this end, the advection problem will be solved for the configuration $\mathcal{M}_1$. The advection of a unit-amplitude planar sinusoidal wave with a unit wavelength λ will be simulated. We will start with an initial field where the wave is developed throughout the domain, and Dirichlet boundary conditions will be imposed on the boundary. The problem statement is to find the mapping $U : \Omega \times [0, T] \to \mathbb{R}$ such that:

$$\begin{aligned} \partial_t U(\mathbf{x}, t) + \mathbf{b}\nabla U(\mathbf{x}, t) &= 0 && \mathbf{x} \in \Omega \\ U(\mathbf{x}, 0) &= U^0(\mathbf{x}) && \mathbf{x} \in \Omega, t = 0 \\ U(\mathbf{x}, t) &= U^D(\mathbf{x}, t) && \mathbf{x} \in \partial\Omega \end{aligned}$$

where $\mathbf{b}$ is the advection velocity, which in this problem will be unitary in the direction of the diagonal of the first octant, namely $\mathbf{b} = (1, 1, 1)/\sqrt{3}$. In Fig. 4.28, a schematic of the problem is shown where the computational domain and the wave propagation direction are represented.

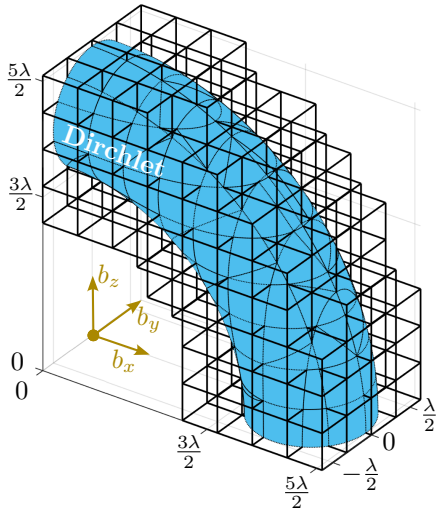


Figure 4.28: *Problem setup*

For the same stabilized Cartesian mesh for a volume fraction of 0.3 and using polynomial approximation degrees ranging from 2 to 4, the effect of the stabilization technique and the subdomain decomposition strategy on the number of integration points required and the solution accuracy will be studied. The same problem is solved by progressively increasing the number of integration points until three solutions with errors of the same order of magnitude are reached.

First, we will focus on the conditioning of the systems presented in the upper rows of Fig. 4.29, Fig. 4.30, and Fig. 4.31. In all cases, it is verified that the conditioning for the weighted extrapolation (WE) stabilization strategy is higher. No differences are observed between the strategies overlapped extrapolation with averaging (OE-Av) and overlapped extrapolation with interpolation (OE-In) since the coefficients of the systems being solved are identical. Additionally, for the degree 4 approximation, we observe that convergence toward the final value is slower for the WE strategy. On the other hand, we see that the integration strategy used has little influence on the conditioning of the systems. A slight increase in the condition number is only observed for the integration that enriches the decomposition into high-degree elements with the definition of the NURBS, which disappears as the number of integration points increases. In the solvable cases, no differences are noted. It should also be pointed out that the conditionings we obtain for degree 4 are quite poor and are at the limit of the values for which solving is reasonable.

Regarding the error in the final solution, represented in the lower rows of Fig. 4.29, Fig. 4.30, and Fig. 4.31, the influence of the integration strategy is imperceptible. The stabilization procedure does have significant effects on the results. On one hand, we observe that as we increase the degree of the base interpolation, a difference appears between the number of integration points required to solve with OE-Av and OE-In compared to WE. Solving using shape functions with rational weighting requires a greater number of integration points for odd approximation degrees, necessitating a 2-order increase in quadrature; while for even degrees, enriching the integration has not been necessary. The results also show that the accuracy of the solution not only depends on the discretization selected for the base mesh but is also closely related to the stabilization procedure employed. The least accurate results are obtained for the OE-Av strategy, and the solution improves as the richness of the approximation in the unstable domains is increased, with the lowest error obtained for the WE case. This effect is more clearly observed in Fig. 4.32, where the normalized errors obtained for the calculation performed using the richest integration have been represented. Discrepancies in the solution due to the stabilization strategy are greater as the degree increases. The results presented in Table 4.5 show that the gain obtained by using the OE-In strategy instead of OE-Av is comparable to that obtained by using the WE strategy instead of OE-In. For degree 4, errors of different orders of magnitude are obtained depending on the procedure used.

Unexpected behavior has been observed with some quadratures. In Fig. 4.29, the solution could not be obtained using a 7th-order integration, even though the resolution for 6th-order quadratures was possible. A similar behavior occurred for order 13 in Fig. 4.30 and Fig. 4.31. The same lack of consistency is observed in

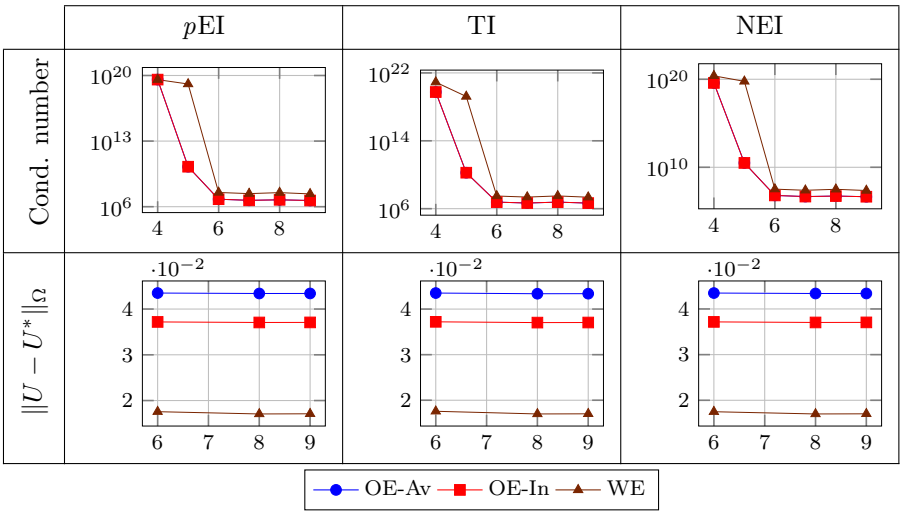


Figure 4.29: *Results of integration test for $p = 2$ and $\mathcal{M}_1$ w.r.t. integration order. High order cells (pEI), transfinite mapping (TI) and nurbs enhanced high order cells (NEI) integration strategies are compared.*

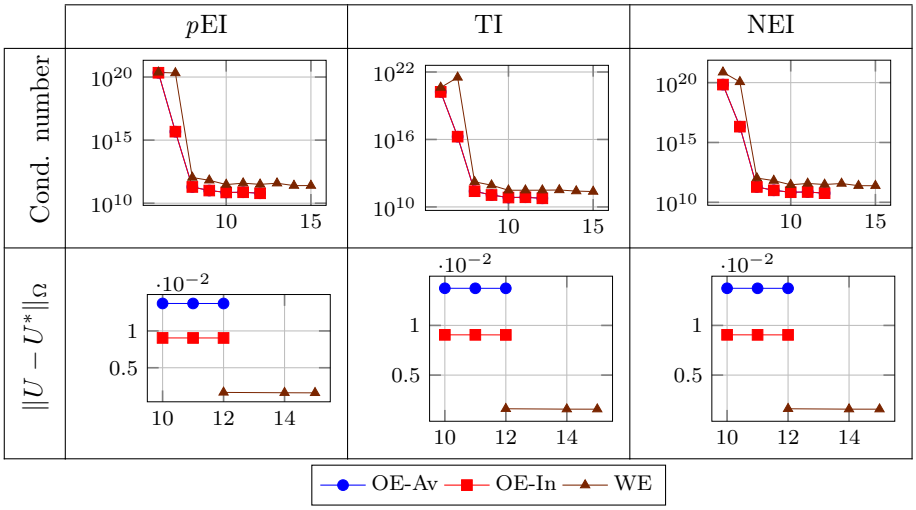


Figure 4.30: *Results of integration test for $p = 3$ and $\mathcal{M}_1$ w.r.t. integration order. High order cells (pEI), transfinite mapping (TI) and nurbs enhanced high order cells (NEI) integration strategies are compared.*

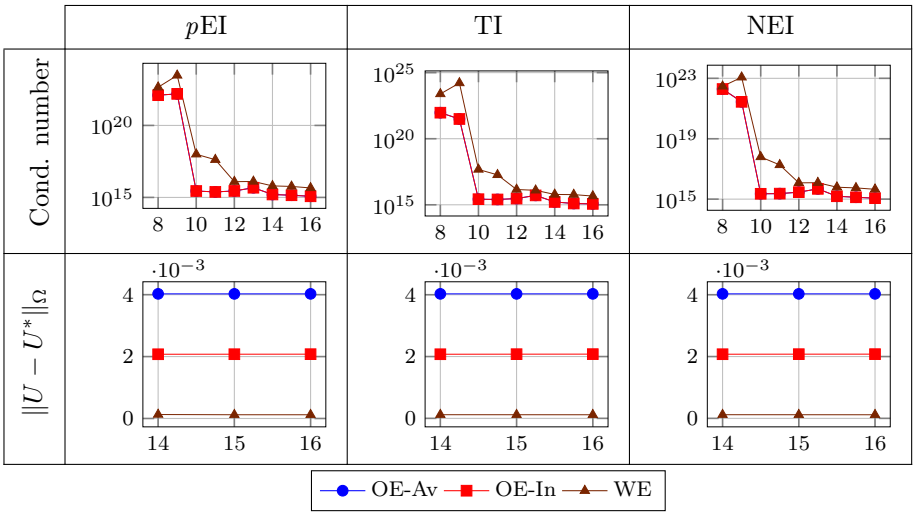


Figure 4.31: *Results of integration test for $p = 4$ and $\mathcal{M}_1$ w.r.t. integration order. High order cells (pEI), transfinite mapping (TI) and nurbs enhanced high order cells (NEI) integration strategies are compared.*

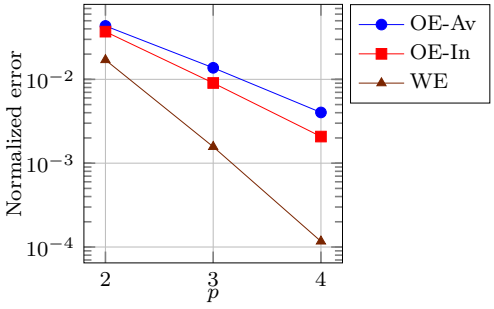


Figure 4.32: *Normalised error for the richest integration scheme*

p	2	3	4
OE-Av → OE-In	0.316540	0.244149	0.092106
OE-In → WE	0.293282	0.229593	0.074436

Table 4.5: *Error reduction factor regarding stabilisation strategy*

the condition number graphs, especially in Fig. 4.31, where the worst conditioning of the systems for these cases is evident. Further studies are needed, but we believe these results are due to issues with the precision of the integration quadratures we are using. The availability of quadratures also poses a limitation when using higher-degree approximations. In solving some electromagnetism problems, we have found that it is necessary to slightly increase the integration order compared to the values obtained in this study. Given that quadratures up to order 20 are provided in [155, 156], we find that the 5th-order approximation requires a number of integration points that exceeds this limit.

4.4.2. Performance of stabilization strategies

In this third example, the behavior of the various proposed stabilization techniques will be compared in terms of the maximum stable time step. For this, the propagation in the positive Z direction of a plane electromagnetic wave with X-polarized electric field will be solved. The domain we will consider is the parallelepiped shown in Fig. 4.33, which corresponds to a portion of a cube with side length equal to λ, lying between two planes with direction vectors $(\sin\pi/9, \cos\pi/9, 0)$ and $(0, \sin\pi/30, \cos\pi/30)$, offset by a distance of 0.4λ in the Y-axis direction. The problem will be solved by applying periodicity conditions on all faces of the geometry.

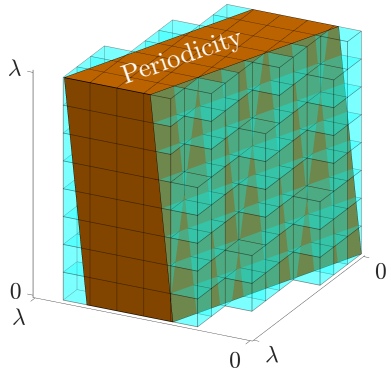


Figure 4.33: *Problem setup*

The approximation basis will be defined on a level 3 mesh constructed from a cubic reference domain with side length λ. This means that the mesh conforms to surfaces perpendicular to Cartesian directions. The inclined planes result in a set of intersected elements that exhibit the MC cutting patterns: 1, 2, 5, 8; as well as a range of volume fractions within the computational domain that varies from $2.15 \cdot 10^{-8}$ to $1-7\cdot10^{-6}$. The indicator we will use to measure the stability of each of the intersected domains will be precisely the volume fraction of the element that intersects with the computational domain α_e.

Uniform discretizations with interpolation degrees of 2, 3, and 4 will be studied. The solution will be advanced in time using the integrators proposed in [163], with the minimum number of steps that share the convergence order of the spatial discretization. The maximum stable time step for stabilized discretizations will be computed, taking into account a criterion for identifying unstable elements where the cut-off value for the parameter will be $\alpha_{th} = \{0.05, 0.1, 0.2, 0.4\}$. As a reference, the maximum stable time step obtained for a discretization of the same type without intersected elements will be used.

The results obtained are presented in Fig. 4.34. In them, we see that increasing the parameter α_{th} results in an increase in the maximum stable time step that can be used. If we compare the results for extreme values of α_{th}, we observe that the discretization with the highest stabilization can be solved using time steps 2.5 to 3 times larger than those used for the least stabilized case. It is also noteworthy that stabilization is somewhat more effective when using the WE strategy. Additionally, for all cases, we see a linear relationship between the cut volume fraction and the time step value, confirming the usefulness of this indicator as an estimator of the stability of the discretization.

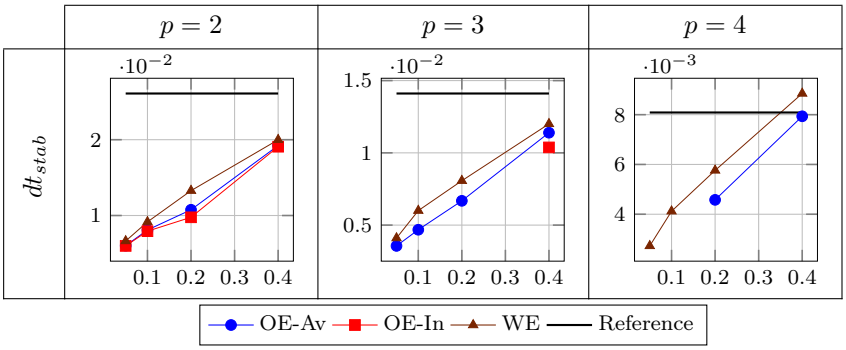


Figure 4.34: *Maximum stable time step w.r.t. the stability threshold (α_{th}).*

When applying stabilization techniques, we not only obtain advantages from the point of view of a stable time step, but also in terms of the conditioning of the systems. In Fig. 4.35, the condition numbers of the worst-conditioned systems for each discretization are shown and compared with the results for the full element. We observe how the conditioning of the systems improves substantially when eliminating elements with very small volume fractions, reducing the condition numbers by several orders of magnitude. The WE technique again shows better performance than the OE, resulting in better-conditioned systems. This stabilization effect is especially relevant for higher orders. For the case with a fourth-order approximation, we see that the condition numbers for poorly stabilized discretizations are above 10^{18} for WE stabilization. In fact, the lower stabilization provided by the OE techniques makes

the conditioning so poor that small numerical errors cause the resolution to become unstable.

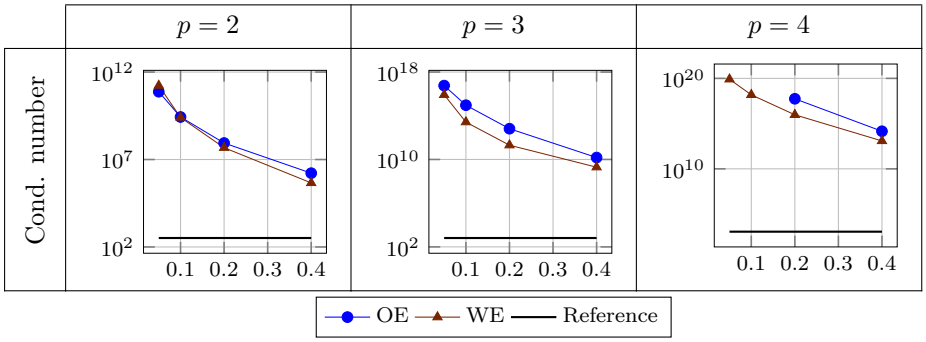


Figure 4.35: *Maximum condition number for stabilised schemes w.r.t. the stability threshold (α_{th}).*

No results could be obtained for the OE-In case and approximation degrees 3 and 4. The stabilization strategy has been used to solve other problems with very promising results, which leads us to think that it may be due to some failure in the implementation of this particular problem. Further studies are needed to determine the source of the instabilities that are occurring in this configuration.

4.4.3. Scattering of PEC sphere

In this final example, we will use the cgDG to solve the scattering of a plane wave incident on a sphere. This problem is relevant because an analytical solution is available for the quantity of interest that is usually the objective of these analyses, the radar cross section [107]. For this reason, it is a benchmark that is commonly used to validate models before applying them to problems in an industrial context.

The configuration we are going to analyze is presented in Fig. 4.36. The scatterer will be a sphere of radius $\lambda/2$, upon which a plane wave with electric field polarization in the X direction propagating along the negative Z-axis is incident.

We consider a cubic computational domain with a side length of 2λ, discretized with a level 4 mesh and elements with an interpolation degree of 4. The discretization will be stabilized using a weighted overlap strategy based on distance, using a stabilization cutoff parameter value of 0.4. The time step is set sufficiently low to ensure the stability of the method, and the simulation is run over time until the radar cross section converges with a relative tolerance of 10^{-4}.

Regarding boundary conditions, perfect electric conductor behavior is imposed on the surface of the sphere, while an absorbing boundary condition is imposed on the outer boundary of the cubic domain. To more accurately represent the behavior of an infinite domain and to prevent wave reflection at the outer boundary, a perfectly

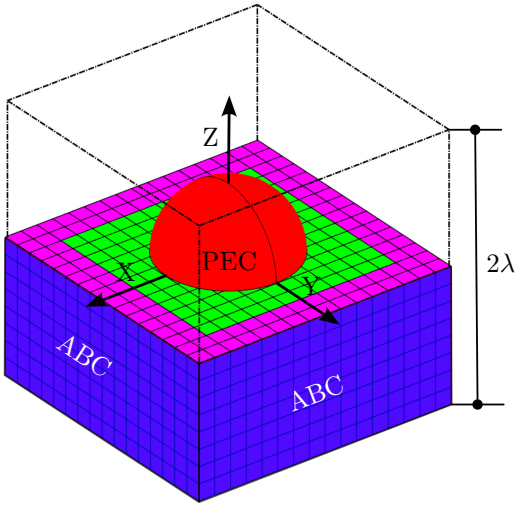
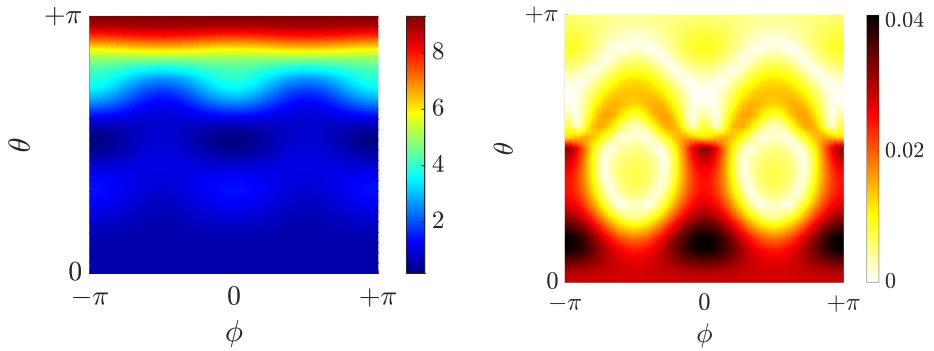


Figure 4.36: *Problem setup*

matched layer (PML) of thickness $\lambda/4$ has been introduced. Specifically, the PML proposed in [115] has been implemented, which constitutes an improvement over the Berenger PML with electromagnetic field splitting [109] used in the 2D case. The proposal in [115] is based on the complex coordinate-stretching technique. This technique uses a coordinate transformation with two governing parameters: the scaling coefficient a and the attenuation coefficients ω. In this implementation, the scaling coefficient is set to one, and the attenuation coefficient grows quadratically from 0 at the inner boundary of the PML to 20 at the outer boundary.

In Fig. 4.37, the results obtained for the steady-state regime of the problem are shown. In Fig. 4.37a, the bistatic radar cross section obtained in the simulation is represented, where ϕ and θ represent the polar coordinates of the measurement point with respect to the direction opposite to the wave propagation direction. Thus, the value of the monostatic RCS would be obtained at the point $\{0, 0\}$. In Fig. 4.37b, the relative errors of the RCS compared to the analytical solution are presented. A good correspondence is observed between the results, with errors below 4% in practically all points. Furthermore, the highest values are located in those areas with lower magnitudes, which could be due to a lack of convergence in the solution.

In Figs. 4.38 and 4.39, three views of the reflected electric field and reflected magnetic field components, respectively, up to time $t = 4T$ are shown on the surface of the scatterer and the three Cartesian planes passing through the center of the sphere. If we analyze the results, we can observe how the PML mitigates nearly all of the wave before it reaches the outer boundary. No artificial oscillations in the internal domain due to reflections at the absorbing boundary are observed. Nor is any artifact introduced by the stabilization and post-processing strategies of the



(a) Radar cross section (dB) *(b) Relative error of the RCS*

Figure 4.37: *Results in the stationary regime for the PEC sphere fo diameter* λ

approximate solution; the representation over the geometry is smooth and within the expected value range. This behavior is maintained throughout the entire simulation, which ends at $t = 9T$ due to the convergence of the RCS.

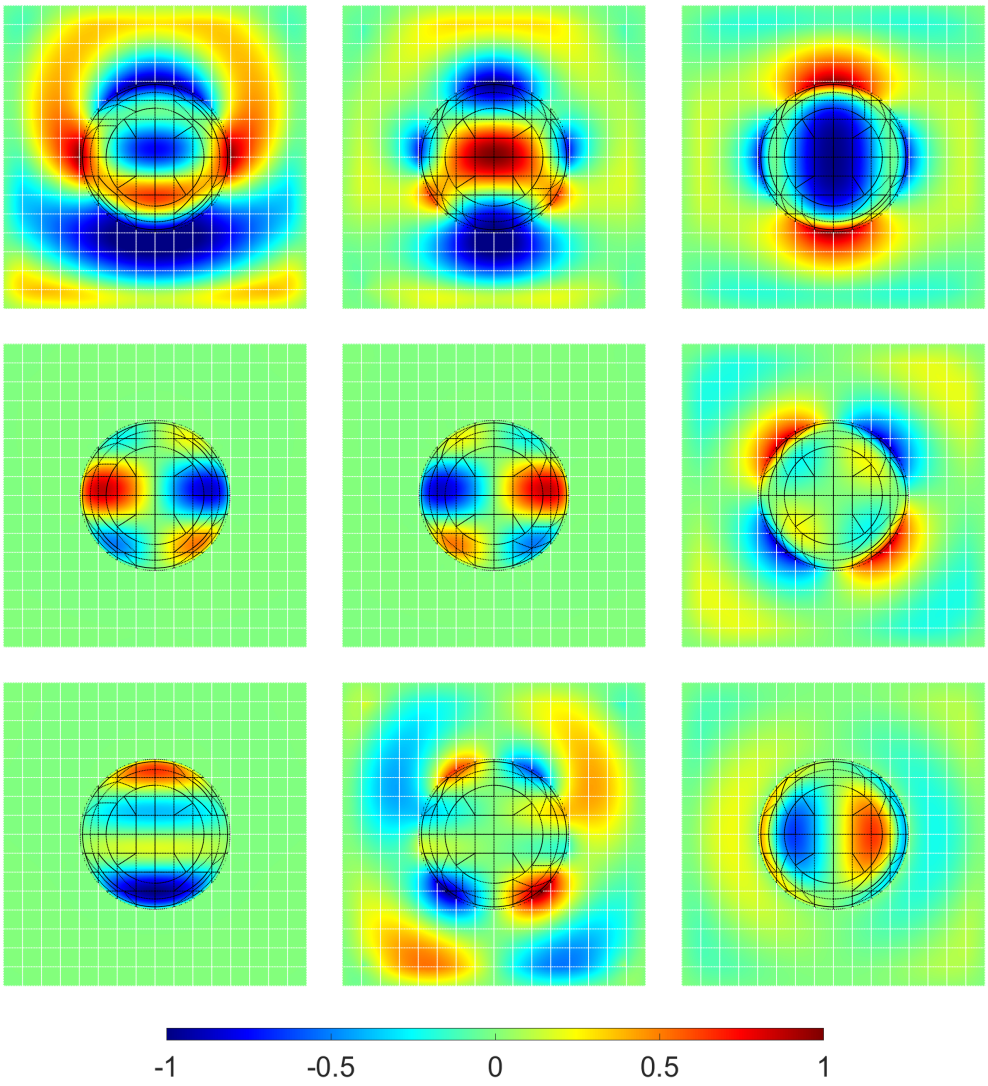


Figure 4.38: *Values of the three components of the scattered electric field (from top to bottom) on the surface of the sphere and the Cartesian planes YZ, XZ, XY (from left to right) (click on the figures to show the animation, available when using Adobe Acrobat).*

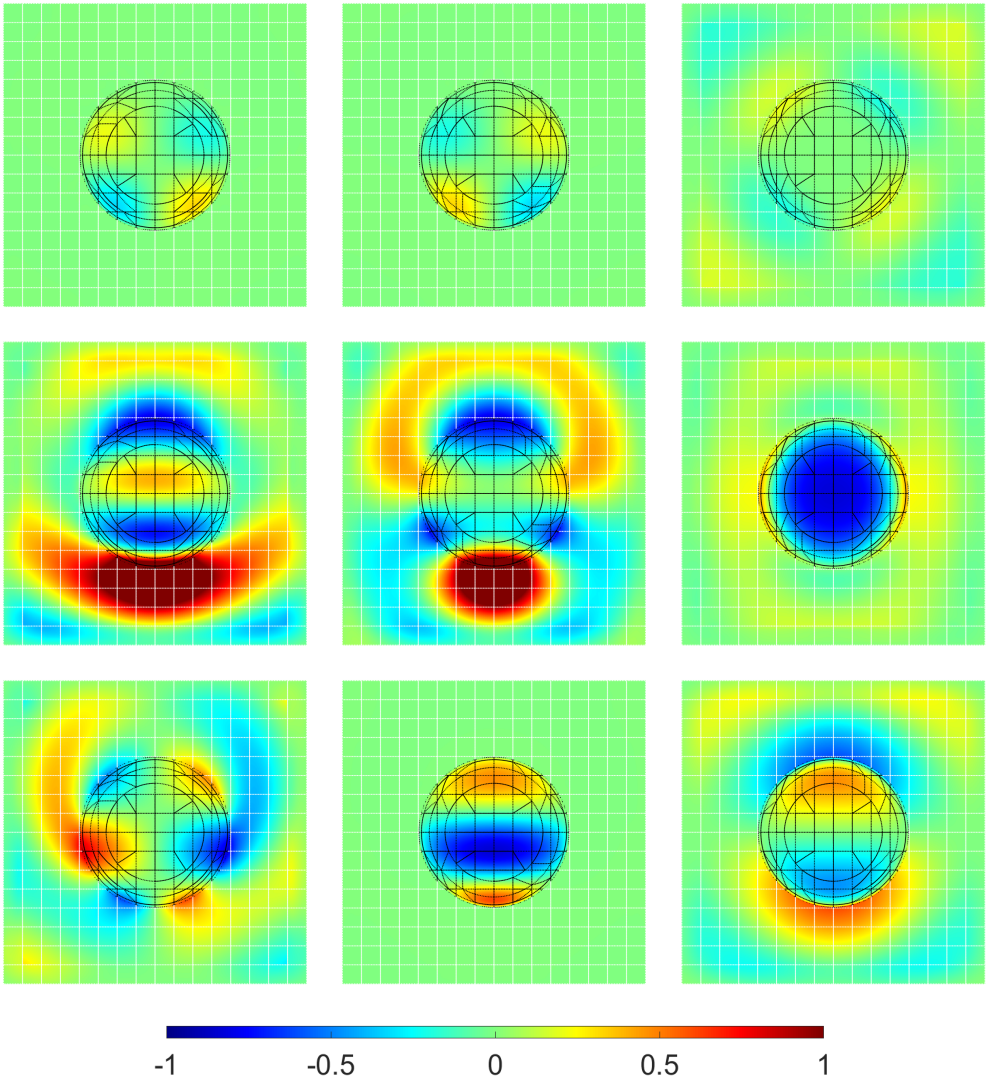


Figure 4.39: *Values of the three components of the scattered magnetic field (from top to bottom) on the surface of the sphere and the Cartesian planes YZ, XZ, XY (from left to right) (click on the figures to show the animation, available when using Adobe Acrobat).*

Chapter 5

hp-Adaptive Strategies in cgDG

In Sections 3.6.1.3 and 3.6.3.1, the necessity of applying mesh adaptation strategies to fully exploit the capabilities of cgDG became evident. In those analyses, active intervention in the discretization design allowed obtaining significantly better results compared to using uniform meshes. Moreover, as is widely recognized and confirmed in those studies, optimal modifications do not always follow the same strategy; depending on the conditions, increasing the approximation degree (p-refinement) may be more effective than reducing the element size (h-refinement), or vice versa.

This last chapter initiates a research line motivated by those results. It presents preliminary studies oriented towards automatic hp-mesh adaptation in the cgDG context. These studies cover all stages in the construction and evaluation of the adaptation process and have resulted in a numerical scheme offering promising outcomes. In Section 5.1, the desired characteristics of the adaptation procedure are analyzed, commenting on some relevant aspects of our proposal. The core of this chapter is in Sections 5.2 and 5.3, where we present the local error measures used to guide the adaptation process, as well as criteria for discerning the most suitable type of refinement for each case. The structure of the complete process, from an initial discretization to its adapted version, is detailed in Section 5.4. Section 5.5 addresses the transfer of fields between meshes, proposing a new projection strategy. Some preliminary studies evaluating the performance of the proposed techniques are collected in Section 5.6.

5.1. Towards automatic hp-adaptive cgDG

This thesis focuses on developing techniques for simulating transient problems governed by hyperbolic PDEs, using the electromagnetic problem as a reference. Such mathematical models are employed to characterize phenomena in which information propagates at a finite speed. From the simulation standpoint with cgDG, this suggests that uniform and invariant discretizations only make sense when the fields being approximated are at least stationary and information is uniformly distributed across the domain.

When the field exhibits a local feature, such as a discontinuity in the medium or a jump induced by a shock wave, the discretization must be enriched locally to better capture its shape. Local features originating from medium properties were already addressed in Section 3.6.3.1 and were managed by involving the user in mesh design. In stationary analyses like those, an experienced analyst could define a discretization appropriate throughout the entire analysis, as the field characteristics are local but stationary. Handling shock waves is significantly more challenging, as the singularity propagates with the field itself, necessitating an evolving discretization whose degrees of freedom replicate this transport phenomenon. Designing such meshes is complex and demands substantial resources.

In both scenarios, the limiting factor is the analyst's time. Accurate results require an experienced user who, drawing on their knowledge of both the problem and the numerical method, deploy significant resources in designing suitable computational meshes for each specific configuration. This guides the refinement strategy design for dgDG. The **challenge** posed is: **to build a general-purpose automatic technology operating in parallel with the time integration loop, locally adapting the calculation mesh to the specific field characteristics at each time step, aiming to minimize the error of the approximate solution, and adhering to simple guidelines set by the user.** The following questions have guided the development of the proposed strategy:

When? The mesh should continuously adapt to the field characteristics being approximated. Thus, adaptation technology should apply after each solution update throughout the entire analysis. However, it was decided to allow users to define adaptation start time μ_{t0}, adaptation end time μ_{tf}, and intervals between adaptations $\mu_{\Delta t}$, enabling resource management.

Where? The adaptation strategy aims to contain the discretization error. Achieving this requires either a local error measure or, at a minimum, insight into the error's spatial distribution over the mesh. Sections 5.2.1 and 5.3.2 introduce two strategies for obtaining this information, between which the user can choose via the parameter $\mu_{\mathcal{E}}$. Based on the selected measure, mesh elements can

be ranked and processed accordingly.

How? According to literature, the most efficient meshes primarily use high-order elements spanning relatively large domain areas. Smaller, lower-order elements are preferable only in regions where the solution is less smooth. Therefore, an effective *hp*-adaptation strategy must locally assess field regularity and act accordingly. Tools for this purpose are presented in Sections 5.2.2 and 5.3.5. Furthermore, user control is often desirable, either to bias the process to investigate specific phenomena or to leverage of prior knowledge about the field. The user-defined parameter $\mu_{\mathfrak{r}}$ is introduced to accommodate this flexibility

How much? Ideally, arbitrarily fine meshes would be employed to achieve maximum accuracy. However, computational costs increase substantially with mesh refinement due to the growing number of degrees of freedom and the corresponding reduction in the maximum stable time step. In practice, limitations in memory and computation time constrain simulations, making it necessary to seek the most accurate solution achievable within the available resources. If $\mathbb{S}$ denotes the set of all admissible discretizations —those that ensure bounded computational costs and stability— then the objective becomes identifying the element of $\mathbb{S}$ that minimizes the expected error. The parameter $\mu_{\mathfrak{o}}$ is also included to enable user control over this error minimization process.

Remark. The mesh adaptation can be addressed by individually selecting and processing elements based on their error level. However, grouping elements with similar error measures and processing them simultaneously has been found to be preferable. This promotes adapted meshes that reproduce solution patterns and prevents adaptation from inducing an uneven error distribution.

Remark. The set of valid meshes $\mathbb{S}$ can be expanded in the context of non-uniform meshes. Typically, elements that condition the temporal integrator's step are a minority in the global mesh. These elements are usually included to capture local field features requiring greater approximation richness, while the degrees-of-freedom density remains significantly lower elsewhere in the domain. Explicit temporal integrators with local time stepping [18, 19, 20] offer a preferable structure in these scenarios, as they apply smaller time steps only in those mesh regions where genuinely necessary. These strategies have been implemented in cgDG for their advantageous combination with mesh adaptation technology.

In Table 5.1, the inputs defining a specific configuration of the proposed *hp*-mesh adaptation strategy and their relevance within the procedure are summarized. Through these inputs, users can easily modify the characteristics of all resulting adapted meshes, adjusting them to their needs. The precise involvement of each input in the adaptation process will be detailed in Section 5.4 while their impact on the outputs of the process is assessed in Section 5.6.

Input	Relevance
μ_{t0}	Parameter controlling the simulation time when the mesh adaptation strategy starts to apply
μ_{tf}	Parameter controlling the simulation time when the last mesh adaptation occurs
$\mu_{\Delta t}$	Parameter controlling the interval between executions of the mesh adaptation routine
$\mu_{\mathcal{E}}$	Parameter determining the strategy used to measure approximation error and field smoothness
$\mu_{\mathfrak{r}}$	Parameter influencing the *hp*-refinement criteria, with higher values favoring *p*-refinement
$\mathbb{S}$	Set of valid discretizations for simulation. It defines the minimum μ_l and maximum μ_u number of nodes permitted in a mesh.
$\mu_{\mathfrak{o}}$	Parameter influencing the stopping criterion for mesh adaptation at a specific instant, with higher values favoring more significant variations.

Table 5.1: *User inputs governing the hp-adaptation procedure*

5.2. Explicit indicators

5.2.1. Error indicator

The development of the explicit error indicator is based on the DG formulation presented in Section 2.2. We recall equation (2.36), which represents the weak form posed over a domain Ω of Maxwell's equations written as conservation laws:

$$\int_{\Omega} \mathbf{W} \cdot \frac{\partial \mathbf{U}}{\partial t} d\Omega + \int_{\Omega} \mathbf{W} \cdot \frac{\partial \mathbf{F}_k(\mathbf{U})}{\partial x_k} d\Omega + \int_{\partial\Omega} \mathbf{W} \cdot [\![\mathbf{F}_{\mathbf{n}}^{-}(\mathbf{U})]\!] d\Gamma = \int_{\Omega} \mathbf{W} \cdot \mathbf{S}(\mathbf{U})\, d\Omega \quad (5.1)$$

Reordering and grouping the terms in this expression, we obtain:

$$\int_{\Omega} \mathbf{W} \cdot \mathbf{R}_i(\mathbf{U}) d\Omega + \int_{\partial\Omega} \mathbf{W} \cdot \mathbf{R}_d(\mathbf{n}, \mathbf{U}) d\Gamma = 0 \quad (5.2)$$

where $\mathbf{R}_i(\mathbf{U})$ and $\mathbf{R}_d(\mathbf{n}, \mathbf{U})$ are the local internal residual and the local residual of the flux in the negative direction of $\mathbf{n}$, respectively:

$$\mathbf{R}_i(\mathbf{U}) = \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_k(\mathbf{U})}{\partial x_k} - \mathbf{S}(\mathbf{U}) \tag{5.3}$$

$$\mathbf{R}_d(\mathbf{n}, \mathbf{U}) = \mathbf{F}^-_{\mathbf{n}}(\mathbf{U}_{\text{out}}) - \mathbf{F}^-_{\mathbf{n}}(\mathbf{U}) \tag{5.4}$$

which define the bilinear form $\mathcal{A}(\mathbf{W}, \mathbf{U})$:

$$\mathcal{A}(\mathbf{W}, \mathbf{U}) = \int_\Omega \mathbf{W} \cdot \mathbf{R}_i(\mathbf{U}) d\Omega + \int_{\partial\Omega} \mathbf{W} \cdot \mathbf{R}_d(\mathbf{n}, \mathbf{U}) d\Gamma \tag{5.5}$$

By definition, the exact solution $\mathbf{U}$ satisfies:

$$\mathbf{R}_i(\mathbf{U}) = \mathbf{0} \tag{5.6}$$

$$\mathbf{R}_d(\mathbf{n}, \mathbf{U}) = \mathbf{0} \tag{5.7}$$

and for every field $\mathbf{W}$ in $[H^1(\Omega)]^3$, it holds that:

$$\mathcal{A}(\mathbf{W}, \mathbf{U}) = 0 \tag{5.8}$$

Let us identify with the superscript h the fields contained in the subspace of functions $\mathbf{V}^h$ considered when solving the problem approximately using the DG formulation, and define the approximation error $\Delta\mathbf{Z}$ of an analytic field $\mathbf{Z}$ as $\Delta\mathbf{Z} = \mathbf{Z} - \mathbf{Z}^h$. By construction, the approximate solution field $\mathbf{U}^h$ satisfies:

Let us denote by $\mathbf{V}^h$ the subspace of functions considered in the computation of approximate solutions using the DG formulation, and use the superscript h to indicate fields belonging to this subspace. Next, we define the approximation error $\Delta\mathbf{Z}$ of an analytic field $\mathbf{Z}$ as $\Delta\mathbf{Z} = \mathbf{Z} - \mathbf{Z}^h$. By construction, the approximate solution field $\mathbf{U}^h$ satisfies:

$$\mathcal{A}(\mathbf{W}^h, \mathbf{U}^h) = 0 \tag{5.9}$$

Restricting $\mathbf{W}$ in equation eq. (5.8) to $\mathbf{V}^h$ and combining the result with eq. (5.9), we obtain the orthogonality property of the DG solution with respect to the operator $\mathcal{A}$:

$$\mathcal{A}(\mathbf{W}^h, \Delta\mathbf{U}^h) = 0 \tag{5.10}$$

Substituting the weight functions by the solution field and combining results (5.8), (5.9), and (5.10), we obtain:

$$\mathcal{A}(\Delta\mathbf{U}^h, \Delta\mathbf{U}^h) = -\mathcal{A}(\Delta\mathbf{U}^h, \mathbf{U}^h) \tag{5.11}$$

Introducing the definition of the bilinear form and applying the Cauchy–Schwarz inequality, we obtain:

$$|\mathcal{A}(\Delta\mathbf{U}^h, \Delta\mathbf{U}^h)| \leq \|\Delta\mathbf{U}^h\|_\Omega \|\mathbf{R}_i(\mathbf{U}^h)\|_\Omega + \|\Delta\mathbf{U}^h\|_{\partial\Omega} \|\mathbf{R}_d(\mathbf{n}, \mathbf{U}^h)\|_{\partial\Omega} \tag{5.12}$$

where $\|\cdot\|_\Omega$ denotes the L^2 vector norm evaluated over the domain Ω.

Let us assume that Ω is a locally Lipschitz domain. The second term in the inequality involves the trace norm of the error, which is bounded using the trace inequality together with the Poincaré–Friedrichs inequality:

$$\|\Delta\mathbf{U}^h\|_{\partial\Omega} \leq C_{T0} \|\Delta\mathbf{U}^h\|_\Omega + C_{T1} \left\|\nabla\left(\Delta\mathbf{U}^h\right)\right\|_\Omega \tag{5.13}$$

$$\leq \left(C_{T0} C_{PF} + C_{T1}\right) \left\|\nabla\left(\Delta\mathbf{U}^h\right)\right\|_\Omega \tag{5.14}$$

Substituting this result into (5.12) and applying again the Poincaré–Friedrichs inequality to the first term in (5.12), we obtain:

$$|\mathcal{A}(\Delta\mathbf{U}^h, \Delta\mathbf{U}^h)| \leq \left[C_{PF} \|\mathbf{R}_i(\mathbf{U}^h)\|_\Omega + \left(C_{T0} C_{PF} + C_{T1}\right) \|\mathbf{R}_d(\mathbf{n}, \mathbf{U}^h)\|_{\partial\Omega}\right] \left\|\nabla\left(\Delta\mathbf{U}^h\right)\right\|_\Omega \tag{5.15}$$

On the other hand, when the exact solution is smooth within Ω, we know from (5.6) and (5.7) that the operators $\mathbf{R}_i$ and $\mathbf{R}_d$ applied to the error are bounded as follows:

$$\|\mathbf{R}_i(\mathbf{U}^h)\|_\Omega = \|\mathbf{R}_i(\Delta\mathbf{U}^h)\|_\Omega = \|\mathbf{R}_i\| \|\Delta\mathbf{U}^h\|_\Omega \tag{5.16}$$

$$\|\mathbf{R}_d(\mathbf{U}^h)\|_{\partial\Omega} = \|\mathbf{R}_d(\Delta\mathbf{U}^h)\|_{\partial\Omega} = \|\mathbf{R}_d\| \|\Delta\mathbf{U}^h\|_{\partial\Omega} \tag{5.17}$$

Starting from the bilinear form applied to the error, considering the bounds of the residuals, and applying the trace and Poincaré–Friedrichs inequalities, we obtain:

$$|\mathcal{A}(\Delta\mathbf{U}^h, \Delta\mathbf{U}^h)| \leq \|\mathbf{R}_i\| \left\|\Delta\mathbf{U}^h\right\|_\Omega^2 + \|\mathbf{R}_d\| \left\|\Delta\mathbf{U}^h\right\|_{\partial\Omega}^2 \tag{5.18}$$

$$\leq c\rho^{\frac{1}{2}} \left(\left\|\Delta\mathbf{U}^h\right\|_\Omega^2 + \left\|\nabla\left(\Delta\mathbf{U}^h\right)\right\|_\Omega^2\right) \tag{5.19}$$

$$\leq c\rho^{\frac{1}{2}} \left(C_{PF} + \frac{1}{C_{PF}}\right) \left\|\Delta\mathbf{U}^h\right\|_\Omega \left\|\nabla\left(\Delta\mathbf{U}^h\right)\right\|_\Omega \tag{5.20}$$

where it is assumed that the gradient term of the error dominates, c denotes the wave propagation speed in the medium, and ρ is the diameter of Ω.

Finally, it is assumed that the procedures leading to (5.15) and (5.20) exhibit the same order of growth, allowing the definition of the error indicator $\mathcal{E}_1$:

$$\mathcal{E}_1 = C_{1,i}\|\mathbf{R}_i(\mathbf{U}^h)\|_\Omega + C_{1,d}\|\mathbf{R}_d(\mathbf{n},\mathbf{U}^h)\|_{\partial\Omega} \tag{5.21}$$

$$\approx \|\Delta\mathbf{U}^h\|_\Omega \tag{5.22}$$

with constants:

$$C_{1,i} = \frac{C_{PF}^2}{c\rho^{\frac{1}{2}}(1+C_{PF})} \tag{5.23}$$

$$C_{1,d} = \frac{C_{PF}\left(C_{T0}C_{PF}+C_{T1}\right)}{c\rho^{\frac{1}{2}}(1+C_{PF})} \tag{5.24}$$

If this indicator is evaluated at the element level, it allows for a local estimation of the error distribution over the mesh, enabling adaptive discretization.

5.2.1.1. Approximation of the constants in the error indicator

The constants $C_{1,i}$ and $C_{1,d}$ in $\mathcal{E}_1$ will be approximated numerically to improve the performance of the indicator. On one hand, the values of C_{PF} and C_{T1} will be obtained by numerically solving the maximization of Rayleigh quotients through the solution of generalized eigenvalue problems. The values of these constants will be equal to the square root of the largest eigenvalue. Specifically, the following problems are to be solved:

$$C_{PF} = \sup_{u\in H^1(\Omega)} \frac{\|u\|_\Omega}{\|\nabla u\|_{H^1(\Omega)}} = \sqrt{\lambda_{\max}^{(1)}} \quad , \quad C_{T0} = \sup_{u\in H^1(\Omega)} \frac{\|u\|_\Omega}{\|u\|_{\partial\Omega}} = \sqrt{\lambda_{\max}^{(2)}} \tag{5.25}$$

which will be approximated by solving the following generalized eigenvalue problems:

$$\mathbf{K}u = \lambda^{(1)}\mathbf{M}u \quad , \quad \mathbf{m}u = \lambda^{(2)}\mathbf{M}u \tag{5.26}$$

where $\mathbf{K}$, $\mathbf{M}$, and $\mathbf{m}$ are the stiffness and mass matrices over the full domain and its boundary, computed using finite elements for the domain Ω. In our case, domain discretizations with a single element will be used, and the interpolation degree will be progressively increased until convergence is reached. In this way, when computing on cut elements, the integration subdomain decomposition defined for the cgDG method can be reused, reducing computational cost.

Finally, C_{T0} is obtained directly from the geometric parameters of the domain:

$$C_{T0} = \sqrt{\frac{|\partial\Omega|}{|\Omega|}} \tag{5.27}$$

where $|\partial\Omega|$ and $|\Omega|$ denote the perimeter and area of Ω, respectively.

5.2.2. Mesh adaptation criteria

Once the error indicator $\mathcal{E}_1$ is available, the next steps in constructing the *hp*-adaptation procedure involve developing strategies to assess the effectiveness of each type of refinement and defining a criterion to terminate the adaptation loop.

The proposed strategies for both objectives are based on evaluating the indicator $\mathcal{E}_1$ on an adapted version of the original discretization. To this end, it is essential to have tools that enable the evaluation of these estimators with minimal loss of accuracy when transitioning from one discretization to another. In Section 5.5, projections are defined that satisfy these requirements and will be used in this stage of the solution-characterization process.

Remark. It has been observed that using the standard L^2 projection, the proposed strategies are significantly less effective. We attribute this behavior to the fact that the difference between the L^2 projection and the best projection according to the DG formulation is of the same order as the error on the refined mesh.

To determine the most suitable refinement strategy, several alternatives have been considered. The two most promising are presented below. The first strategy consists in obtaining two auxiliary discretizations, each resulting from applying a different type of refinement. After formulating and solving the corresponding cgDG systems, it is possible to obtain the error indicators $\mathcal{E}_1^{h^{\uparrow}}$ and $\mathcal{E}_1^{p^{\uparrow}}$ for the h-refined and p-refined versions of the original discretization, respectively. The refinement strategy yielding the lower error indicator is then selected for adaptation.

For this alternative, the parameter to be compared with $\mu_{\mathfrak{r}}$ in the context of the *hp*-refinement criterion is defined as:

$$\mathfrak{r}_0 = \frac{\mathcal{E}_1^{p^{\uparrow}}}{\mathcal{E}_1^{h^{\uparrow}}} \tag{5.28}$$

The second strategy consists of obtaining only the h-refined version of the initial discretization and analyzing the distribution of the error indicator among the new elements. If we use $\{\mathcal{E}_1^{h^{\uparrow}}\}$ to denote the set of indicators evaluated on each of the new elements in the h-refined mesh, and represent the mean and standard deviation operators as $\mathbb{E}$ and σ, respectively, the proposed parameter is:

$$\mathfrak{r}_1 = \frac{\mathbb{E}(\{\mathcal{E}_1^{h^{\uparrow}}\})}{\sigma(\{\mathcal{E}_1^{h^{\uparrow}}\})} \tag{5.29}$$

This parameter aims to identify irregular distributions of error within elements of the original mesh to decide on h-refinement. The underlying idea is that the error will concentrate in regions where the solution is less smooth and that these regions are limited in extent, smaller than the element size. In cases where the error is uniformly distributed across the element, the approximation is understood to be too rigid to capture solution oscillations.

An illustrative case that conveys the intuition behind this strategy is presented in Fig. 5.1. It features a shock wave $\mathbf{U}^{\mathrm{sh}}$ propagating at a certain angle θ relative to

the planes of a Cartesian mesh. Assuming that element e of the original mesh has been selected for refinement, the next step is to evaluate the error indicator on each of the elements $\{e_1, e_2, e_3, e_4\}$ of the locally h-refined mesh. Since the wave front lies within e_1, it is expected that this element will exhibit significantly higher error levels. Indeed, it can be anticipated that the majority of this error will manifest as a jump in the flux entering e_1 through the faces intersected by the singularity.

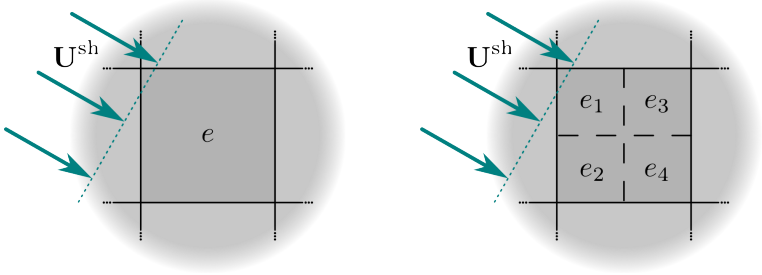


Figure 5.1: *Hypothetical scheme with a shock wave entering into an element selected for refinement*

Although both proposed strategies have been successfully employed within the adaptation scheme, the first one presents disadvantages. First, the need to generate and solve two alternative refinements makes this strategy more complex and computationally expensive. Moreover, setting a threshold for the hp-refinement criterion based on parameter $\mathfrak{r}_0$ is challenging, as its values tend to lie within a narrow range and, without fine tuning, h-refinement is usually favored. For these reasons, the use of parameter $\mathfrak{r}_1$ has been adopted, despite its less intuitive interpretation and occasionally less consistent behavior.

Finally, it remains to define a criterion to determine when to terminate the adaptation loop. The ultimate goal of the refinement process is to distribute the nodes within the domain in such a way that maximum accuracy is achieved with the available resources. Therefore, the stopping criterion must be governed by a measure of the expected error in the new discretization. In this work, we choose to evaluate the error estimator on successive discretization proposals generated during the adaptation loop and to terminate the process when the effectiveness of the proposals decreases. Applying this criterion results in meshes that exhibit a uniform error distribution per element, which has been shown to be optimal in other finite element simulation contexts.

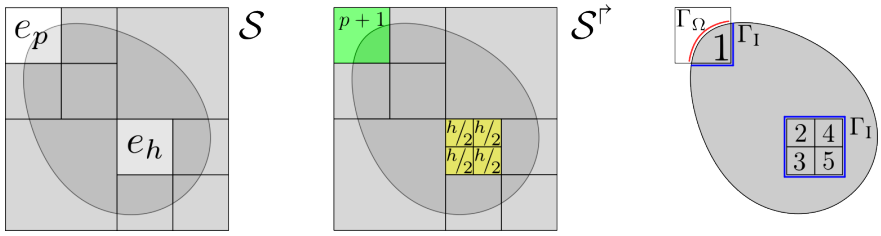
Let ${}_{\mathcal{S}}\mathcal{E}^{\mathrm{r}}_{1,b}$ denote the error indicator over the best mesh proposal, and ${}_{\mathcal{S}}\mathcal{E}^{\mathrm{r}}_{1,i}$ the same indicator evaluated over the proposal generated by the current adaptation iteration. With this notation, the parameter to be compared against the user-defined threshold $\mu_{\mathfrak{o}}$ for evaluating the stopping criterion is given by:

$$\mathfrak{o}_1 = \frac{{}_{\mathcal{S}}\mathcal{E}^{\mathrm{r}}_{1,i}}{{}_{\mathcal{S}}\mathcal{E}^{\mathrm{r}}_{1,b}} \tag{5.30}$$

Note that evaluating the indicators exclusively over the subdomains where the discretization has changed — namely, the set of elements involved in the adaptation up to that poin t— yields a parameter with a similar meaning. However, this variant is less stable due to the changing domain of evaluation and makes it more difficult for the user to define a consistent and reliable threshold for the stopping criterion.

In any case, evaluating the error indicator on a new discretization—whether globally ($\mathfrak{o}_1$) or at the element level ($\mathfrak{r}_0, \mathfrak{r}_1$)—only involves local computations and has low computational cost. As pointed out in Section 2.2.2, thanks to the combination of the DG formulation with explicit time integrators, the cgDG method decomposes the global problem for computing the time derivative of the field into a set of decoupled element-wise problems. Since the proposed error indicator only depends on the field and its time derivative at a specific subdomain and instant, its evaluation on a new mesh only requires computing the time derivative that cgDG would provide for this new configuration and evaluating (5.22) on the desired domain.

Let us consider the example shown in Fig. 5.2 with a mesh $\mathcal{S}$ containing two elements, e_h and e_p, which have been selected for h- and p-refinement, respectively. The field and its time derivative in all elements of $\mathcal{S}^{\uparrow}$ marked in gray will be preserved from $\mathcal{S}$. However, the time derivative in the refined elements (green for p-refinement and yellow for h-refinement) will change. The update of the time derivative in these subdomains is achieved by solving the 5 systems associated with the new elements. The only additional information required is the magnitude of the incoming flux due to the exterior field, which is obtained either by evaluating the current field outside the element on Γ_I or by applying the problem's boundary conditions on Γ_Ω. Therefore, it is not strictly necessary to solve the problem over the entire domain using $\mathcal{S}^{\uparrow}$ in order to determine the updated solution with the adapted mesh.



(a) *Original discretization* ***(b)*** *Refined discretization* ***(c)*** *Systems to solve*

Figure 5.2: *Scheme for computing the solution on a refined mesh via local update of the field time derivative*

5.3. Multigrid indicators

5.3.1. Set of calculation meshes

The second proposed adaptation procedure aligns with multigrid strategies [99] and is based on parallel resolution of the same problem with three distinct discretizations. A reference discretization $\dot{\mathcal{S}}$ of the domain Ω is considered, with a total of $\dot{N}$ degrees of freedom associated with elements characterized by sizes $\dot{\mathbf{h}}$ and approximation degrees $\dot{\mathbf{p}}$. From $\dot{\mathcal{S}}$, two additional discretizations $\check{\mathcal{S}}$ and $\hat{\mathcal{S}}$ are defined through uniform approximation degree reduction and uniform element size refinement, respectively.

In Table 5.2, a summary of the discretizations considered and their relationships is provided. The functions $h(\mathbf{x})$ and $p(\mathbf{x})$ associate each point $\mathbf{x}$ in Ω with the size and approximation degree of the element in $\mathcal{S}$ containing that point. Likewise, Fig. 5.3 illustrates an example of a set of three discretizations that satisfy the imposed constraints.

Discretization	Element sizes	Approximation degrees
$\check{\mathcal{S}}$	$\check{\mathbf{h}} = \dot{\mathbf{h}}$	$\check{\mathbf{p}} = \dot{\mathbf{p}} - 1$
$\dot{\mathcal{S}}$	$\dot{\mathbf{h}}$	$\dot{\mathbf{p}}$
$\hat{\mathcal{S}}$	$\hat{\mathbf{h}} = [\hat{h}_e : \hat{h}(\mathbf{x}) = 2^{-1}\dot{h}(\mathbf{x})]$	$\hat{\mathbf{p}} = [\hat{p}_e : \hat{p}(\mathbf{x}) = \dot{p}(\mathbf{x})]$

Table 5.2: *Summary of discretizations required for the multi-discretization adaptation scheme and their relationships.*

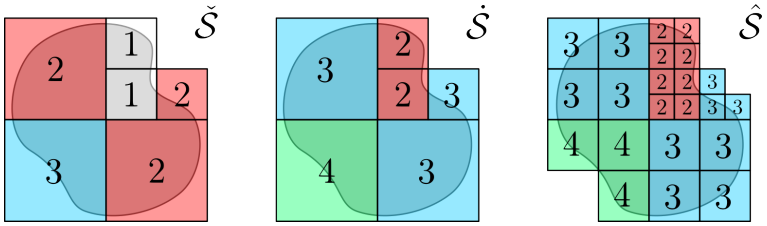


Figure 5.3: *Set of discretizations valid for use in the multi-discretization scheme (numbers and colors identify the approximation degree for each element).*

The finest discretization and the one used by the user to evaluate all quantities of interest will be $\hat{\mathcal{S}}$. However, the proposed mesh hierarchy ultimately restricts the set of admissible fine discretizations by $\check{\mathcal{S}}$. Specifically, the only additional restriction imposed by $\check{\mathcal{S}}$ on $\hat{\mathcal{S}}$, beyond those already enforced by $\dot{\mathcal{S}}$, concerns the minimum admissible value in $\hat{\mathbf{p}}$. If the lowest order approximation in $\hat{\mathcal{S}}$ is linear, then $\dot{\mathcal{S}}$ effectively governs the discretization by defining the partition of Ω.

Continuing the example presented in Fig. 5.3, Fig. 5.4 illustrates two discretizations $\mathcal{S}_1$ and $\mathcal{S}_2$ close to $\hat{\mathcal{S}}$ but incompatible with $\dot{\mathcal{S}}$. Specifically, elements with highlighted degree values in red violate the restriction on $\hat{\mathbf{p}}$, while elements with highlighted contours do not comply with the size relation required for $\hat{\mathbf{h}}$.

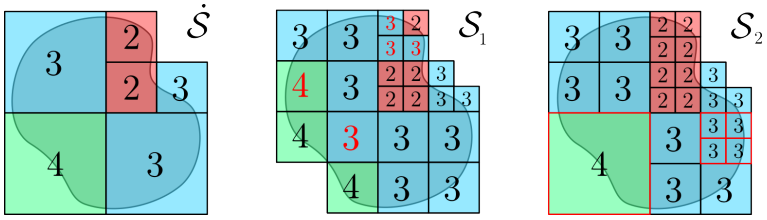


Figure 5.4: *Examples of discretizations $\mathcal{S}_1$ and $\mathcal{S}_2$ violating approximation degree and element size constraints imposed by the reference discretization $\dot{\mathcal{S}}$.*

Regarding computational cost, calculation times will be comparable to those required for resolving the finest discretization in isolation. The most costly preprocessing step is calculating intersections and designing quadratures. Due to the hierarchical relationship between meshes, these procedures only need to be completed for $\hat{\mathbf{p}}$, reusing results in coarser mesh calculations. Moreover, for uniform meshes, the number of nodes significantly increases between discretizations, with $\hat{N} \approx 4\dot{N} = 4(\check{N}^{\frac{1}{2}} + 1)^2$. The increased number of fluxes and systems to solve at each iteration of the temporal integration loop is even more significant. Each mesh element division quadruples these calculations, justifying $\hat{\mathcal{S}}$'s computational cost dominance.

5.3.2. Error indicator

With the configuration established, the adaptation of $\check{\mathcal{S}}$ can be viewed as a consequence of operating on $\dot{\mathcal{S}}$. Therefore, to guide the mesh adaptation strategy, it is sufficient to understand the error distribution in $\dot{\mathcal{S}}$. For this purpose, the error of the solution $\dot{\mathbf{U}}$ is evaluated as its difference with respect to the solution $\hat{\mathbf{U}}$ computed on the fine mesh. In this section, the notation for the approximate fields has been simplified by omitting the superscript h. In any case, no confusion should arise, as they are clearly associated with one of the discretizations.

The literature contains several results on a priori error estimators for DG methods applied to hyperbolic problems; see, for instance, [164, 165, 166], which demonstrate algebraic convergence to the exact solution as the mesh size decreases. Similarly, numerical analyses such as those presented in Section 3.6.1.1 confirm that the behavior predicted by mathematical theory indeed manifests in practice. If we consider a norm $\|\cdot\|_{\square^s(\Omega)}$, which may vary depending on the specific reference, all optimal results provide error estimates of the form:

$$\|\mathbf{U} - \dot{\mathbf{U}}\|_{\square^s(\Omega)} \leq \dot{C}_{\square} \dot{h}^{\eta} \|\mathbf{U}\|_{\square^s(\Omega)} \tag{5.31}$$

where the constant $\dot{C}_{\square}$ depends on the regularity of the domain, the chosen norm, and the approximation degree, and the exponent $\dot{\eta}$ again depends on the approximation degree and the smoothness of the solution s. This expression is explicit with respect to the element size, allowing us to track error evolution across sequences of h-refined discretizations. Using the ratio of the convergence laws for the discretizations $\hat{\mathcal{S}}$ and $\dot{\mathcal{S}}$, and considering the construction constraints described in Section 5.3.1, one can relate the errors of $\hat{\mathbf{U}}$ and $\dot{\mathbf{U}}$:

$$\|\mathbf{U} - \hat{\mathbf{U}}\|_{\square^s(\Omega)} \leq 2^{-\dot{\eta}} \|\mathbf{U} - \dot{\mathbf{U}}\|_{\square^s(\Omega)} \tag{5.32}$$

Applying the inverse triangle inequality yields

$$\|\hat{\mathbf{U}} - \dot{\mathbf{U}}\|_{\square^s(\Omega)} \geq \left(1 - 2^{-\dot{\eta}}\right) \|\mathbf{U} - \dot{\mathbf{U}}\|_{\square^s(\Omega)} \tag{5.33}$$

This inequality shows that most of the error in the coarse solution is captured by comparing it with the solution obtained on the fine mesh. Additionally, in this work, we assume the hypothesis that convergence to the exact solution exhibits similar behavior both in integral and pointwise senses. Thus, we define:

$$\hat{\Delta}\dot{\mathbf{U}} = \hat{\mathbf{U}} - \dot{\mathbf{U}} \approx \Delta\dot{\mathbf{U}} \tag{5.34}$$

as the approximate error field for the solution $\dot{\mathbf{U}}$.

This field allows us to approximately evaluate quantities that involve the error on the reference mesh. In particular, in this work we use the quantity $\dot{\mathcal{E}}_2$ as the error indicator on $\dot{\mathcal{S}}$, which is derived directly from the bilinear form $\mathcal{A}$ defined in (5.5) by adding the L^1-norm of its two terms:

$$\dot{\mathcal{E}}_2 = \left\| \hat{\Delta}\dot{\mathbf{U}} \cdot \mathbf{R}_i(\hat{\Delta}\dot{\mathbf{U}}) \right\|_{L^1(\Omega_{\dot{e}})} + \left\| \hat{\Delta}\dot{\mathbf{U}} \cdot \mathbf{R}_d(\mathbf{n}, \hat{\Delta}\dot{\mathbf{U}}) \right\|_{L^1(\partial\Omega_{\dot{e}})} \tag{5.35}$$

If we define the DG-norm $|||\cdot|||_{\Omega}$ as:

$$|||\mathbf{V}|||_{\Omega} = \|\mathbf{V} \cdot \mathbf{R}_i(\mathbf{V})\|_{L^1(\Omega)} + \|\mathbf{V} \cdot \mathbf{R}_d(\mathbf{n}, \mathbf{V})\|_{L^1(\partial\Omega)} \tag{5.36}$$

this error measure is simply:

$$\dot{\mathcal{E}}_2 = \left|\left| \hat{\Delta}\dot{\mathbf{U}} \right|\right|_{\Omega_{\dot{e}}} \tag{5.37}$$

Remark. If one examines (5.4) applied to $\Delta\dot{\mathbf{U}}$, it becomes clear that there are cases in which $\mathbf{R}_d$ can be computed exactly from $\dot{\mathbf{U}}$, and using $\hat{\Delta}\dot{\mathbf{U}}$ introduces inaccuracies in the computation of $\dot{\mathcal{E}}_2$ that could be avoided. Two relevant cases are: (1) boundary segments where the incoming numerical flux does not depend on the interior field, and (2) interfaces between elements, provided they do not coincide with point or line sources. The first case can be easily identified, while the second requires a more detailed analysis of the problem configuration at each time step. In this work, we will modify the computation of $\dot{\mathcal{E}}_2$ only for (1)-cases, namely, where the numerical flux is independent of the approximate field.

5.3.3. Auxiliary estimates of polynomial approximations in cgDG

By applying approximation estimators, we aim to predict the behavior of $\mathcal{E}_2$ as the discretization is modified. In [167, 168], polynomial approximation and inverse estimators are derived for polygonal domains with essentially arbitrary shapes. The interior elements of discretizations with Cartesian meshes are characterized by their regularity (they are affine transformations of reference elements), and for these, conventional estimators can be considered. However, the approximation over cut elements presents partial support domains whose geometry can be arbitrarily complex.

In Lemma 4.31 of [168], error bounds for polynomial approximations are established. The bounds used in this work include certain modifications to adapt them to the cgDG framework. The estimate for the global approximation error reads:

$$\|\Delta\pi_p(v)\|_\Omega \leq C_1 \frac{\rho_\Omega^k}{p^s} \|\mathfrak{E}u\|_{H^s(K)} \tag{5.38}$$

while the trace estimate is given by:

$$\|\Delta\pi_p(v)\|_{\partial\Omega_i} \leq \phi(p,k,\partial\Omega_i)^{\frac{1}{2}} \, |\partial\Omega_i|^{\frac{1}{2}} \, \frac{\rho_\Omega^{k-1}}{p^{s-\frac{1}{2}}} \|\mathfrak{E}u\|_{H^s(K)} \tag{5.39}$$

with

$$\phi(p,k,\partial\Omega_i) = C_2 \min\left(\frac{\rho_\Omega^2}{|\partial\Omega_i| \, \psi\left(\partial\Omega_i\right)}, p \right) \tag{5.40}$$

The parameters appearing in these inequalities, along with their meaning and most relevant dependencies, are as follows:

- Ω: curved polygon in $\mathbb{R}^2$ with locally Lipschitz boundary
- K: parallelogram containing Ω such that $\rho_\Omega \leq C_\rho \rho_K$
- $\partial\Omega_i$: portion of the boundary of Ω for which there exists a point $\mathbf{x}_i^0$ in Ω visible from any point $\mathbf{x}$ on $\partial\Omega_i$
- k: positive integer defined as $k = \min(p+1, s)$, where $s > 1$ measures the smoothness of the approximated function
- $\mathfrak{E}$: linear extension operator $H^s(\Omega) \to H^s(K)$ for which there exists a constant $C_\mathfrak{E} > 0$ depending on the Lipschitz constant of Ω and on s, such that $\|\mathfrak{E}u\|_{H^s(K)} \leq C_\mathfrak{E} \|u\|_{H^s(\Omega)}$ (see [169])
- ψ: function measuring the visibility of the boundary $\partial\Omega_i$ from the interior of Ω

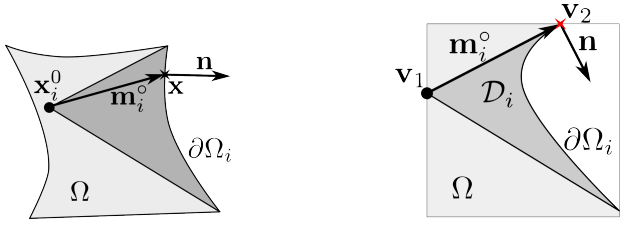
C_1, C_2: positive constants depending only on s, C_ρ, and the regularity of K and Ω

The modifications with respect to the original expressions concern the projection π_p, the choice of the extension domain K, and the definition of ψ. First, the projection space has been enriched by including cross terms that arise when considering the tensor product of polynomials, rather than the space of polynomials with total degree at most p. This modification only affects the constants, but not the structure of the expressions. The choice of a square extension domain is natural given the new approximation space. Moreover, identifying a valid extension domain in cgDG is trivial since the full element domain contains Ω. It is worth noting that the value of C_ρ will increase for cut elements with small support domains. However, the extension strategy used to stabilize the solution compensates for this effect and removes the possibility of encountering elements with arbitrarily small support domains for which $C_\rho \to \infty$.

Let us now discuss the definition of ψ in more detail. Equation (5.39) is originally obtained from a geometric setting such as the one shown in Fig. 5.5a. In this setting, it is assumed that there exists a point $\mathbf{x}_i^0$ in Ω such that $\mathbf{m}_i^\circ(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) > 0$, with $\mathbf{m}_i^\circ(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_i^0)$, for every point $\mathbf{x}$ on $\partial\Omega_i$, from which one derives:

$$\psi^\circ(\partial\Omega_i) = \sup_{\mathbf{x}_i^0 \in \Omega} \min_{\mathbf{x} \in \partial\Omega_i} (\mathbf{m}_i^\circ(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})) \tag{5.41}$$

In the cgDG context, $\partial\Omega$ is naturally partitioned according to changes in the definition of the domain boundary, and—as described in Section 3.4—each of these segments is associated with an integration subdomain of Ω. By construction, the vertex of each integration subdomain opposite to the side $\partial\Omega_i$ satisfies the condition imposed on the point $\mathbf{x}_i^0$ across all of $\partial\Omega_i$, except at its endpoints. This is illustrated in Fig. 5.5b. The boundary of the integration subdomain $\mathcal{D}_i$ contains $\partial\Omega_i$, and the vertex $\mathbf{v}_1$ satisfies the condition required of $\mathbf{x}_i^0$ throughout $\partial\Omega_i$, except at $\mathbf{v}_2$ where the vector $\mathbf{v}_2 - \mathbf{v}_1$ is perpendicular to $\mathbf{n}(\mathbf{v}_2)$. To avoid additional computations when evaluating convergence, it is convenient to relax the condition on $\mathbf{x}_i^0$ such that vertices like $\mathbf{v}_1$ are considered valid points.



(a) *Curved polytopic scheme* ***(b)*** *cgDG scheme*

Figure 5.5: *Comparison of curved polytopic and cgDG settings for the evaluation of trace estimates*

The original estimator for the trace is derived by analyzing the divergence of the function $v^2\mathbf{m}_i^\circ$, leading to

$$\|v\|^2_{\partial\Omega_i} \leq \frac{2+\zeta}{\min\limits_{\mathbf{x}\in\partial\Omega_i}(\mathbf{m}_i^\circ\cdot\mathbf{n})}\|v\|^2_{\Omega_i} + \frac{\max\limits_{\mathbf{x}\in\partial\Omega_i}|\mathbf{m}_i^\circ|_2^2}{\zeta\min\limits_{\mathbf{x}\in\partial\Omega_i}(\mathbf{m}_i^\circ\cdot\mathbf{n})}\|\nabla v\|^2_{\Omega_i} \tag{5.42}$$

for any positive ζ, and then substituting $\zeta = p$ and bounding the norm of $\mathbf{m}_i^\circ$ using the diameter of the domain. After some manipulations and term grouping, one arrives at (5.41).

In our case, the trace norm will be estimated based on the geometric configuration shown in Fig. 5.6. First, a domain Ω_i is defined, bounded by $\mathbf{x}_i^0$ and $\partial\Omega_i$, from which a partition into r subdomains $\{\Omega_i^j\}$ is constructed. Next, we identify the domain $\tilde{\Omega}_i^j$, which is the triangle formed by the vertices of Ω_i^j. The boundary of $\tilde{\Omega}_i^j$ opposite to $\mathbf{x}_i^0$ is denoted as $\partial\tilde{\Omega}_i^j$, with $\tilde{\mathbf{n}}_i^j$ its outward normal, and $\tilde{\mathfrak{n}}_i^j$ the vector obtained by applying the absolute value to each component of $\tilde{\mathbf{n}}_i^j$.

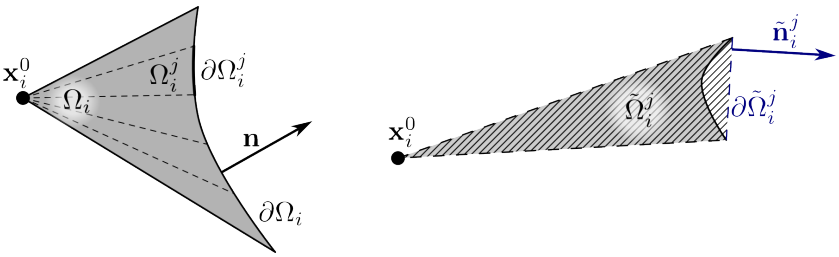


Figure 5.6: *Geometric setting in the inference of trace estimates*

We now propose to use the vector $\tilde{\mathbf{m}}_i^j = \mathbf{m}_i^\circ(\mathbf{x}) \odot \tilde{\mathfrak{n}}_i^j$, where $\odot$ denotes the Hadamard (element-wise) product, as a substitute for the vector $\mathbf{m}_i^\circ$ employed in the original procedure. This new choice preserves the key properties—constant divergence and tangency with respect to the sides of $\tilde{\Omega}_i^j$ that include $\mathbf{x}_i^0$—which made the adoption of $\mathbf{m}_i^\circ$ suitable:

$$\nabla\cdot\tilde{\mathbf{m}}_i^j(\mathbf{x}) = \left|\tilde{\mathbf{n}}_i^j\right|_1 \quad , \quad \tilde{\mathbf{m}}_i^j(\mathbf{x})\cdot\mathbf{n}_i(\mathbf{x}) = 0\ \forall\ \mathbf{x}\in\{\partial\Omega_i^j\}^c \tag{5.43}$$

where $|\mathbf{v}|_n$ denotes the L^n norm of the vector $\mathbf{v}$.

By analyzing the divergence of $v^2\tilde{\mathbf{m}}_i^j$ over one of the subdomains, we obtain:

$$\int_{\partial\Omega_i^j} v^2\tilde{\mathbf{m}}_i^j\cdot\mathbf{n}\,d\Gamma = \int_{\Omega_i^j}\nabla\cdot(v^2\tilde{\mathbf{m}}_i^j)\,d\Omega \tag{5.44}$$

$$\min_{\mathbf{x}\in\partial\Omega_i^j}\left(\tilde{\mathbf{m}}_i^j\cdot\mathbf{n}\right)\|v\|^2_{\partial\Omega_i^j} \leq \left|\tilde{\mathbf{n}}_i^j\right|_1\|v\|^2_{\Omega_i^j} + 2\max_{\mathbf{x}\in\partial\Omega_i^j}\left|\tilde{\mathbf{m}}_i^j\right|_2\|v\|_{\Omega_i^j}\|\nabla v\|_{\Omega_i^j} \tag{5.45}$$

Summing over all members of the partition and introducing the parameter $\zeta > 0$, we obtain:

$$\|v\|^2_{\partial\Omega_i} \leq \sum_{j=1}^{r} \left(\frac{\left|\tilde{\mathbf{n}}_i^j\right|_1 + \zeta}{\min\limits_{\mathbf{x}\in\partial\Omega_i^j} \left(\tilde{\mathbf{m}}_i^j \cdot \mathbf{n}\right)} \|v\|^2_{\Omega_i^j} + \frac{\max\limits_{\mathbf{x}\in\partial\Omega_i^j} \left|\tilde{\mathbf{m}}_i^j\right|_2^2}{\zeta \min\limits_{\mathbf{x}\in\partial\Omega_i^j} \left(\tilde{\mathbf{m}}_i^j \cdot \mathbf{n}\right)} \|\nabla v\|^2_{\Omega_i^j} \right) \tag{5.46}$$

When $r \to \infty$, it holds that $\tilde{\mathbf{n}}_i^j \to \mathbf{n}$. If we define $\mathbf{m}_i = \mathbf{m}_i^\circ \odot \mathfrak{n}$ over $\partial\Omega_i$, we obtain:

$$\|v\|^2_{\partial\Omega_i} \leq \frac{\sqrt{2} + \zeta}{\min\limits_{\mathbf{x}\in\partial\Omega_i} (\mathbf{m}_i \cdot \mathbf{n})} \|v\|^2_{\Omega_i^j} + \frac{\max\limits_{\mathbf{x}\in\partial\Omega_i} |\mathbf{m}_i|_2^2}{\zeta \min\limits_{\mathbf{x}\in\partial\Omega_i} (\mathbf{m}_i \cdot \mathbf{n})} \|\nabla v\|^2_{\Omega_i^j} \tag{5.47}$$

By applying the same manipulations as in (5.42), we arrive at the new expression for ψ, which is simply:

$$\psi(\partial\Omega_i) = \sup_{\mathbf{x}_i^0 \in \Omega} \min_{\mathbf{x}\in\partial\Omega_i} (\mathbf{m}_i(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})) \tag{5.48}$$

This new trace estimate is bounded as long as $\mathbf{m}_i \cdot \mathbf{n} > 0$ at every point on $\partial\Omega_i$. The new condition requires that the domain Ω_i has an opening angle at the vertex $\mathbf{x}_i^0$ of less than 90°, but it allows for subdomains with tangency points. This alternative approach is convenient for evaluation in cgDG, as it permits the direct use of the integration subdomains or others trivially derived from them.

The field $\mathbf{m}_i$ can also be used to derive new versions of other estimators, as shown in Appendix C, where a new expression is provided for the trace-inverse estimate for polynomial functions.

5.3.4. Estimates on multigrid error indicator

With appropriate manipulation and using the approximation estimates presented in the previous section, one can obtain predictions about the convergence behavior of the error measure $\mathcal{E}_2$. Following the philosophy of Richardson methods [72, 74], it is possible to use these results to estimate error values on meshes different from the original, provided that their relationship is known. This section explores that idea and derives relations between the error measures for meshes having the characteristics discussed in Section 5.3.1.

First, let us introduce the projection operator $\mathbf{\Pi}_p$, which will be discussed in more detail in Section 5.5. This operator is nothing more than the standard projection π_p from $L^2(\Omega)$ into $\mathcal{Q}^p(\Omega)$, applied component-wise to vector-valued functions, and satisfies:

$$\Delta\dot{\mathbf{U}} = \mathbf{U} - \dot{\mathbf{U}} \tag{5.49}$$
$$= \mathbf{U} - \mathbf{\Pi}_{\dot{p}}(\mathbf{U}) + \mathbf{\Pi}_{\dot{p}}(\mathbf{U}) - \dot{\mathbf{U}} \tag{5.50}$$
$$= \mathbf{U} - \mathbf{\Pi}_{\dot{p}}(\mathbf{U}) + \mathbf{\Pi}_{\dot{p}}(\mathbf{U} - \dot{\mathbf{U}}) \tag{5.51}$$
$$= \Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}) + \mathbf{\Pi}_{\dot{p}}(\Delta\dot{\mathbf{U}}) \tag{5.52}$$

Taking into account the projection of the DG solution error, substituting into the bilinear form, and recalling (5.9), we obtain:

$$\mathcal{A}(\Delta\dot{\mathbf{U}}, \Delta\dot{\mathbf{U}}) = \mathcal{A}(\Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}), \dot{\mathbf{U}}) + \mathcal{A}(\mathbf{\Pi}_{\dot{p}}(\Delta\dot{\mathbf{U}}), \dot{\mathbf{U}}) \tag{5.53}$$
$$= \mathcal{A}(\Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}), \dot{\mathbf{U}}) \tag{5.54}$$
$$\leq \left\| \Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}) \cdot \mathbf{R}_i(\dot{\mathbf{U}}) \right\|_{L^1(\Omega_{\dot{e}})} + \left\| \Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}) \cdot \mathbf{R}_d(\mathbf{n}, \dot{\mathbf{U}}) \right\|_{L^1(\partial\Omega_{\dot{e}})} \tag{5.55}$$

In the context of other DG methods [164, 170], it has been shown that the projection error dominates over the projection of the error, a result that is key to deriving convergence laws. In cgDG, the same optimal convergence rates predicted by theoretical studies have been confirmed numerically. Therefore, we will assume that the projection error is also the dominant term in the cgDG setting.

The weak form of the residual equations (5.6) and (5.7), together with this hypothesis, implies that:

$$\left\| \Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}) \cdot \mathbf{R}_i(\Delta\dot{\mathbf{U}}) \right\|_{L^1(\Omega_{\dot{e}})} \gg \left\| \mathbf{\Pi}_{\dot{p}}(\Delta\dot{\mathbf{U}}) \cdot \mathbf{R}_i(\Delta\dot{\mathbf{U}}) \right\|_{L^1(\Omega_{\dot{e}})} \tag{5.56}$$
$$\left\| \Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}) \cdot \mathbf{R}_d(\mathbf{n}, \Delta\dot{\mathbf{U}}) \right\|_{L^1(\partial\Omega_{\dot{e}})} \gg \left\| \mathbf{\Pi}_{\dot{p}}(\Delta\dot{\mathbf{U}}) \cdot \mathbf{R}_d(\mathbf{n}, \Delta\dot{\mathbf{U}}) \right\|_{L^1(\partial\Omega_{\dot{e}})} \tag{5.57}$$

and inserting these results into the bilinear form of the error and recalling (5.36) yields:

$$\dot{\mathcal{E}}_2^* = \left| \left| \left| \Delta\dot{\mathbf{U}} \right| \right| \right|_{\Omega_{\dot{e}}} \tag{5.58}$$
$$\approx \left\| \Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}) \cdot \mathbf{R}_i(\dot{\mathbf{U}}) \right\|_{L^1(\Omega_{\dot{e}})} + \left\| \Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U}) \cdot \mathbf{R}_d(\mathbf{n}, \dot{\mathbf{U}}) \right\|_{L^1(\partial\Omega_{\dot{e}})} \tag{5.59}$$
$$\leq \|\Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U})\|_{\Omega_{\dot{e}}} \left\| \mathbf{R}_i(\dot{\mathbf{U}}) \right\|_{\Omega_{\dot{e}}} + \|\Delta\mathbf{\Pi}_{\dot{p}}(\mathbf{U})\|_{\partial\Omega_{\dot{e}}} \left\| \mathbf{R}_d(\mathbf{n}, \dot{\mathbf{U}}) \right\|_{\partial\Omega_{\dot{e}}} \tag{5.60}$$

Finally, by introducing the approximation estimates (5.38) and (5.39), and assuming that the lower bound is determined by the geometric characteristics of the domain, we arrive at:

$$\dot{\mathcal{E}}_2^* \leq \left(C_1 \|\dot{\mathbf{R}}_i\|_{\Omega_{\dot{e}}} + C_2 \frac{\dot{p}^{\frac{1}{2}}}{\Psi^{\frac{1}{2}}(\partial\Omega_{\dot{e}})} \|\dot{\mathbf{R}}_d\|_{\partial\Omega_{\dot{e}}} \right) \frac{\rho_{\Omega_{\dot{e}}}^{\dot{k}}}{\dot{p}^s} \|\mathfrak{E}\mathbf{U}\|_{H^s(K_{\dot{e}})} \tag{5.61}$$

where Ψ denotes the inverse of the sum of the inverses of the values obtained for ψ over each of the subdomains composing $\partial\Omega_{\dot{e}}$, and the notation has been compacted to simplify exposition.

This expression allows us to establish theoretical relationships between the error indicators computed for different meshes. These relationships will depend either on the change in the approximation degree or on the change in element size, since, by construction, only one of these two properties is modified at a time. For this reason, it suffices to study separately the effects of h- and p–refinement, which are, in fact, the only analytically tractable scenarios. Ultimately, these results provide insight into the expected error on the finest discretization and the regularity of the solution.

We begin by analyzing how h-refinement affects the relation between discretizations $\dot{\mathcal{S}}$ and $\hat{\mathcal{S}}$. First, note that the indicator will be evaluated over the domain $\Omega_{\dot{e}}$, which is contained in element $\dot{e}$ of the coarse mesh. This implies that for each evaluation, the contribution from up to 4 elements of the refined mesh must be considered; according to the mesh hierarchy in cgDG, these are the children elements of $\dot{e}$. Denoting this set of elements by $\mathfrak{C}_{\dot{e}}$, we have:

$${}_{\dot{e}}\hat{\mathcal{E}}_2^* \leq \sum_{\hat{e}_i \in \mathfrak{C}_{\dot{e}}} \left(C_1 \|\hat{\mathbf{R}}_i\|_{\Omega_{\hat{e}_i}} + C_{2,\hat{e}_i} \frac{\hat{p}^{\frac{1}{2}}}{\Psi^{\frac{1}{2}}(\partial\Omega_{\hat{e}_i})} \|\hat{\mathbf{R}}_d\|_{\partial\Omega_{\hat{e}_i}} \right) \frac{\rho_{\Omega_{\hat{e}_i}}^{\hat{k}}}{\hat{p}^s} \|\mathfrak{E}\mathbf{U}\|_{H^s(K_{\hat{e}_i})} \tag{5.62}$$

where we use that C_1 is independent of the specific element in $\mathfrak{C}_{\dot{e}}$ (it only depends on s and on the regularity of the extension domain, which in cgDG is always a parallelogram). Applying the Cauchy–Schwarz inequality:

$${}_{\dot{e}}\hat{\mathcal{E}}_2^* \leq C_1 \left(\|\hat{\mathfrak{R}}_i\|_{\Omega_{\dot{e}}} + \Upsilon_{\mathfrak{C}_{\dot{e}}} \hat{p}^{\frac{1}{2}} \|\hat{\mathfrak{R}}_d\|_{\partial\Omega_{\dot{e}}} \right) \frac{\varrho_{\mathfrak{C}_{\dot{e}}}^{\hat{k}}}{\hat{p}^s} \|\mathfrak{E}\mathbf{U}\|_{H^s(K_{\dot{e}})} \tag{5.63}$$

where

$$\|\hat{\mathfrak{R}}_i\|_{\Omega_{\dot{e}}}^2 = \sum_{\hat{e}_i \in \mathfrak{C}_{\dot{e}}} \|\hat{\mathbf{R}}_i\|_{\Omega_{\hat{e}_i}}^2 \tag{5.64}$$

$$\|\hat{\mathfrak{R}}_d\|_{\partial\Omega_{\dot{e}}}^2 = \sum_{\hat{e}_i \in \mathfrak{C}_{\dot{e}}} \|\hat{\mathbf{R}}_d\|_{\partial\Omega_{\hat{e}_i}}^2 \tag{5.65}$$

$$\Upsilon_{\mathfrak{C}_{\dot{e}}} = \max_{\hat{e}_i \in \mathfrak{C}_{\dot{e}}} (\Upsilon_{\hat{e}_i}) = \max_{\hat{e}_i \in \mathfrak{C}_{\dot{e}}} \left(\frac{C_2}{C_1 \Psi^{\frac{1}{2}}(\partial\Omega_{\hat{e}_i})} \right) \tag{5.66}$$

$$\varrho_{\mathfrak{C}_{\dot{e}}} = \max_{\hat{e}_i \in \mathfrak{C}_{\dot{e}}} \left(\rho_{\Omega_{\hat{e}_i}} \right) \tag{5.67}$$

Finally, setting $\hat{p} = \dot{p}$ yields the relation

$$\frac{{}_{\dot{e}}\hat{\mathcal{E}}_2^*}{\dot{\mathcal{E}}_2^*} \leq \frac{\|\hat{\mathfrak{R}}_i\|_{\Omega_{\dot{e}}} + \Upsilon_{\mathfrak{C}_{\dot{e}}}\, \dot{p}^{\frac{1}{2}} \|\hat{\mathfrak{R}}_d\|_{\partial\Omega_{\dot{e}}}}{\|\dot{\mathbf{R}}_i\|_{\Omega_{\dot{e}}} + \Upsilon_{\dot{e}}\, \dot{p}^{\frac{1}{2}} \|\dot{\mathbf{R}}_d\|_{\partial\Omega_{\dot{e}}}} \left(\frac{\varrho_{\mathfrak{C}_{\dot{e}}}}{\rho_{\Omega_{\dot{e}}}}\right)^{\dot{k}} \tag{5.68}$$

from which an upper bound for the error ${}_{\dot{e}}\hat{\mathcal{E}}_2^*$ can be obtained by replacing $\dot{\mathcal{E}}_2^*$ with its approximation $\dot{\mathcal{E}}_2$:

$${}_{\dot{e}}\hat{\mathcal{E}}_2 \leq \frac{\|\hat{\mathfrak{R}}_i\|_{\Omega_{\dot{e}}} + \Upsilon_{\mathfrak{C}_{\dot{e}}}\, \dot{p}^{\frac{1}{2}} \|\hat{\mathfrak{R}}_d\|_{\partial\Omega_{\dot{e}}}}{\|\dot{\mathbf{R}}_i\|_{\Omega_{\dot{e}}} + \Upsilon_{\dot{e}}\, \dot{p}^{\frac{1}{2}} \|\dot{\mathbf{R}}_d\|_{\partial\Omega_{\dot{e}}}} \left(\frac{\varrho_{\mathfrak{C}_{\dot{e}}}}{\rho_{\Omega_{\dot{e}}}}\right)^{\dot{k}} \dot{\mathcal{E}}_2 \tag{5.69}$$

In the case where the approximation degree is modified, the ratio of the errors can be derived based on the known relationship between polynomial degrees. Slightly abusing the notation by identifying $\check{e} = \dot{e}$, we obtain:

$$\frac{\dot{\mathcal{E}}_2^*}{\check{\mathcal{E}}_2^*} \leq \frac{\|\dot{\mathbf{R}}_i\|_{\Omega_{\dot{e}_i}} + \Upsilon_{\dot{e}} \dot{p}^{\frac{1}{2}} \|\dot{\mathbf{R}}_d\|_{\partial\Omega_{\dot{e}_i}}}{\|\check{\mathbf{R}}_i\|_{\Omega_{\dot{e}_i}} + \Upsilon_{\dot{e}} \check{p}^{\frac{1}{2}} \|\check{\mathbf{R}}_d\|_{\partial\Omega_{\dot{e}_i}}} \left(\frac{\check{p}}{\dot{p}}\right)^{s} \rho_{\Omega_{\dot{e}_i}}^{\dot{k}-\check{k}} \tag{5.70}$$

If the solution's regularity is limited and belongs only to $H^{s'}(K_{\dot{e}})$, with $s' \leq \check{p}+1$, the convergence ratio is reduced to $r^*_{\mathcal{E}_2,s'}$ and the relation becomes:

$$\left.\frac{\dot{\mathcal{E}}_2^*}{\check{\mathcal{E}}_2^*}\right|_{H^{s'}} \leq \frac{\|\dot{\mathbf{R}}_i\|_{\Omega_{\dot{e}_i}} + \Upsilon_{\dot{e}} \dot{p}^{\frac{1}{2}} \|\dot{\mathbf{R}}_d\|_{\partial\Omega_{\dot{e}_i}}}{\|\check{\mathbf{R}}_i\|_{\Omega_{\dot{e}_i}} + \Upsilon_{\dot{e}} \check{p}^{\frac{1}{2}} \|\check{\mathbf{R}}_d\|_{\partial\Omega_{\dot{e}_i}}} \left(\frac{\check{p}}{\dot{p}}\right)^{s'} = r^*_{\mathcal{E}_2,s'} \tag{5.71}$$

Remark. To evaluate the ratios (5.68) and (5.71), it is necessary to determine the relationship between the global approximation constant C_1 and the trace constant C_2. To estimate this relationship, a random sampling is performed from the function space generated by the trigonometric basis $\{\cos(n_1\xi_1), \sin(m_1\xi_1)\} \otimes \{\cos(n_2\xi_2), \sin(m_2\xi_2)\}$ with $\{n_1, m_1, n_2, m_2\} \in \mathbb{N}$. These functions are then projected onto the polynomial space $\mathcal{Q}^p$ using the projection operator π_p defined on the image of the domain Ω_e mapped to the a reference element $[-\pi, +\pi]^2$. Finally, the ratios between the global error and the trace error are computed, and Υ_e is estimated from the average value, based on (5.38) and (5.39).

5.3.5. Mesh adaptation criteria

In this section, we discuss how the results of error estimators and the set of mesh candidates can be employed to guide the adaptation strategy. First, we define the parameter that will determine whether to perform *h*-refinement or *p*-refinement on an element that has previously been selected due to its high error indicator.

According to (5.34), the error on the meshes $\dot{\mathcal{S}}$ and $\check{\mathcal{S}}$ can be approximately computed by evaluating the difference between their solutions with respect to $\hat{\mathbf{U}}$. Thus, for each domain Ω_{e_i} contained in both elements $\dot{e}_i$ and $\check{e}_i$, the quantities $\dot{\mathcal{E}}_2$, $\check{\mathcal{E}}_2$, and their ratio $r_{\mathcal{E}_2}$ can be calculated as follows:

$$r_{\mathcal{E}_2} = \frac{\dot{\mathcal{E}}_2}{\check{\mathcal{E}}_2} \tag{5.72}$$

Comparing this ratio with the theoretical bound in (5.71) yields an indicator of the regularity of the solution in Ω_{e_i}. If the numerically computed ratio is indeed smaller than the theoretical one, then the assumptions leading to (5.71) are likely satisfied, implying that the solution may present some form of discontinuity within $\Omega_{\dot{e}_i}$. In such cases, it is more advantageous to reduce the element size. Conversely, if the inequality is not satisfied, this indicates that increasing the approximation degree will be more efficient.

With this in mind, the following parameter is proposed for use in the *hp*-refinement criterion within the multigrid-based adaptation framework:

$$\mathfrak{r}_2 = \frac{r^*_{\mathcal{E}_2,\dot{p}}}{r_{\mathcal{E}_2}} \tag{5.73}$$

On the other hand, the objective of each refinement step is to minimize the expected error in the discretization for subsequent stages of the time integration loop. As in the scheme based on the explicit error indicator, the estimator computed on the *hp*-adapted discretization is evaluated and compared with the original estimate. The considerations discussed in Section 5.2.2 regarding the locality of the calculations remain equally valid in this context.

As discussed in Section 5.3.1, mesh adaptation within the multigrid context takes place at the level of $\dot{\mathcal{S}}$, the reference discretization. After each iteration of the refinement loop, an adaptation proposal for $\dot{\mathcal{S}}$ is available, resulting in a candidate mesh $\dot{\mathcal{S}}_i^{\Gamma}$. It is proposed to solve the cgDG formulation over the modified elements in order to obtain the updated time derivative of the field $\dot{\mathbf{U}}_i^{\Gamma}$, and with these results compute the updated error estimators $\dot{\mathcal{E}}_{2,i}^{\Gamma}$. Finally, $\dot{\mathcal{E}}_{2,i}^{\Gamma}$ is compared with the estimator $\dot{\mathcal{E}}_{2,b}^{\Gamma}$ associated with the best adaptation proposal evaluated so far, thereby assessing the effectiveness of the discretization $\dot{\mathcal{S}}_i^{\Gamma}$.

The parameter $\mathfrak{o}_2$ used to quantify the optimality of each proposal is defined as the ratio:

$$\mathfrak{o}_2 = \frac{\dot{\mathcal{E}}_{2,i}^{\Gamma}}{\dot{\mathcal{E}}_{2,b}^{\Gamma}} \tag{5.74}$$

Two important remarks must be made regarding the proposed strategy. First, the effective evaluation of the estimator on the adapted mesh is only valid if the computation preserves the accuracy of $\dot{\mathbf{U}}$. As will be shown in Section 5.5, this can be achieved by employing appropriate projection operators without altering the computational structure of cgDG.

Second, it is important to emphasize that within the mesh adaptation loop, only the reference discretization is modified and solved, while the other two discretizations remain unchanged. In this context, one might question the validity of using $\hat{\mathbf{U}}$ for

evaluating the error estimator in regions where $\dot{\mathcal{S}}$ undergoes refinement. First, we note that $\dot{\mathbf{U}}$ contains an accumulated error from previous time steps, which will in any case be larger than that accumulated by $\hat{\mathbf{U}}$.

Furthermore, due to the refinement strategy that will be applied, the number of degrees of freedom associated with a domain $\Omega_{\dot{e}}$ after refinement will always be lower than that associated with the same domain in $\hat{\mathcal{S}}$. Fig. 5.7 shows an example comparing the two discretizations resulting from refining a reference mesh configuration against the corresponding fine discretization. Looking at the number of nodes in each discretization (values in parentheses), it is evident that this number is consistently higher for $\hat{\mathcal{S}}$.

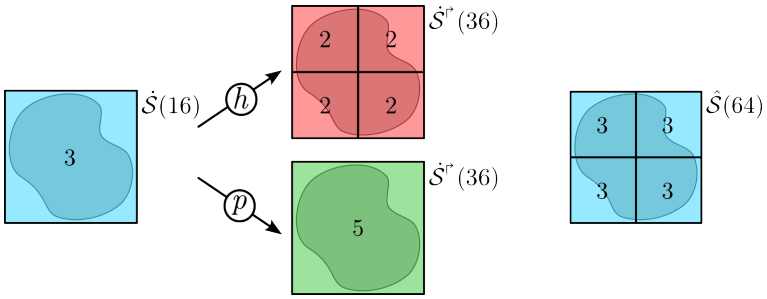


Figure 5.7: *Comparison between possible refinements of the reference discretization and the fine discretization within the adaptation loop*

In any case, after each evaluation, the results of $\dot{\mathcal{E}}_i^{\vdash}$ can be compared with the bound previously obtained for $\hat{\mathcal{E}}_2$ in (5.69). Estimates on $\dot{\mathcal{S}}_i^{\vdash}$ that are on the same order as $\hat{\mathcal{E}}_2$ are not meaningful, as they would invalidate the initial assumption in (5.34). In such cases, the domains will be assigned the error estimate from $\hat{\mathcal{S}}$ scaled by a user-defined amplification coefficient. It should be noted that none of the computations carried out so far have resulted in such a situation.

Remark. The estimator ${}_{\dot{e}}\hat{\mathcal{E}}_2$ and the parameter $\mathfrak{o}_2$ have been derived by applying results that model the asymptotic behavior of the error. This implies that, for the obtained values to be reliable, all three meshes used in the evaluation must lie within the asymptotic convergence regime toward the exact solution.

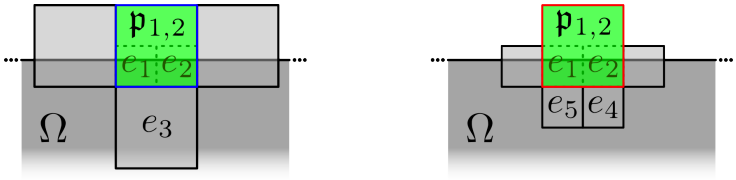
5.4. *hp*-Adaptive strategy workflow

This section includes the algorithms that comprise the discretization adaptation procedure. The overall structure of the workflow appears in Algorithm 4. The procedure starts from the current discretization $\mathcal{S}$ and a series of user-defined parameters

controlling the characteristics of the final result: the set of all admissible discretizations $\mathbb{S}$, the desired node count range $[\mu_l, \mu_u]$, the chosen error measurement strategy $\mu_{\mathcal{E}}$, and the thresholds $\mu_{\mathfrak{r}}$ and $\mu_{\mathfrak{o}}$ for the *hp*-refinement and adaptation termination criteria. Within the workflow, two blocks are distinguished: the first dedicated to analyzing the initial discretization and the second forming the adaptation loop proper.

The *hp*-adaptation strategy is shown in Algorithm 5, which is subdivided into the *hp*-refinement and coarsening strategies detailed in Algorithm 6 and Algorithm 7, respectively. Notably, the coarsening strategy is designed to prioritize increasing element size over decreasing approximation degree. Evidence of this is the continuous revision of previously de-refined elements $\mathfrak{C}_c$, switching the de-refinement type from p to h when possible.

The inclusion of Algorithm 8 at the end of the adaptation block in Algorithm 4 also aligns with this goal. This routine arises from the presence of unstable elements in the adapted mesh. To apply the stabilization strategy defined in Section 3.5, these elements must belong to a mesh level equal to or lower than their neighbors. As seen in Fig. 5.8, this imposes constraints during the adaptation process. Elements e_1 and e_2, initially stable for having most of their domain inside Ω, are selected for de-refinement. Initially, their common parent element $\mathfrak{p}_{1,2} = \mathfrak{p}_1 = \mathfrak{p}_2$ would be proposed as a replacement. However, it is likely that $\mathfrak{p}_{1,2}$ needs stabilization, which is only directly possible in the configuration shown in Fig. 5.8a due to the presence of e_3. In Fig. 5.8b, e_4 and e_5 belong to a lower mesh level and are not valid for stabilizing $\mathfrak{p}_{1,2}$.



(a) *Valid scheme for h-coarsening* ***(b)*** *Invalid scheme for h-coarsening*

Figure 5.8: *Comparison of coarsening schemes on elements with unstable parent elements*

Algorithm 8 checks the feasibility of applying h-coarsening even in these significantly more complex cases. The idea is to find the largest subset of elements such that, upon replacing them with corresponding elements from a lower mesh level, all of them have neighborhoods of the same level. The proposed algorithm identifies all possible combinations and discards those that do not meet the required neighborhood condition. Even when size increase is not allowed, approximation degree reduction is always possible, so elements discarded for h-coarsening are p-coarsened.

It should be noted that when an element's size is modified, its interpolation degree is also altered. The reason behind this is the pursuit of progressive mesh adaptation. Specifically, the polynomial degree of the resulting elements after applying h-refinement will be the minimum that implies an increase in degree-of-freedom density, while after h-coarsening, it will be the maximum that implies a reduction in

that density. Fig. 5.9 shows on the left and right examples of h-refinement and h-coarsening starting from elements of degree 3. One can see how adopting this mixed strategy avoids abrupt changes in the number of degrees of freedom.

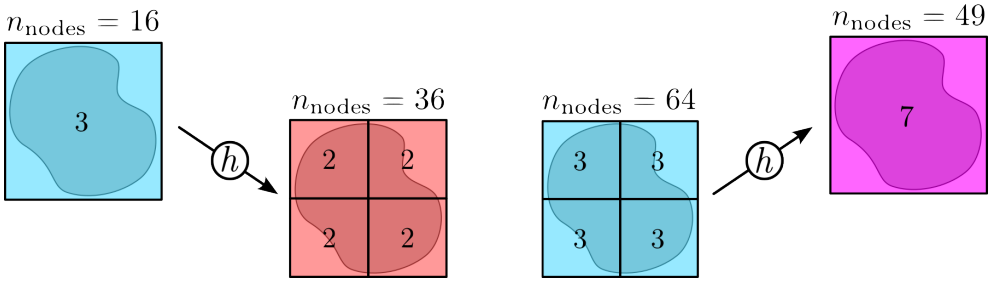


Figure 5.9: *Degree changes derived from h-adaptation*

Remark. The comments on approximation degree modification during h-adaptation do not apply to mesh relations for error estimators based on multigrid as discussed in Section 5.3.1. There, pure h-refinements were considered as indicated by the constraints in Table 5.2.

Finally, it should be noted that both error measurement and the parameters used in the *hp*-refinement and stopping criteria will depend on whether the explicit error indicator or the multigrid strategy is used and will take the forms defined in Sections 5.2 and 5.3, respectively. However, the refinement process structure is common, and for generalization, these quantities are denoted here without the subscripts indicating their strategy.

Remark. The output of the adaptation procedure is the new mesh that will be used in the next stages of the time integration loop. However, to proceed with the computation, an appropriate approximation over the new mesh structure must be obtained first. For this, the projectors presented in Section 5.5 will be applied.

Algorithm 4 Instantaneous *hp*-adaptation of the mesh based on error measures

function HPADAPTDISCRETIZATION($\mathcal{S}, \mathbb{S}, \mu_l, \mu_u, \mu_\mathcal{E}, \mu_\mathfrak{r}, \mu_\mathfrak{o}$)
  ▷ *Arrange elements regarding their error measure* ◁
  ${}_\mathcal{S}\mathcal{E} \leftarrow$ COMPUTEERRORMEASURE($\mathcal{S}, \mu_\mathcal{E}$)
  $\mathfrak{C}_e \leftarrow$ FORMELEMENTCLUSTERS($\mathcal{S}, {}_\mathcal{S}\mathcal{E}$)
  $\mathfrak{C}_e \leftarrow \mathfrak{C}_e$ clusters sorted in ascending error measure
  ▷ *Mesh adaptation loop* ◁
  $\mathcal{S}_0^\Rsh, \mathcal{S}^\Rsh \leftarrow \mathcal{S}, \mathcal{S}$ ▷ *Reference and active meshes in the adaptation procedure*
  ${}_\mathcal{S}\mathcal{E}_T \leftarrow {}_\mathcal{S}\mathcal{E}$ ▷ *Target error measure*
  if n_nodes in $\mathcal{S} \in [\mu_l, \mu_u]$ **then**
    $\mathcal{S}_b^\Rsh \leftarrow \mathcal{S}$ ▷ *Best mesh meeting user's criterion*
  $\mathfrak{C}_c \leftarrow \emptyset$
  while $\mathfrak{C}_e \neq \emptyset$ **do**
    ▷ *hp-Adapt active mesh* ◁
    $\mathcal{S}^\Rsh, \mathfrak{C}_e, \mathfrak{C}_c, \mathfrak{C}_u \leftarrow$ ADAPTACTIVEDISCRETIZATION($\mathcal{S}, \mathcal{S}_0^\Rsh, \mathcal{S}^\Rsh, \mathbb{S}, \mathfrak{C}_e, \mathfrak{C}_c, \mu_l, \mu_u, \mu_\mathcal{E}, \mu_\mathfrak{r}$)
    ▷ *Check cluster's availability* ◁
    if n_nodes in $\mathcal{S}^\Rsh \in [\mu_l, \mu_u]$ **and** $\#\mathfrak{C}_e = 1$ **then**
      $\mathfrak{C}_e \leftarrow \emptyset$
    ▷ *Compute error related measures for the proposed mesh* ◁
    ${}_\mathcal{S}\mathcal{E}_1 \leftarrow$ COMPUTEERRORMEASURE($\mathcal{S}^\Rsh, \mu_\mathcal{E}$)
    $\mathfrak{o} \leftarrow$ COMPUTESTOPPINGPARAMETER($\mathcal{S}^\Rsh, \mu_\mathcal{E}, {}_\mathcal{S}\mathcal{E}_T$)
    ▷ *Update core variables and evaluate stopping criteria* ◁
    if $\mathcal{S}_b^\Rsh$ exists **then**
      if n_nodes in $\mathcal{S}_i < \mu_l$ and $\mathfrak{C}_e \neq \emptyset$ **then**
        $\mathcal{S}_0^\Rsh \leftarrow \mathcal{S}^\Rsh$
      else
        if n_nodes in $\mathcal{S}^\Rsh \in [\mu_l, \mu_u]$ **then**
          if $\mathfrak{o} < \mu_\mathfrak{o}$ **then**
            $\mathcal{S}_0^\Rsh, \mathcal{S}_b^\Rsh, {}_\mathcal{S}\mathcal{E}_T \leftarrow \mathcal{S}^\Rsh, \mathcal{S}^\Rsh, {}_\mathcal{S}\mathcal{E}_1$
          else
            return $\mathcal{S}_b^\Rsh$
        else if n_nodes in $\mathcal{S}_i > \mu_u$ **then**
          if $\mathfrak{o} < \mu_\mathfrak{o}$ **then**
            $\mathcal{S}_0^\Rsh \leftarrow \mathcal{S}^\Rsh$
          else
            return $\mathcal{S}_b^\Rsh$
    else if n_nodes in $\mathcal{S}^\Rsh \in [\mu_l, \mu_u]$ or $\mathfrak{C}_e = \emptyset$ **then**
      $\mathcal{S}_0^\Rsh, \mathcal{S}_b^\Rsh, {}_\mathcal{S}\mathcal{E}_T \leftarrow \mathcal{S}^\Rsh, \mathcal{S}^\Rsh, {}_\mathcal{S}\mathcal{E}_1$
    else
      $\mathcal{S}_0^\Rsh \leftarrow \mathcal{S}^\Rsh$
  $\mathcal{S} \leftarrow$ COARSENELEMENTSWITHUNSTABLEPARENTS($\mathcal{S}, \mathcal{S}_b^\Rsh, \mathbb{S}, \mathfrak{C}_u$)
  return $\mathcal{S}$

Algorithm 5 Active mesh *hp*-adaptation

function AdaptActiveDiscretization($\mathcal{S}$,$\mathcal{S}_0^{\uparrow}$,$\mathcal{S}^{\uparrow}$,$\mathbb{S}$,$\mathfrak{C}_e$,$\mathfrak{C}_c$,μ_l,μ_u,$\mu_{\mathcal{E}}$,$\mu_{\mathfrak{r}}$)
  if n_{nodes} in $\mathcal{S}_0^{\uparrow} < \mu_u$ **then**
    ▷ *Compute refined version of the active mesh* ◁
    $\mathcal{S}^{\uparrow}, \mathfrak{C}_e \leftarrow$ RefineDiscretization($\mathcal{S}^{\uparrow}$,$\mathbb{S}$,$\mathfrak{C}_e$,$\mu_{\mathcal{E}}$,$\mu_{\mathfrak{r}}$)
  if n_{nodes} in $\mathcal{S}_0^{\uparrow} > \mu_l$ **then**
    ▷ *Compute coarsened version of the active mesh* ◁
    $\mathcal{S}^{\uparrow}, \mathfrak{C}_e, \mathfrak{C}_c, \mathfrak{C}_u \leftarrow$ CoarsenDiscretization($\mathcal{S}$,$\mathcal{S}^{\uparrow}$,$\mathbb{S}$,$\mathfrak{C}_e$,$\mathfrak{C}_c$,$\mathfrak{C}_u$)
  return $\mathcal{S}^{\uparrow}$,$\mathfrak{C}_e$,$\mathfrak{C}_c$,$\mathfrak{C}_u$

Algorithm 6 Active mesh *hp*-refinement

function RefineDiscretization($\mathcal{S}^{\uparrow}$,$\mathbb{S}$,$\mathfrak{C}_e$,$\mu_{\mathcal{E}}$,$\mu_{\mathfrak{r}}$)
  $\mathfrak{c}_r \leftarrow$ set in $\mathfrak{C}_e$ with the highest error measure
  $\mathfrak{C}_e \leftarrow \mathfrak{C}_e - \{\mathfrak{c}_r\}$
  for $e_i \in \mathfrak{c}_r$ **do**
    $\mathcal{S}_h^{\uparrow}, \mathcal{S}_p^{\uparrow} \leftarrow$ ComputeRefinedMeshes($\mathcal{S}^{\uparrow}, e_i$)
    if $\mathcal{S}_h^{\uparrow} \in \mathbb{S}$ and $\mathcal{S}_p^{\uparrow} \in \mathbb{S}$ **then**
      $\mathfrak{r} \leftarrow$ComputeHpRefinementParameter($\mathcal{S}^{\uparrow}$,$\mathcal{S}_h^{\uparrow}$,$\mathcal{S}_p^{\uparrow}$,$\mu_{\mathcal{E}}$)
      if $\mathfrak{r} > \mu_{\mathfrak{r}}$ **then**
        ▷ *Some kind of discontinuity is expected in the underlying field* ◁
        $\mathcal{S}^{\uparrow} \leftarrow \mathcal{S}_h^{\uparrow}$
      else if $\mathfrak{r} \leq \mu_{\mathfrak{r}}$ **then**
        ▷ *Underlying field is expected to been smooth* ◁
        $\mathcal{S}^{\uparrow} \leftarrow \mathcal{S}_p^{\uparrow}$
    else if $\mathcal{S}_h^{\uparrow} \in \mathbb{S}$ **then**
      $\mathcal{S}^{\uparrow} \leftarrow \mathcal{S}_h^{\uparrow}$
    else if $\mathcal{S}_p^{\uparrow} \in \mathbb{S}$ **then**
      $\mathcal{S}^{\uparrow} \leftarrow \mathcal{S}_p^{\uparrow}$
  return $\mathcal{S}^{\uparrow}, \mathfrak{C}_e$

Algorithm 7 Active mesh *hp*-coarsening

function CoarsenDiscretization($\mathcal{S}$,$\mathcal{S}^{\uparrow}$,$\mathbb{S}$,$\mathfrak{C}_e$,$\mathfrak{C}_c$,$\mathfrak{C}_u$)
  $\mathfrak{c}_c \leftarrow$ set in $\mathfrak{C}_e$ with the smallest error measure
  $\mathfrak{C}_e, \mathfrak{C}_c \leftarrow \mathfrak{C}_e - \{\mathfrak{c}_c\}, \mathfrak{C}_c \cup \{\mathfrak{c}_c\}$
  $\mathfrak{C}_u \leftarrow \emptyset$
  for $e_i \in \mathfrak{c}_c$ **do**
    $[\mathcal{S}_h^{\uparrow}, \mathcal{S}_p^{\uparrow}] \leftarrow$ ComputeCoarsenedMeshes($\mathcal{S}^{\uparrow}$,e_i)
    $\mathfrak{p}_i \leftarrow e_i$'s parent in cgFEM mesh hierarchy
    $\mathbb{F}_i \leftarrow$ elements from $\mathcal{S}$ that descend from $\mathfrak{p}_i$
    $c_g \leftarrow [h_{e_j} = h_{e_i} \; \forall \; e_j \in \mathbb{F}_i]$?
    $c_r \leftarrow$ Are all $e_j \in \mathbb{F}_i$ over the same domain?
    $c_p \leftarrow [p_{e_j} \leq p_{e_i} \; \forall \; e_j \in \mathbb{F}_i]$?
    $c_s \leftarrow$ Is $\mathfrak{p}_i$ stable?
    if c_g **and** c_r **and** c_p **then**
      if c_s **then**
        $\mathcal{S}^{\uparrow} \leftarrow \mathcal{S}_h^{\uparrow}$
      else
        $\mathfrak{C}_u \leftarrow \mathfrak{C}_u \cup e_i$
    else if $\mathcal{S}_p^{\uparrow} \in \mathbb{S}$ **then**
      $\mathcal{S}^{\uparrow} \leftarrow \mathcal{S}_p^{\uparrow}$
  return $\mathcal{S}^{\uparrow}$,$\mathfrak{C}_e$,$\mathfrak{C}_c$

Algorithm 8 Mesh *hp*-coarsening regarding complex cases derived from stability issues

function CoarsenElementsWithUnstableParents($\mathcal{S}$,$\mathcal{S}^{\uparrow}$,$\mathbb{S}$,$\mathfrak{C}_u$)
  $\mathfrak{E}'_h \leftarrow \{\emptyset\}$
  $\mathfrak{E}_h \leftarrow$ the powerset of $\mathfrak{C}_u$
  $\mathfrak{S}_h^{\uparrow} \leftarrow \{\mathcal{S}_k^{\uparrow} :$ComputeHCoarsenedMesh($\mathcal{S}^{\uparrow}$,$\mathfrak{e}_k$) $\forall \; \mathfrak{e}_k \in \mathfrak{E}_h\}$
  while $\mathfrak{E}'_h \neq \mathfrak{E}_h$ **do**
    $\mathfrak{E}'_h \leftarrow \mathfrak{E}_h$
    for $e_i \in \mathfrak{C}_u$ **do**
      $\mathfrak{N}_i \leftarrow e_i$'s neighbor elements
      if $\nexists \mathfrak{e}_k \in \mathfrak{E}_h : \mathfrak{N}_i / \mathfrak{e}_k = \emptyset$ **then**
        for $\mathfrak{e}_k \in \mathfrak{E}_h : e_i \in \mathfrak{e}_k$ **do**
          $\mathfrak{E}_h \leftarrow \mathfrak{E}_h - \{\mathfrak{e}_k\}$
          $\mathfrak{S}_h^{\uparrow} \leftarrow \mathfrak{S}_h^{\uparrow} - \{\mathcal{S}_k^{\uparrow}\}$
  $k \leftarrow$ index of the set $\mathfrak{E}_h$ with the maximum number of elements
  $\mathcal{S}^{\uparrow} \leftarrow$ ComputePCoarsenedMesh($\mathcal{S}_k^{\uparrow}$,$\mathfrak{C}_u - \mathfrak{e}_k$)
  return $\mathcal{S}^{\uparrow}$

5.5. Inter-discretization field projection

The simulation of transient problems with *hp*-mesh adaptation requires transferring the approximate solution between different discretizations. Each time the adaptation process is applied, the approximate solution from the original mesh must be transferred to the new mesh to proceed with time integration. As discussed in Sections 5.1, 5.2.2 and 5.4, appropriate projection operators are essential both for estimating local regularity of the solution field in the explicit indicator approach and for evaluating the benefits of a proposed mesh modification.

This work uses two projection operators. First, we define $\pi_{\Omega,p}$ as the standard projection operator that provides the polynomial in $\mathcal{Q}^p(\Omega)$ minimizing the L^2 norm error with respect to the function u. Thus, the projection satisfies:

$$\int_{\Omega} w\left(u - \pi_{\Omega,p}(u)\right) = 0 \quad \forall \quad w \in \mathcal{Q}^p(\Omega) \tag{5.75}$$

Second, the projection operator $\mathbf{\Pi}^*_{\Omega,p}$ is defined based on the DG formulation of the electromagnetic problem developed in Section 2.2. Consider a problem defined in Ω with solution $\mathbf{U}$ computed using cgDG for a single-element discretization at time t. Within the time integration loop, the proposed computational scheme evaluates the temporal variation $\partial_t \mathbf{U}$ by solving the system:

$$\int_{\Omega} \mathbf{W} \cdot \partial_t \mathbf{U} d\Omega = -\int_{\Omega} \mathbf{W} \cdot \left(\frac{\partial \mathbf{F}_k\left(\mathbf{U}\right)}{\partial x_k} - \mathbf{S}\left(\mathbf{U}\right) \right) d\Omega - \int_{\partial\Omega} \mathbf{W} \cdot [\![\mathbf{F}^-_{\mathbf{n}}\left(\mathbf{U}\right)]\!] d\Gamma \tag{5.76}$$

where $\mathbf{W}$, $\mathbf{U}$, and $\partial_t \mathbf{U}$ are polynomial vectors in $[\mathcal{Q}^p(\Omega)]^q$ with q the number of field components.

Remark. Typically, $q = 3$ in the planar case. However, when Ω lies within a PML domain, an extra component is added and $q = 4$.

Assume this discretization results from a mesh adaptation step. Then, $\mathbf{W}$ and $\partial_t \mathbf{U}$ remain in $[\mathcal{Q}^p(\Omega)]^q$, but the approximate solution is defined on the previous mesh and belongs to a different function space. Let $\mathbf{U}^{\text{↰}}$ denote the approximate field just before the update:

$$\mathbf{U}^{\text{↰}} = \lim_{\Delta t \to 0^-} \left(\mathbf{U}\left(t + \Delta t\right)\right) \tag{5.77}$$

Let the domain partition $\mathcal{S}$ be such that $\Omega_e \in \mathcal{S}$ is the region associated with element e in a Cartesian mesh with N elements. Then $\mathbf{U}^{\text{↰}}$ can be written as $\sum_{e=1}^{N} \mathbf{U}^{\text{↰}}_e$, where $\mathbf{U}^{\text{↰}}_e \in [\mathcal{Q}^{p_e}(\Omega_e)]^q$.

If $\partial\Omega_{e_1}$ is the boundary of domain Ω_{e_1}, we denote Γ_{e_1,e_2} as the portion of $\partial\Omega_{e_1}$ shared with Ω_{e_2}, and Γ_{e_1} as the intersection of $\partial\Omega_{e_1}$ with $\partial\Omega$. Fig. 5.10 shows an example where the current mesh at t^+ is preceded by a discretization composed of 4 elements.

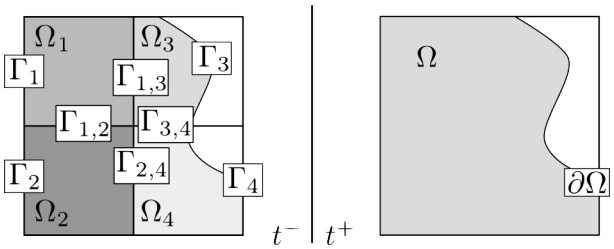


Figure 5.10: *Example of discretizations of the same domain Ω before and after mesh adaptation at time t*

With this notation, the system to solve for $\partial_t \mathbf{U}$ without loss of accuracy is:

$$\begin{aligned}\int_\Omega \mathbf{W}\cdot\partial_t\mathbf{U}d\Omega = & \\ & -\sum_{e=1}^{N}\int_{\Omega_e}\mathbf{W}\cdot\left(\frac{\partial\mathbf{F}_k\left(\mathbf{U}_e^{\text{↰}}\right)}{\partial x_k}-\mathbf{S}\left(\mathbf{U}_e^{\text{↰}}\right)\right)d\Omega \\ & -\sum_{e=1}^{N}\int_{\Gamma_e}\mathbf{W}\cdot[\![\mathbf{F}_{\mathbf{n}}^{-}\left(\mathbf{U}_e^{\text{↰}}\right)]\!]d\Gamma \\ & -\sum_{e_1=1}^{N-1}\sum_{e_2=e_1+1}^{N}\int_{\Gamma_{e_1,e_2}}\mathbf{W}\cdot\mathbf{F}_{\mathbf{n}}^{*}\left(\mathbf{U}_{e_2}^{\text{↰}}-\mathbf{U}_{e_1}^{\text{↰}}\right)d\Gamma\end{aligned} \tag{5.78}$$

where the operator $\mathbf{F}_{\mathbf{n}}^{*}$ is defined as:

$$\mathbf{F}_{\mathbf{n}}^{*}\left(\mathcal{U}\right)=\mathbf{F}_{\mathbf{n}}^{+}\left(\mathcal{U}\right)-\mathbf{F}_{\mathbf{n}}^{-}\left(\mathcal{U}\right) \tag{5.79}$$

and quantifies the flux magnitude at the boundary such that the integral:

$$\int_{\partial\mathcal{O}}\mathcal{U}\cdot\mathbf{F}_{\mathbf{n}}^{*}\left(\mathcal{U}\right)d\Gamma \tag{5.80}$$

represents the total energy crossing the boundary of $\mathcal{O}$ under field $\mathcal{U}$.

If we group all the right-hand side integrals in (5.78) into the bilinear operator $\mathcal{B}_\Pi\left(\mathbf{W},\mathbf{U}^{\text{↰}}\right)$, the projection $\mathbf{\Pi}^{\circ}_{\Omega,p}\left(\mathbf{U}^{\text{↰}}\right)\in[\mathcal{Q}^p(\Omega)]^q$ is defined as:

$$\mathcal{B}_\Pi\left(\mathbf{W},\mathbf{U}^{\text{↰}}-\mathbf{\Pi}^{\circ}_{\Omega,p}\left(\mathbf{U}^{\text{↰}}\right)\right)=\mathbf{0}\quad\forall\quad\mathbf{W}\in[\mathcal{Q}^p(\Omega)]^q \tag{5.81}$$

This yields the field in $[\mathcal{Q}^p(\Omega)]^q$ that produces the same $\partial_t \mathbf{U}$ as the original $\mathbf{U}^{\curlywedge}$. Since (5.81) involves only divergence-related terms, it is underdetermined. Alternatively, we solve:

$$\mathcal{B}_\Pi\Big(\mathbf{W}, \mathbf{U}^{\curlywedge} - \mathbf{\Pi}^*_{\Omega,p}\Big(\mathbf{U}^{\curlywedge}\Big)\Big) + \frac{1}{\kappa T}\int_\Omega \mathbf{W}\cdot\Big(\mathbf{U}^{\curlywedge} - \mathbf{\Pi}^*_{\Omega,p}\Big(\mathbf{U}^{\curlywedge}\Big)\Big) = \mathbf{0} \quad \forall \quad \mathbf{W} \in [\mathcal{Q}^p(\Omega)]^q \tag{5.82}$$

where the second term is a regularization penalizing deviation from the L^2 projection, T is the characteristic time scale, and κ (set to 10^6) controls the regularization intensity. This system can also be interpreted as minimizing energy error in the projected field at time $t + \kappa T$ assuming linear evolution.

Note that evaluating $\mathbf{\Pi}^*_{\Omega,p}$ is more costly than $\pi_{\Omega,p}$ due to the number of terms and coupling among field components, requiring larger systems to solve. Yet, its use maintains cgDG accuracy without altering the formulation in Section 3.1. As noted in Section 5.4, after each mesh adaptation, we compute both projections. $\mathbf{\Pi}^*_{\Omega,p}$ provides the input for the weak form evaluation, while the L^2 projection gives the field at time t for explicit time integration.

Remark. When $N = 1$ and $p_1 \leq p$, both projections coincide since no information is lost. In such cases, projection values are evaluated directly at the new mesh nodes.

Remark. Properties of Cartesian meshes can be exploited to accelerate projection computations, similarly to how they simplify the electromagnetic formulation in cgDG (see Section 3.1).

5.6. Numerical results

In this final section, we present a set of preliminary numerical studies and results that demonstrate the potential of the proposed strategies. The evolution of the two-dimensional systems being solved is governed by Maxwell's equations under TE^z polarization. Regarding time integration, all analyses are carried out using a classical Runge-Kutta integration scheme with a time step small enough to ensure that the error due to time integration is negligible compared to the spatial discretization error.

5.6.1. Studies on the performance of error measures

This first block of numerical results focuses on analyzing the performance of the two error measures introduced in this chapter: the explicit indicator $\mathcal{E}_1$ presented in Section 5.2.1, and the multigrid estimator $\mathcal{E}_2$ developed in Section 5.3.2.

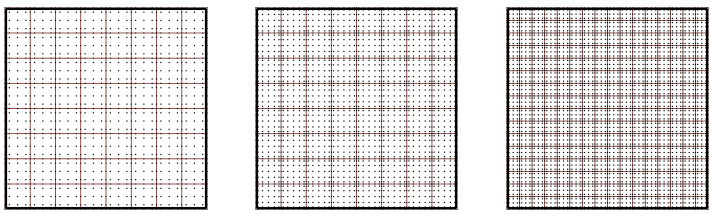
5.6.1.1. Smooth solution in a square domain computed with uniform meshes

The first analysis considers a square domain $\Omega = [0,1]^2$ on which boundary conditions are imposed so that the analytical solution of the problem is given by:

$$\mathbf{U} = \begin{Bmatrix} -\sin(\pi t)\cos(\pi x)\sin(\pi y) \\ \sin(\pi t)\sin(\pi x)\cos(\pi y) \\ -\cos(\pi t)\cos(\pi x)\cos(\pi y) \end{Bmatrix} \tag{5.83}$$

The field evolution is simulated from $t_0 = 0$ to $t_f = 1$, and the error measures are evaluated at the final time.

The computational domain is discretized using three Cartesian meshes with different element sizes and approximation degrees. All meshes are uniform and generated from the same reference domain. The starting point for the cgDG discretization is a 1% scaled version of the original domain, centered on it (*bounding-box* $= [-0.005, 1.005]^2$). This reference domain is discretized using two level-3 Cartesian meshes (8×8 elements) with polynomial degrees 4 and 6, respectively, and a third level-4 Cartesian mesh (16×16 elements) with polynomial degree 4. To identify each mesh, we use the notation $\mathcal{S}_{l,p}$, where the subscripts l and p denote the mesh level and polynomial degree, respectively. Thus, the described meshes correspond to $\mathcal{S}_{3,4}$, $\mathcal{S}_{3,6}$, and $\mathcal{S}_{4,4}$.



(a) $\mathcal{S}_{3,4}$ *(b)* $\mathcal{S}_{3,6}$ *(c)* $\mathcal{S}_{4,4}$

Figure 5.11: *Set of meshes used in the analysis of error measures on a smooth solution in a square domain.*

The purpose of $\mathcal{E}_1$ and $\mathcal{E}_2$ is to guide the adaptation process, so we analyze the distribution of the error measure among the elements. The magnitudes of the error values vary considerably, making the direct visualization of specific values difficult to interpret. Instead, we analyze the clustering of elements according to their error measure.

In Fig. 5.12, we show the classification of elements for each mesh when the clustering is based on $\mathcal{E}_1$ and on $\|\Delta\mathbf{U}\|$, which serves as the reference. It can be observed that the indicator identifies more high-error groups near the boundary. This is because the values of $\mathcal{E}_1$ are slightly higher in these regions, leading to a finer classification. We interpret this result as a consequence of having more information about the exact

field on boundary elements. Incorporating the boundary conditions into the estimator evaluation implies that the residual term on the boundary has a significantly greater contribution compared to when only the jump in the solution between elements is considered. In any case, the use of $\mathcal{E}_1$ as a tool to guide the adaptation process appears reasonable, as the indicator generally captures the distribution of the error. We should only expect slightly finer discretizations near the domain boundaries.

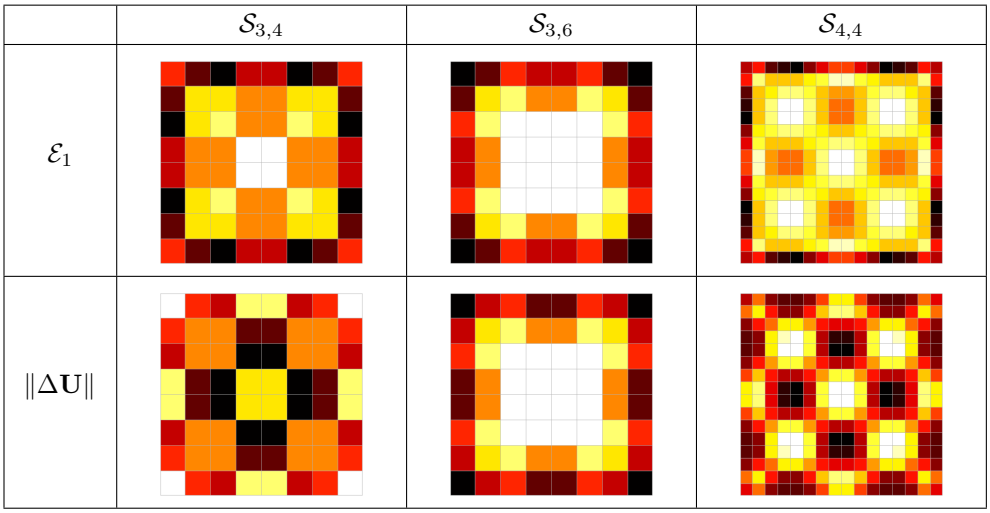


Figure 5.12: *Error-based clustering for a problem with smooth solution in a square domain using the explicit error indicator. (Darker colors indicate higher error levels)*

In Fig. 5.13, the same results are shown for $\mathcal{E}_2$ and its corresponding reference $|||\Delta\mathbf{U}|||$. The first thing we notice is the loss of resolution in the classification compared to the previous case. This is due to the fact that error measurement is performed over 2×2 patches of elements, corresponding to those contained within an element of the coarser mesh level. Following the notation used in Section 5.3.1, the meshes $\mathcal{S}_{3,4}$, $\mathcal{S}_{3,6}$, and $\mathcal{S}_{4,4}$ represented here correspond to $\hat{\mathcal{S}}$ (the computational mesh), while the error is measured over the structure of $\dot{\mathcal{S}}$ (the reference mesh), which would be the meshes $\mathcal{S}_{2,4}$, $\mathcal{S}_{2,6}$, and $\mathcal{S}_{3,4}$, respectively. This becomes particularly evident when observing the clustering of elements for the third mesh (top-right), which reproduces exactly the result obtained for the reference of the first mesh (bottom-left).

In any case, this coarser classification still manages to identify the error distribution. Focusing on the representations of the degree-4 meshes, we can observe that the darker patches group elements with higher error levels. Although guiding the adaptation process with this strategy will not be optimal due to the loss of resolution at the level of the computational mesh, it is still sufficient to identify the regions of the mesh that require modification.

The results for the higher-degree mesh are worse. In fact, refinement is predicted in the central region, which is actually the area with the lowest error. Additional studies

are needed to thoroughly understand this phenomenon, but based on the current information, we associate this behavior with the use of high interpolation degrees. In this example, $\mathcal{S}_{3,6}$ has a total of 3136 nodes, whereas its reference mesh $\mathcal{S}_{2,6}$ has only 784. This large jump in the number of degrees of freedom could be causing a redistribution of the error throughout the domain, which would discourage the use of this strategy when considering large elements with very high-order interpolation.

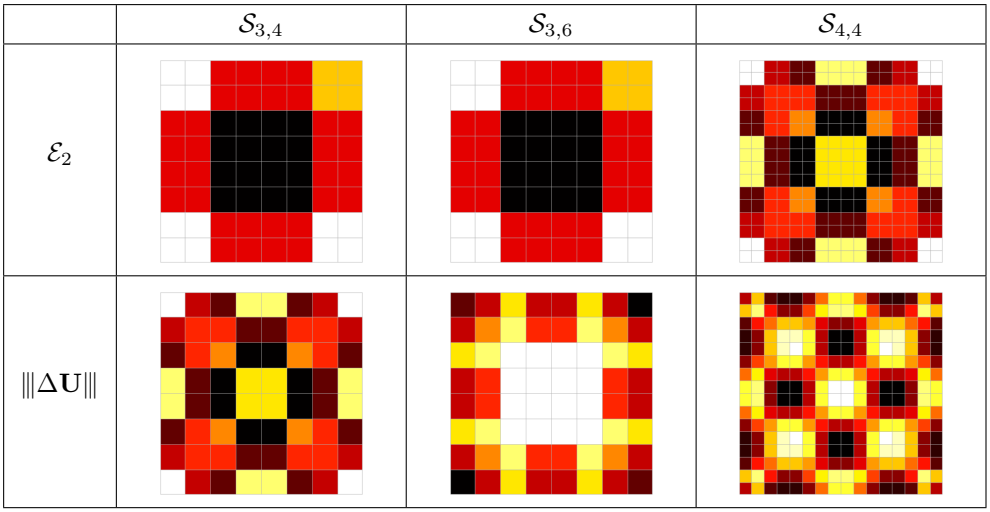


Figure 5.13: *Error-based clustering for a problem with smooth solution in a square domain using the multigrid error estimator. (Darker colors indicate higher error levels)*

Next, we study the global behavior of the proposed strategies. The results for the error indicator shown in Table 5.3 reveal that $\mathcal{E}_1$ does not consistently predict the cgDG solution error. It can only be used to obtain a rough estimate of its order of magnitude. In light of these results, one might question the effectiveness of the parameter $\mathfrak{o}_1$, which is used as a stopping criterion in the adaptation process when this strategy is applied.

Mesh	$\mathcal{E}_1$	$\|\Delta\mathbf{U}\|$	Θ_1
$\mathcal{S}_{3,4}$	2.92×10^{-6}	3.31×10^{-6}	0.88
$\mathcal{S}_{3,6}$	5.69×10^{-8}	2.40×10^{-8}	2.37
$\mathcal{S}_{4,4}$	5.65×10^{-8}	1.07×10^{-7}	0.52

Table 5.3: *Global error measures and effectivity for a problem with smooth solution in a square domain using the explicit error indicator.*

At the level of global predictions, the multigrid strategy outperforms the one based on the explicit indicator. Table 5.4 reports both the estimated and actual

error values on meshes $\dot{\mathcal{S}}$ and $\hat{\mathcal{S}}$, as well as the corresponding effectivities. First, it is worth highlighting the consistency of the results on the reference meshes. The range of variation in the effectivities is narrow and always greater than one, which allows for reliable comparison of the global performance of different discretizations. Furthermore, the good results confirm the validity of the hypothesis (5.34) regarding the local convergence of the solution.

The extrapolation of results from the reference mesh to the computational mesh is satisfactory for the lower-degree discretizations, preserving nearly the same effectivity observed in the coarser mesh. The effectivity value of 0.2 obtained for mesh $\mathcal{S}_{3,6}$ is anomalous and, considering that the errors with this discretization are on the order of 10^{-13}, it could be attributed to numerical round-off errors.

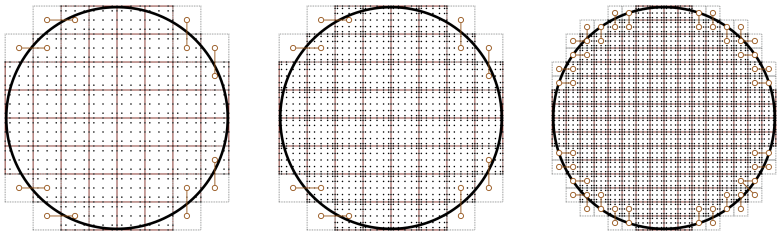
Mesh	$\hat{\mathcal{E}}_2$	$\hat{\mathcal{E}}_2^*$	$\hat{\Theta}_2$	$\dot{\mathcal{E}}_2$	$\dot{\mathcal{E}}_2^*$	$\dot{\Theta}_2$
$\mathcal{S}_{3,4}$	2.72×10^{-9}	1.66×10^{-9}	1.63	1.28×10^{-6}	7.73×10^{-7}	1.65
$\mathcal{S}_{3,6}$	4.41×10^{-14}	2.12×10^{-13}	0.20	2.81×10^{-11}	1.46×10^{-11}	1.92
$\mathcal{S}_{4,4}$	6.04×10^{-12}	3.53×10^{-12}	1.71	2.82×10^{-9}	1.66×10^{-9}	1.69

Table 5.4: *Global error measures and effectivities for a problem with smooth solution in a square domain using the multigrid error estimator.*

5.6.1.2. Smooth solution in the domain of a circle computed with uniform meshes

For the second study, the same analytical solution, evaluation time, and meshes as in the previous case are considered, but in this case the domain Ω is a circle of radius $R = 0.5$ centered at $\mathbf{c} = \{0.5, 0.5\}$. This configuration is used to explore the sensitivity of the error measures to the presence of boundary elements with different cut patterns, some of which require stabilization. Fig. 5.14 shows the computational meshes, with unstable elements represented with dotted lines and connected to the elements used for their stabilization. The conclusions obtained for the different discretizations are similar, so only the results for mesh $\mathcal{S}_{4,4}$ will be discussed.

The clustering results shown in Fig. 5.15 display a behavior of $\mathcal{E}_1$ similar to that in the previous study, with a tendency of the indicator to yield higher values along the boundary. It is verified that the indicator is able to effectively detect extended elements with higher error. However, when using the exact error norm, a greater number of groups appear on the boundary, indicating that the indicator is actually underestimating the error in those regions.



(a) $\mathcal{S}_{3,4}$ *(b)* $\mathcal{S}_{3,6}$ *(c)* $\mathcal{S}_{4,4}$

Figure 5.14: *Set of meshes used in the analysis on error measures for a smooth solution in a circular domain.*

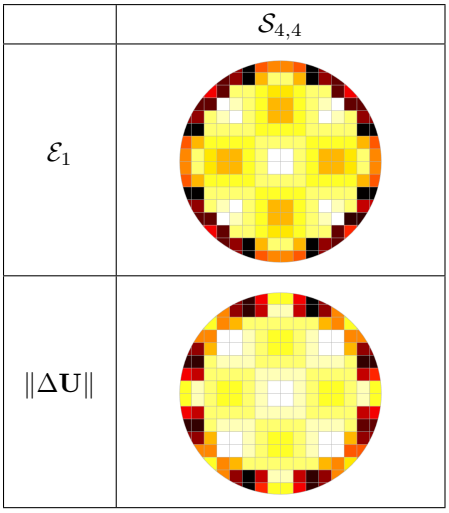


Figure 5.15: *Error-based clustering for a problem with smooth solution in a circular domain using the explicit error indicator. (Darker colors indicate higher error levels)*

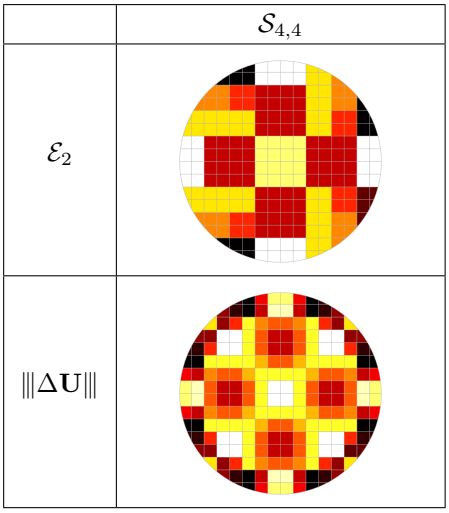


Figure 5.16: *Error-based clustering for a problem with smooth solution in a circular domain using the multigrid error estimator. (Darker colors indicate higher error levels)*

Another indication of this overly optimistic behavior is found when analyzing the results at the global level. In Table 5.5, we see that the effectiveness of the indicator for the mesh $\mathcal{S}_{4,4}$ is only 0.55. Comparing these global results with those shown in Table 5.3, we observe that the indicator tends to underestimate the error level as the approximation degree increases or the mesh level decreases.

In Fig. 5.16, the error distribution predicted by $\mathcal{E}_2$ again reflects the same loss of resolution due to its evaluation on the reference mesh. However, in this case, the differences with respect to the exact distribution over the computational mesh are more significant. If we recall the pairing of elements for stabilization (see Fig. 5.14) and compare it with the error distribution, we observe that the largest discrepancies are located in those zones. On the left side of the domain, the computational mesh

Mesh	$\mathcal{E}_1$	$\|\Delta\mathbf{U}\|$	Θ_1
$\mathcal{S}_{3,4}$	1.15×10^{-5}	7.65×10^{-6}	1.51
$\mathcal{S}_{3,6}$	8.34×10^{-8}	2.94×10^{-8}	2.84
$\mathcal{S}_{4,4}$	2.11×10^{-7}	3.84×10^{-7}	0.55

Table 5.5: *Global error measures and effectiveness for a problem with smooth solution in a circular domain using the explicit error indicator.*

$\mathcal{S}_{4,4}$ requires stabilization, whereas in that same region the reference mesh $\mathcal{S}_{3,4}$ does not. Since the greatest error accumulates in the extended elements, $\mathcal{E}_2$ fails to predict the need to improve the discretization precisely where it is necessary.

In Table 5.6, we observe that the global results for the reference mesh remain quite good, although there is a decrease in effectiveness values, reaching levels below unity. This underestimation of the error level can be attributed to boundary elements in the computational mesh with greater error than expected, thereby invalidating (5.34).

Mesh	$\hat{\mathcal{E}}_2$	$\hat{\mathcal{E}}_2^*$	$\hat{\Theta}_2$	$\dot{\mathcal{E}}_2$	$\dot{\mathcal{E}}_2^*$	$\dot{\Theta}_2$
$\mathcal{S}_{3,4}$	1.12×10^{-8}	9.03×10^{-9}	1.24	9.28×10^{-6}	9.20×10^{-6}	1.01
$\mathcal{S}_{3,6}$	2.94×10^{-13}	1.41×10^{-12}	0.21	9.27×10^{-10}	1.04×10^{-9}	0.89
$\mathcal{S}_{4,4}$	2.43×10^{-11}	3.94×10^{-11}	0.62	8.90×10^{-9}	9.06×10^{-9}	0.98

Table 5.6: *Global error measures and effectiveness for a problem with smooth solution in a circular domain using the multigrid error estimator.*

We explain the reasoning in more detail. When an interior element of the mesh is refined in h, its domain is subdivided, and the discretization is enriched by reducing the support domain of the interpolation functions by a factor of 0.25. In contrast, the support of the interpolation functions in cut elements depends not only on the element size but also on its position relative to the domain. As a result, when a cut element is refined in h, the support of the interpolation functions is not actually reduced by the expected factor of 0.25. Ultimately, the improvements provided by the refined mesh will not be as significant as one might expect. This effect can be exacerbated when considering domain extensions introduced for stabilization purposes.

The influence of boundary elements on the results over the computational mesh is considerable. We attribute the observed loss of performance to the fact that the extrapolation process takes into account the domains associated with elements before stabilization. This configuration does not reflect the actual structure of the mesh and represents an aspect to be improved in the current implementation.

5.6.1.3. Smooth solution in a square domain computed with a non-uniform mesh

In this third example of the section, we study the performance of the error indicators for the same problem as in Section 5.6.1.1, but now using a stationary mesh $\mathcal{S}_{\mathbf{l},\mathbf{p}}$ with elements of varying sizes and polynomial degrees. The mesh is constructed from the same bounding box $[-0.005, 1.005]^2$ and is generated with a random distribution of element sizes and degrees, ensuring compatibility with both error estimation strategies. The mesh used in this study is shown in Fig. 5.17, and it contains elements with mesh levels and polynomial degrees in the ranges $[3, 8]$ and $[2, 6]$, respectively.

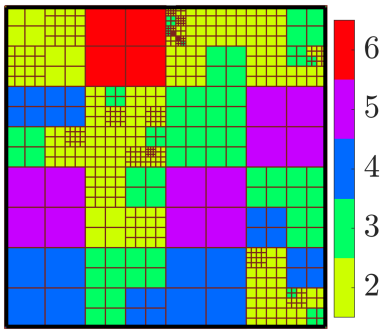


Figure 5.17: *Non-uniform stationary mesh for a square domain.*

Figs. 5.18 and 5.19 show the results of the element classification based on the error measures. The multigrid strategy maintains the same behavior as in the uniform mesh case and is capable of correctly identifying the error distribution, following the same patterns as the classification based on its reference. The explicit error indicator strategy appears to be more affected by local variations in the discretization, although it still captures the overall trends. In this example, where the references for both strategies yield more divergent classifications, it is interesting to observe that the results obtained with $\mathcal{E}_1$ lie midway between the two.

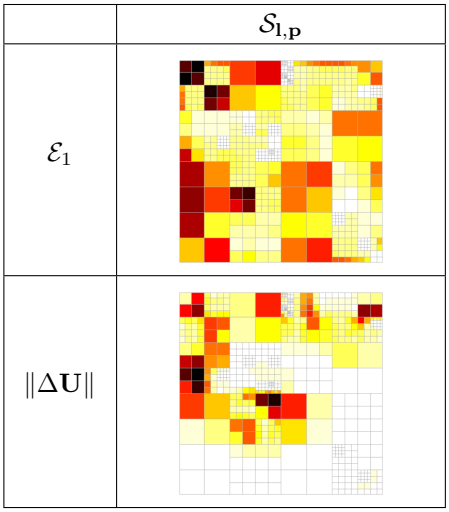


Figure 5.18: *Error-based clustering for a problem with smooth solution on a non-uniform stationary mesh in the context of the explicit error indicator. (Darker colors indicate higher error levels)*

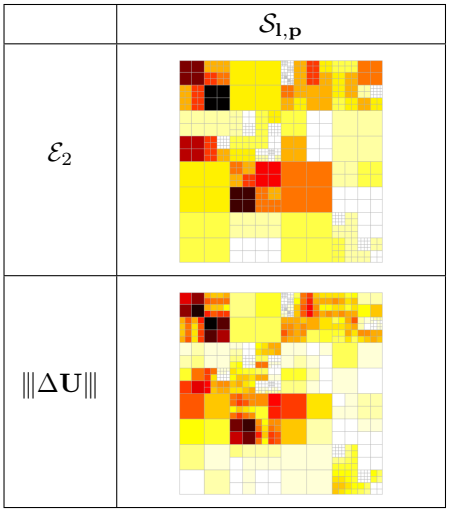


Figure 5.19: *Error-based clustering for a problem with smooth solution on a non-uniform stationary mesh in the context of the multigrid error estimator. (Darker colors indicate higher error levels)*

The global results presented in Tables 5.7 and 5.8 confirm the suspicions raised after analyzing the error distribution results. The effectiveness of the indicator $\mathcal{E}_1$ drops significantly. In contrast, the multigrid strategy provides good performance both on the reference mesh and in the extrapolation to the computational mesh. This favorable behavior is attributed to the local nature of the DG formulation, which enabled the convergence analysis presented in Section 5.3.4 to be carried out at the element level. Furthermore, the effectiveness values are very similar to those obtained for meshes $\mathcal{S}_{3,4}$ and $\mathcal{S}_{4,4}$ in the first analysis. This comparison highlights the robustness of $\mathcal{E}_2$ and reinforces the idea that the results obtained for $\mathcal{S}_{3,6}$ were anomalous.

Mesh	$\mathcal{E}_1$	$\|\Delta\mathbf{U}\|$	Θ_1
$\mathcal{S}_{\mathbf{l},\mathbf{p}}$	1.03×10^{-5}	1.54×10^{-4}	0.07

Table 5.7: *Global error and effectivity measures for a problem with smooth solution on a non-uniform stationary mesh in the context of the explicit error indicator.*

Mesh	$\hat{\mathcal{E}}_2$	$\hat{\mathcal{E}}_2^*$	$\hat{\Theta}_2$	$\dot{\mathcal{E}}_2$	$\dot{\mathcal{E}}_2^*$	$\dot{\Theta}_2$
$\mathcal{S}_{\mathbf{l},\mathbf{p}}$	6.15×10^{-7}	3.88×10^{-7}	1.59	2.11×10^{-5}	1.26×10^{-5}	1.68

Table 5.8: *Global error and effectivity measures for a problem with smooth solution on a non-uniform stationary mesh in the context of the multigrid error estimator.*

5.6.1.4. Discontinuous solution in a square domain computed with a non-uniform mesh

In this final analysis of the results block, we evaluate the capacity of the proposed strategies to operate in a context where the solution is not smooth. To this end, we use the same non-uniform mesh and square domain from the previous example, but in this case, the simulated field follows the expression:

$$\mathbf{U}(\mathbf{x},t) = \begin{Bmatrix} 0 \\ g(x,t) \\ g(x,t) \end{Bmatrix} \tag{5.84}$$

with

$$g(x,t) = \begin{cases} 0 & \text{if } t-(x-x_0) \leq 0 \\ \cos\left(2\pi(t-x)\right)\sin\left(\frac{\pi}{4L}(t-(x-x_0))\right) & \text{if } t-(x-x_0) \in [0,L] \\ \cos\left(2\pi(t-x)\right) & \text{if } t-(x-x_0) > L \end{cases} \tag{5.85}$$

which represents a planar wave of unit wavelength and unit speed propagating from left to right. At $t = 0$, it has zero amplitude to the right of x_0, increasing to the left until it reaches the value of a unit-amplitude wave at a distance L from x_0. This is a field compatible with Maxwell's equations without requiring any source term, but for small values of L, it exhibits a sharp amplitude variation.

We simulate the field (5.84) using parameters $x_0 = 0.5$ and $L = 10^{-3}$, over the time interval $t_0 = 0$ to $t_f = 10^{-3}$. This configuration implies that the solution will initially be null in the right half of the mesh and evolves such that the wavefront enters the first layer of elements to the right of $x = 0$, which should approximately reflect the maximum field amplitude on their left side at the end of the simulation.

The results in Figs. 5.20 and 5.21 show that the error is concentrated in the elements that receive the wavefront. It is also observed that the results in Fig. 5.20 exhibit greater dispersion than those in Fig. 5.21. This is attributed to the fact that the magnitudes computed in the second case are more closely related to the DG formulation. Consequently, they reflect an upwinding nature, with the error propagating from left to right. Regarding the performance of the error measurement strategies, it is very similar to that observed in the previous case, with both approaches correctly capturing the error distribution.

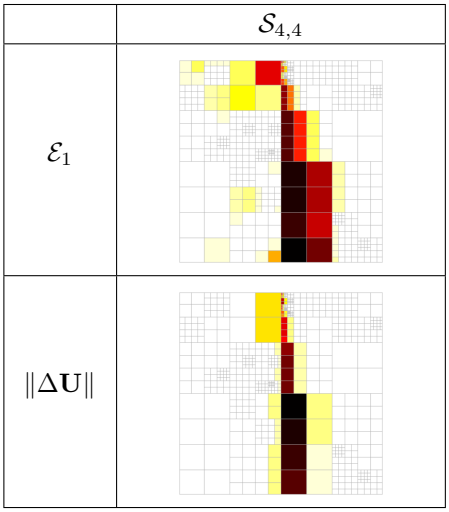


Figure 5.20: *Error-based clustering for a problem with discontinuous solution on a non-uniform stationary mesh in the context of the explicit error indicator. (Darker colors indicate higher error levels)*

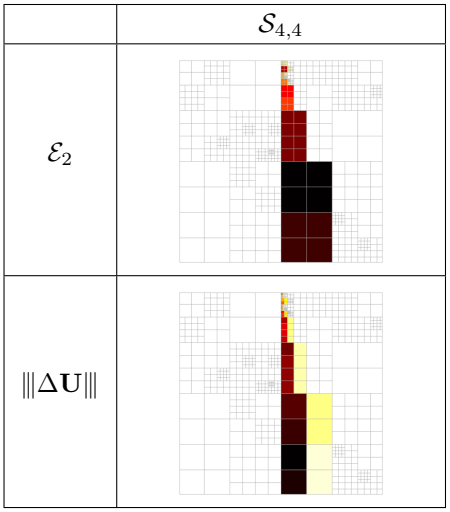


Figure 5.21: *Error-based clustering for a problem with discontinuous solution on a non-uniform stationary mesh in the context of the multigrid error estimator. (Darker colors indicate higher error levels)*

As shown in Tables 5.9 and 5.10, the lower regularity of the solution negatively impacts the global results. The loss of performance in the multigrid-based estimator is particularly significant. Poorer results in the extrapolation could be expected, since the evaluation using (5.69) does not account for the discontinuity in the solution, and the exponent is always computed based on the interpolation degree. Indeed, this loss in estimation effectiveness is also observed in the results. However, more relevant is the decrease in the effectiveness of the error estimate on the reference mesh, which drops to 0.24. Therefore, in the context of problems with low-regularity solutions, neither strategy enables a reliable prediction of the global error present in the solution.

Mesh	$\mathcal{E}_1$	$\|\Delta\mathbf{U}\|$	Θ_1
$\mathcal{S}_{\mathrm{l},\mathbf{p}}$	4.95×10^{-2}	2.09×10^{-2}	2.36

Table 5.9: *Global error and effectivity measures for a problem with discontinuous solution on a non-uniform stationary mesh in the context of the explicit error indicator.*

Mesh	$\hat{\mathcal{E}}_2$	$\hat{\mathcal{E}}_2^*$	$\hat{\Theta}_2$	$\dot{\mathcal{E}}_2$	$\dot{\mathcal{E}}_2^*$	$\dot{\Theta}_2$
$\mathcal{S}_{\mathrm{l},\mathbf{p}}$	1.41×10^{-2}	6.12×10^{-1}	0.02	2.10×10^{-1}	8.66×10^{-1}	0.24

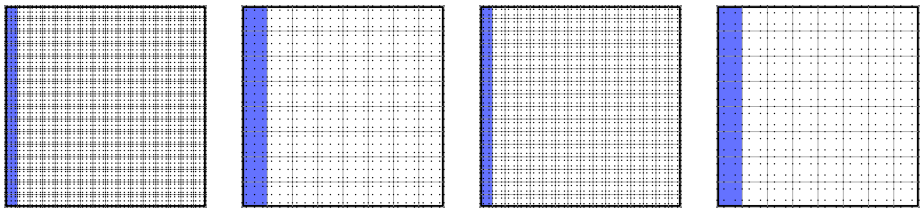
Table 5.10: *Global error and effectivity measures for a problem with discontinuous solution on a non-uniform stationary mesh in the context of the multigrid error estimator.*

5.6.1.5. General conclusions

The numerical studies presented thus far show that both proposed error measurement techniques are capable of identifying the error distribution in the discretization and are suitable for guiding mesh adaptation strategies. The indicator $\mathcal{E}_1$ has advantages in capturing local variations in the error but tends to penalize boundary elements compared to interior ones. The estimator $\mathcal{E}_2$ enables a robust quantification of global errors on both the reference mesh and the computational mesh when the solution is smooth. Furthermore, it provides a consistent prediction of the error distribution even when using meshes with elements of highly varying sizes and approximation degrees. However, this prediction breaks down when the computational mesh includes very high-degree elements.

5.6.2. Study on the performance of regularity indicators

The second analysis we propose focuses on the parameters $\mathfrak{r}$ used in the *hp*-refinement criterion. The computational setup again considers the square domain $\Omega = [0, 1]^2$ and uniform meshes within a bounding box extending 0.5% beyond the boundaries of Ω in all directions. In this study, we use the four uniform meshes shown in Fig. 5.22, which contain level-3 and level-4 elements with interpolation degrees 3 and 4.



(a) $\mathcal{S}_{4,4}$ *(b)* $\mathcal{S}_{3,4}$ *(c)* $\mathcal{S}_{4,3}$ *(d)* $\mathcal{S}_{3,3}$

Figure 5.22: *Set of meshes used in analysis of the regularity indicators*

The problem considered is a plane wave described by the expression (5.84) with parameters $x_0 = 0$ and $L = 2 \times 10^{-3}$. The simulation starts at $t_0 = 0$ and the field is allowed to evolve. Note that at the initial time, the field is zero throughout the domain and can thus be initialized without any error. As mentioned, we are interested in the behavior of $\mathfrak{r}_1$ and $\mathfrak{r}_2$. Specifically, we will study the values of these parameters for the leftmost layer of elements in the mesh (see blue shaded elements in Fig. 5.22); that is, the layer containing the boundary through which the wave enters Ω. The exact solution exhibits the same characteristics inside all these elements, and ideally, the regularity indicators should be equally uniform. To facilitate analysis, we

will study the evolution of the mean value and standard deviation of the parameters within this first layer of elements for all four meshes.

The initial results are retrieved immediately to study the effect of the incident wavefront. Let h_l denote the element side length for a mesh of level l. The time instants t_1, t_2, and t_3 selected for the analysis correspond to the integration time step Δt, the time at which the wavefront has advanced a distance equal to h_8, and the time at which it reaches $x = h_7$, respectively. For the studied elements in the level-4 meshes, this means the wave enters the elements, progresses to $1/16$ of their width, and then to $1/8$ of their width, respectively. The field is then allowed to evolve further, and when the front reaches $x = 3\,h_4$, results are recorded at t_4, t_5, and t_6 following the same time pattern. Fig. 5.23 presents a schematic showing the position of the wavefront at each of these time instants with respect to the domain boundary, using the interfaces of the level-4 mesh elements as a reference.

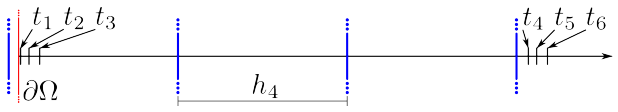
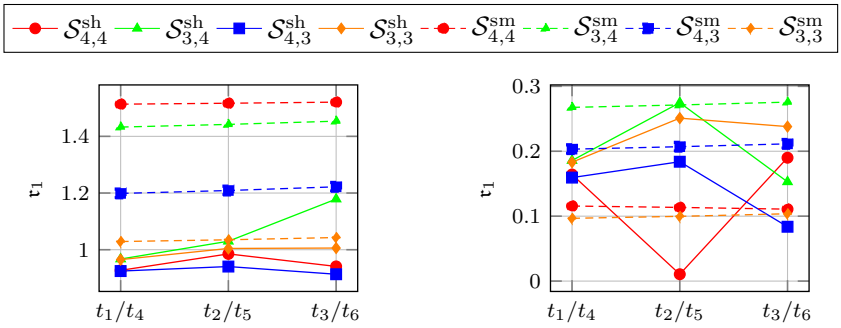


Figure 5.23: *Graphical representation of the temporal scheme used for the analysis of the regularity indicators*

The results obtained at t_1, t_2, and t_3 will be used to characterize the behavior of the regularity indicators when the solution exhibits discontinuities, and will be identified in the plots with solid lines and in the notation using the superscript "sh". Conversely, the results at t_4, t_5, and t_6 will be representative of a smooth solution, will be plotted with dashed lines, and denoted with the superscript "sm". For clarity, we define $\mathcal{T}^{\mathrm{sm}}$ and $\mathcal{T}^{\mathrm{sh}}$ as the sets $\{t_1, t_2, t_3\}$ and $\{t_4, t_5, t_6\}$, respectively.

We begin by analyzing the behavior of $\mathfrak{r}_1$. Fig. 5.24a shows that the parameter indeed tends to attain higher values over $\mathcal{T}^{\mathrm{sh}}$. If each discretization is analyzed individually, the indicator is always greater over $\mathcal{T}^{\mathrm{sh}}$. However, when considering all discretizations together, it is not possible to define a single threshold that reliably distinguishes when the solution is smooth. Furthermore, Fig. 5.24b shows that the variability in the results is comparable to the differences observed between $\mathcal{T}^{\mathrm{sh}}$ and $\mathcal{T}^{\mathrm{sm}}$. This more detailed analysis anticipates that the parameter $\mathfrak{r}_1$ will not consistently identify smooth fields, and therefore the refinement type selected will not always be optimal. In any case, based on these results and the experience gained from other analyses, a threshold value of $\mu_{\mathfrak{r}} = 1.05$ in the *hp*-refinement criterion yields reasonably good adaptation performance.

The indicator $\mathfrak{r}_2$ yields more promising results. In Fig. 5.25a, it can be seen that the mean values obtained over $\mathcal{T}^{\mathrm{sh}}$ are now significantly higher than those over $\mathcal{T}^{\mathrm{sm}}$. Even for the mesh $\mathcal{S}_{3,3}$, which yields the lowest values over $\mathcal{T}^{\mathrm{sh}}$ and the highest over $\mathcal{T}^{\mathrm{sm}}$, a clear distinction can still be made between the values of $\mathfrak{r}_2$ corresponding to a smooth solution and those with discontinuities. This is illustrated in Fig. 5.26a. As for the standard deviation values shown in Figs. 5.25b and 5.26b, they are sufficiently

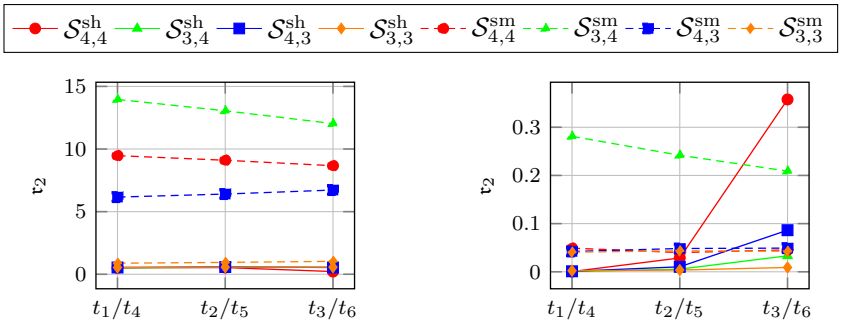


(a) *Mean value* ***(b)*** *Standard deviation*

Figure 5.24: *Evolution of the regularity indicator within the explicit indicator strategy*

small to ensure that the parameter $\mathfrak{r}_2$ can reliably guide the *hp*-refinement of the mesh. With a threshold value $\mu_{\mathfrak{r}} = 0.8$, the refinement strategy should be able to operate optimally in all cases.

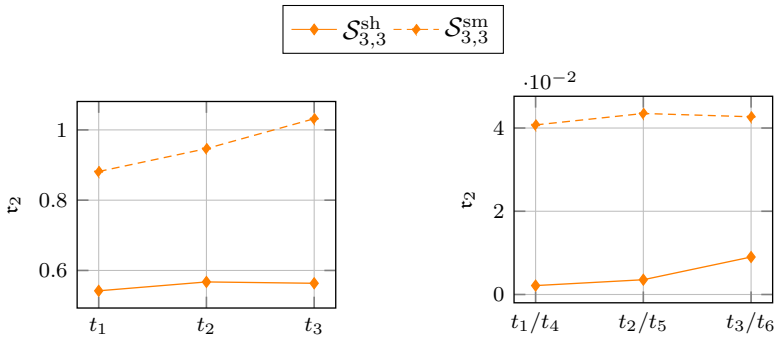
It is important to emphasize that the derivations leading to (5.71) relied on error estimation results that are valid in the asymptotic regime. Nevertheless, the parameter $\mathfrak{r}_2$ successfully identifies the regularity of the solution even on the mesh $\mathcal{S}_{3,3}$. The meshes involved in this computation are $\dot{\mathcal{S}} = \mathcal{S}_{2,3}$ and $\check{\mathcal{S}} = \mathcal{S}_{2,2}$, both of which are very coarse discretizations, containing only 256 and 144 nodes, respectively.



(a) *Mean value* ***(b)*** *Standard deviation*

Figure 5.25: *Evolution of the regularity indicator within the multigrid strategy*

At a general level, it is worth noting that no significant reduction in performance was observed for either parameter as time advanced. This result is to be expected, since the discretization tends to smooth the solution, and such behavior should be reflected in the indicators. Further studies are needed to determine to what extent $\mathfrak{r}_1$ and, in particular, $\mathfrak{r}_2$ are insensitive to this effect.



(a) *Mean value* ***(b)*** *Standard deviation*

Figure 5.26: *Evolution of the regularity indicator within the multigrid strategy for the coarser mesh*

5.6.3. Study on the performance of *hp*-adaptation strategy

The final two studies included in this chapter focus on the overall performance of the *hp*-mesh adaptation strategy. These studies will employ the threshold values $\mu_{\mathfrak{r}}$ recommended in the previous section.

5.6.3.1. Shock wave in squared domain

In the first study of this block, we solve the same problem posed in Section 5.6.2, using the mesh $\mathcal{S}_{3,3}$ as the initial discretization. In this case, the analysis covers the time interval from $t_0 = 0$ to $t_f = 3T$, and during the simulation, the mesh is adapted following the procedure described in Section 5.4, with the number of nodes in the adapted meshes constrained to the range [1000, 1250] ($\mathcal{S}_{3,3}$ contains 1024 nodes).

The objective of the study is to assess the impact of the two remaining user-defined parameters: the refinement delay $\mu_{\Delta t}$ and the threshold for the stopping criterion $\mu_{\mathfrak{o}}$. We analyze the evolution of the maximum element-wise error, measured in the L^2 norm, under the parameter sets $\mu_{\Delta t} = \{f_0, f_{32}, f_{128}, f_{512}\}$ and $\mu_{\mathfrak{o}} = \{0.9, 1.001, 1.1\}$. A refinement delay f_n means that the mesh will be adapted n times per wave period. Given that the analysis was carried out using a time step of 2^{-10}, the highest refinement frequency corresponds to adapting the mesh every two time integration steps, while the lowest frequency corresponds to a simulation without mesh adaptation.

The analysis of the results is approached in two stages: first, we study the evolution of the error over the time interval $[0, T]$, that is, while the wave front traverses the computational domain (see Fig. 5.27); and then we analyze the performance of the adaptation strategy once the field is fully developed (see Fig. 5.28). In the tables and

throughout the analysis, the adaptation procedures based on the explicit indicator and the multigrid strategy will be referred to as $\mathfrak{A}_1$ and $\mathfrak{A}_2$, respectively.

Regarding the results in the interval $[0, T]$, it can generally be stated that $\mathfrak{A}_2$ performs better than $\mathfrak{A}_1$, especially towards the end of the interval. In most cases, the errors of the adapted solutions are lower at the early stages of the simulation and tend to increase as time progresses. This outcome stems from the greater availability of nodes at the beginning of the simulation, which the algorithm can assign to the region where the field is active, thus densifying the discretization in the region of zero field. Later, the algorithm must redistribute the nodes that were initially concentrated, and this is when its performance becomes compromised. The jumps observed in the error for $\mathfrak{A}_1$ can be explained in this way—they are caused by the wave front entering a large element that was not properly h-refined. In fact, using $\mathfrak{A}_1$ turns out to be counterproductive when compared to the results obtained with the stationary mesh. This behavior is not observed in $\mathfrak{A}_2$, which provides more consistent results, particularly at the highest refinement frequency.

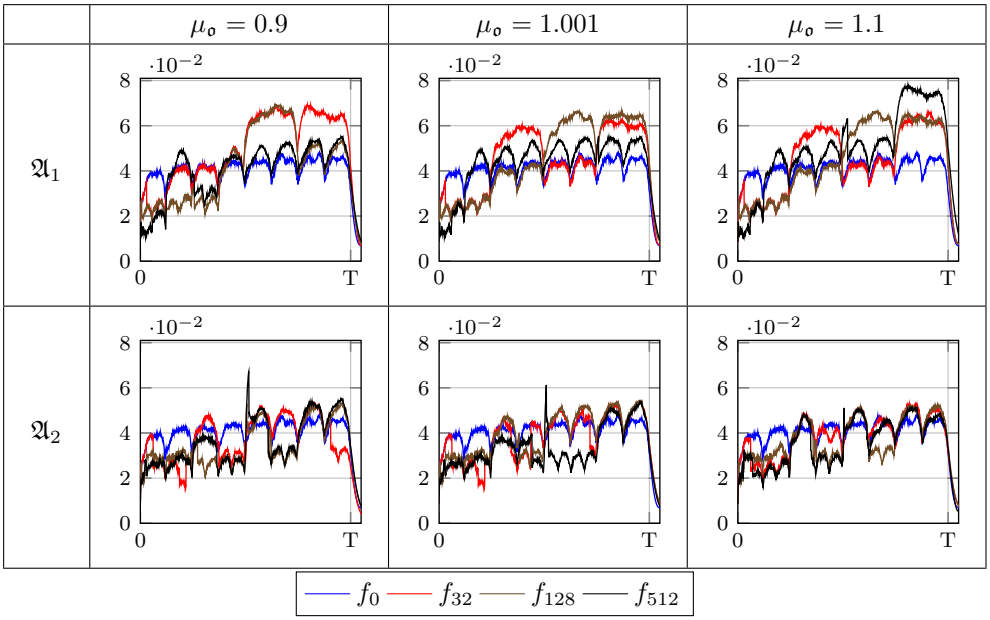


Figure 5.27: *Evolution of* $\|\Delta \boldsymbol{U}\|_{\Omega_e}^{\max}$ *due to a wave front traversing the domain*

Overall, the effect of the parameter μ_o is relatively mild. For $\mathfrak{A}_1$, the results corresponding to f_{512} worsen as μ_o increases. For the same refinement frequency, the results for $\mathfrak{A}_2$ seem to suggest that a refinement strategy that allows more freedom to modify the mesh (higher μ_o) can be beneficial in the later stages. However, setting μ_o to a value close to one appears to be advantageous.

Except for the case $\mu_o = 1.1$ with $\mathfrak{A}_1$, using the highest adaptation frequency yields better overall results. Nevertheless, for intermediate frequencies, performance is quite similar and irregular over the interval, especially for $\mathfrak{A}_1$.

We now turn to the analysis of the results after the wave front has exited the computational domain. The plots in Fig. 5.28 show that, in no case, is an adapted mesh able to match the error level of the uniform mesh. This is understood to be due to the fact that, in order to reach the uniform mesh configuration, the adaptation procedure must go through intermediate meshes that are less accurate than the discretizations reached at the end of the previous interval.

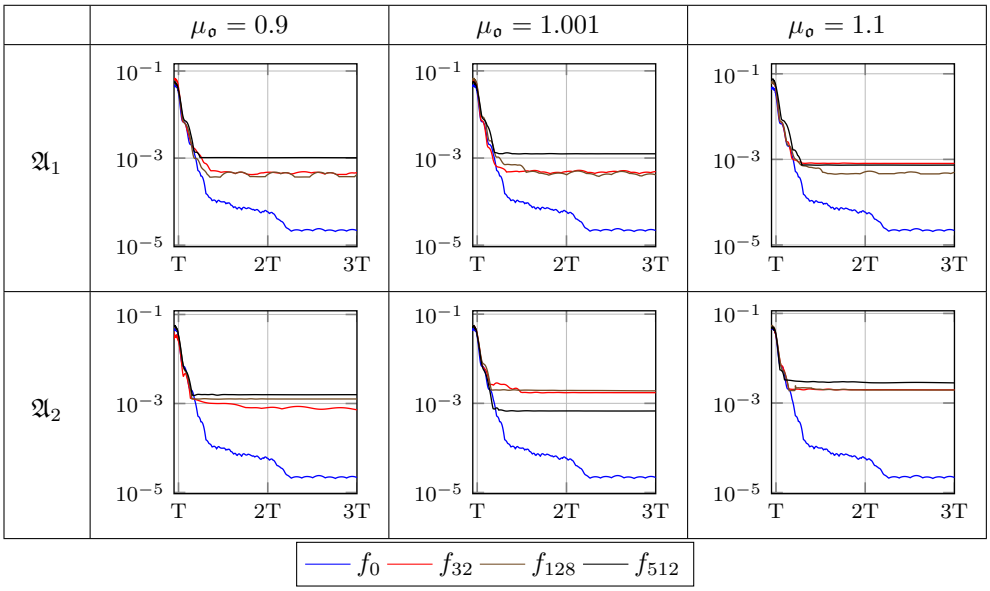


Figure 5.28: *Evolution of* $\|\Delta \boldsymbol{U}\|_{\Omega_e}^{\max}$ *once a wave front has left the domain*

It is interesting to observe that intermediate refinement frequencies tend to yield slightly better results, whereas f_{512} almost always leads to the worst performance. For $\mathfrak{A}_1$, the simulation with f_{128} exhibits the best behavior across all three configurations. The superior performance of these intermediate frequencies in the regime with a fully developed wave can be explained by the adaptation behavior during the previous stage. If the algorithm avoids refining the mesh around the wave front, two advantages are gained for the smooth-field stage: on one hand, the stage begins with a more uniform distribution of nodes, and on the other, the front will have been smoothed out, allowing the discretization to adapt to a more regular solution than the exact one.

In this regime, the variability of the results with respect to μ_o does not allow us to identify any clear trend.

Finally, we comment on the error evolution observed on the stationary mesh. We note that the error induced by the passage of the wave front contaminates the solution for a significant period after it has left the domain. In fact, we observe that the uniform mesh does not reach a stabilized error until approximately $2.25T$.

Figs. 5.29 and 5.30 show how the mesh nodes are distributed throughout the simulation across different element levels and polynomial degrees, respectively. These distributions correspond to the simulation using $\mu_{\Delta t} = f_{512}$ and $\mu_{\mathfrak{o}} = 1.001$, although the conclusions can be extrapolated to the other cases. First, it is observed that the adaptation strategy eventually proposes meshes at the upper end of the allowed node range, indicating that the algorithm correctly identifies that discretizations with more degrees of freedom yield lower errors. It is also observed that $\mathfrak{A}_2$ temporarily exceeds the maximum allowed number of nodes. This is due to the fact that adapting over the reference mesh results in larger jumps in the computational mesh than would occur if the adaptation were performed directly on the latter. This same mechanism explains the greater variability experienced by the mesh in $\mathfrak{A}_2$. Especially during the early stages of the simulation, significant fluctuations in the number of nodes associated with different element sizes and approximation degrees are observed. During this initial stage, the greater flexibility of $\mathfrak{A}_1$ is also apparent, as it proposes meshes with elements ranging from level 1 to 5. A mesh with such characteristics in $\mathfrak{A}_2$ is unlikely, as the presence of a level-1 element in the computational mesh would imply that the reference mesh consists of a single element.

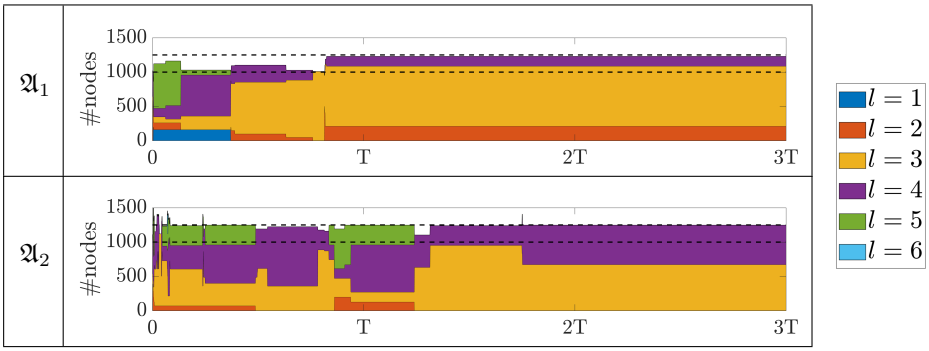


Figure 5.29: *Evolution of the number of nodes grouped by element's levels for* $\mu_{\Delta t} = f_{512}$ *and* $\mu_{\mathfrak{o}} = 1.001$

Finally, it is worth noting that $\mathfrak{A}_1$ stabilizes the discretization earlier than $\mathfrak{A}_2$, in most cases even before the wave front exits the computational domain. This rapid stabilization is achieved regardless of the value of $\mu_{\mathfrak{o}}$, which—within the studied range—does not appear to control the convergence rate toward the stationary discretization. For $\mathfrak{A}_2$, there is a suggestion that increasing $\mu_{\mathfrak{o}}$ delays the convergence to a stationary mesh. However, the results are inconclusive, and therefore not shown.

Lastly, Fig. 5.31 shows the evolution of the mesh and the electromagnetic energy density for the simulation with $\mu_{\Delta t} = f_{128}$ and $\mu_{\mathfrak{o}} = 1.001$. We observe that $\mathfrak{A}_2$

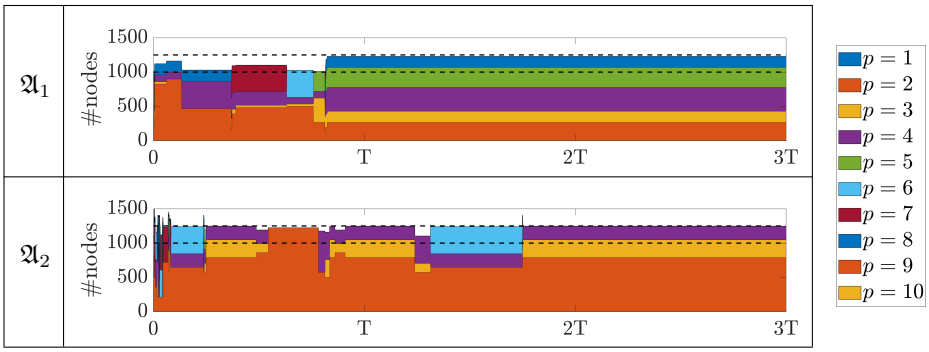
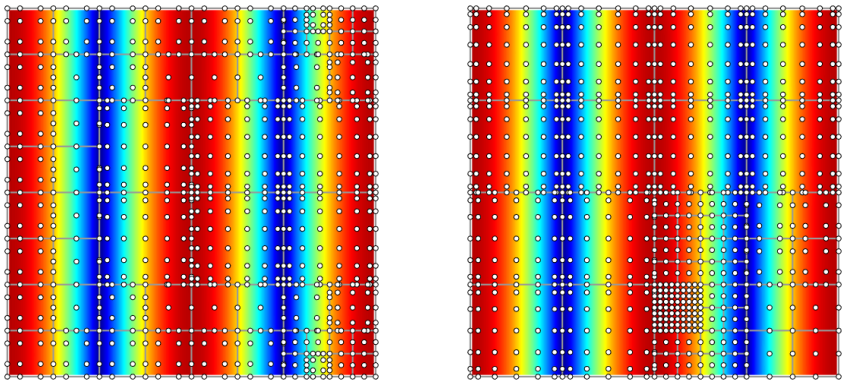


Figure 5.30: *Evolution of the number of nodes grouped by element's degrees for* $\mu_{\Delta t} = f_{512}$ *and* $\mu_{\mathfrak{o}} = 1.001$

correctly identifies the position of the wave front and concentrates the nodes around it. Only near time T does the lack of available degrees of freedom lead to coarser discretizations in some areas than would be desirable. Also noteworthy is the concentration of degrees of freedom around the wave front and their subsequent redistribution once the solution enters its stationary phase. This behavior is not observed in $\mathfrak{A}_1$. However, the alternative based on the explicit indicator manages to preserve the symmetry of the mesh, which $\mathfrak{A}_2$ does not.



(a) $\mathfrak{A}_1$ ***(b)*** $\mathfrak{A}_2$

Figure 5.31: *Mesh and electromagnetic energy density evolution in time with hp-adaptation with* $\mu_{\mathfrak{o}} = 1.001$ *and* $\mu_{\Delta t} = T/128$ *as parameters of the hp-adaptation strategy (click on the figures to show the animation, available when using Adobe Acrobat)*

The application of the mesh adaptation strategy to a problem with a discontinuous solution has yielded promising results, especially for $\mathfrak{A}_2$. Although further research is needed to improve the adaptation during the stationary phase, the design of meshes for this type of solution is relatively straightforward and was not the main goal of the

proposed adaptation strategy. It would be worthwhile to investigate the behavior of the method in a context with richer discretizations, where the procedure would have more freedom to modify the mesh locally.

5.6.3.2. NACA0012 profile

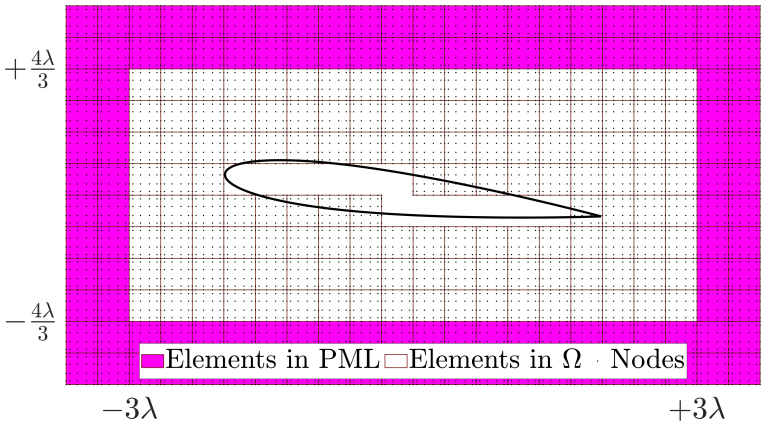
In this final analysis, we solve the scattering problem of a PEC NACA 0012 profile with a chord length of 4λ when it is struck by a plane wave propagating at an angle of 10° with respect to the profile's axis of symmetry. This is a problem similar to the one addressed in Section 3.6.3.1, which largely motivated the development of automatic *hp*-adaptation strategies.

The configuration solved corresponds to that shown in Fig. 5.32, where the profile is tilted at an angle of 7.5° with respect to the horizontal and centered in a rectangular domain Ω of dimensions $6\lambda \times 8\lambda/3$. This domain is discretized using a mesh of 18×8 elements of 5th order. To improve the absorption of reflected waves at the outer boundary of Ω, a PML consisting of two layers of elements of the same size and degree is added. The inclination of the profile causes both the trailing and leading edges to fall inside mesh elements (see Figs. 5.32b and 5.32c), which severely compromises the accuracy of the results. This initial discretization $\mathcal{S}_0$ contains a total of 252 elements and 9072 nodes.

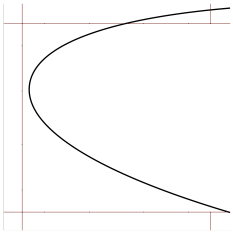
The reference solution will be taken as the one computed on a fine mesh with a total of 17656 nodes, which has been manually *hp*-refined around both the trailing and leading edges. This mesh was constructed by aligning the symmetry axes of the mesh and the geometry, significantly enhancing the solution's accuracy in the most challenging regions. A representation of this mesh is shown in Fig. 5.33.

The problem is solved exclusively for the reflected field, and the quantity of interest to be obtained is the scattered width (SW). At the initial time $t_0 = 0$, the incident wave is assumed to extend across the entire domain, and the evolution of the reflected field (initially zero) is simulated until convergence of the SW is reached with a tolerance of 10^{-3}.

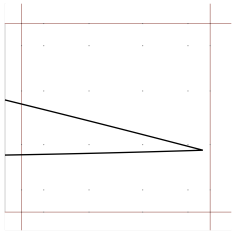
The same problem is solved eight times, starting from the initial mesh $\mathcal{S}_0$ and applying the automatic *hp*-adaptation procedure with parameters $\mu_{\Delta t} = \{f_0, f_{10}, f_{20}, f_{40}\}$ and the two error estimation strategies. Relatively low adaptation frequencies are chosen since the mesh is ultimately expected to be governed by the geometry, whose sharp angle is the source of the singularities. Furthermore, in this case, the focus lies on the stationary phase of the problem, making it unnecessary to accurately capture the evolution of the wavefront from the profile to the outer boundary. To reduce computational cost, the adaptation process is configured to start once the field is already partially developed. Using the user parameter μ_{t0} defined in Section 5.1, the simulation sets $\mu_{t0} = 2T$. A lower value of the parameter $\mu_{\mathbf{o}}$ is also adopted, specifically $\mu_{\mathbf{o}} = 0.9$, in order to stabilize the adaptation process earlier and avoid over-refinement in regions where it is not strictly necessary. The range of node counts allowed for the *hp*-adapted meshes is set to $[9000, 9500]$.



(a) *Global view*

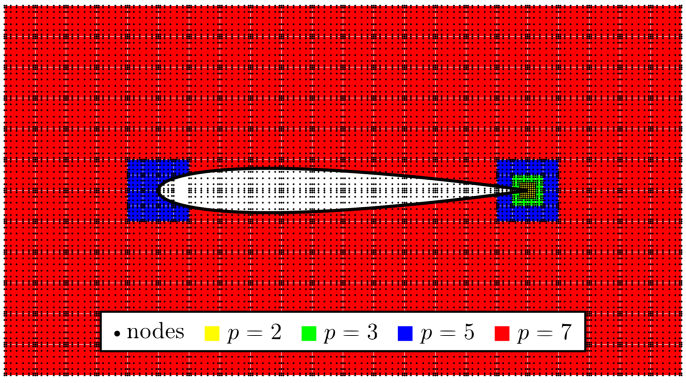


(b) *Detail on the leading edge*

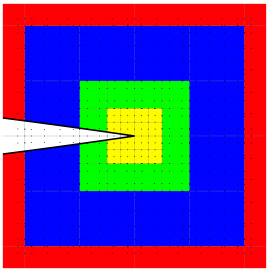


(c) *Detail on the trailing edge*

Figure 5.32: *Setup for the scattering of a 4λ NACA0012 profile*



(a) *Global view*



(b) *Detailed view on user's hp-refinement arround trailing edge*

Figure 5.33: *Fine mesh used to compute the reference solution of the problem*

In none of the simulations employing the error indicator-based strategy $\mathfrak{A}_1$ was it possible to achieve convergence of the SW. In fact, during the simulation with $\mu_{\Delta t} = f_{20}$, an instability was detected in the lower-central region of the profile (see Fig. 5.34). It has been verified that the source of this instability does not originate from the time integration step, but rather stems from another cause. Further studies are required to better understand this phenomenon, although it is suspected that the origin may be related to domain extensions. Similar issues were observed during the computations for the reference solution, which disappeared once the discretization allowed elements with a smaller percentage of their domain over Ω. In Fig. 5.34, it can be seen that the configuration used in $\mathfrak{A}_1$ results in a mesh with large elements, which may at some point have led to an unstable extrapolation of the solution in one such element.

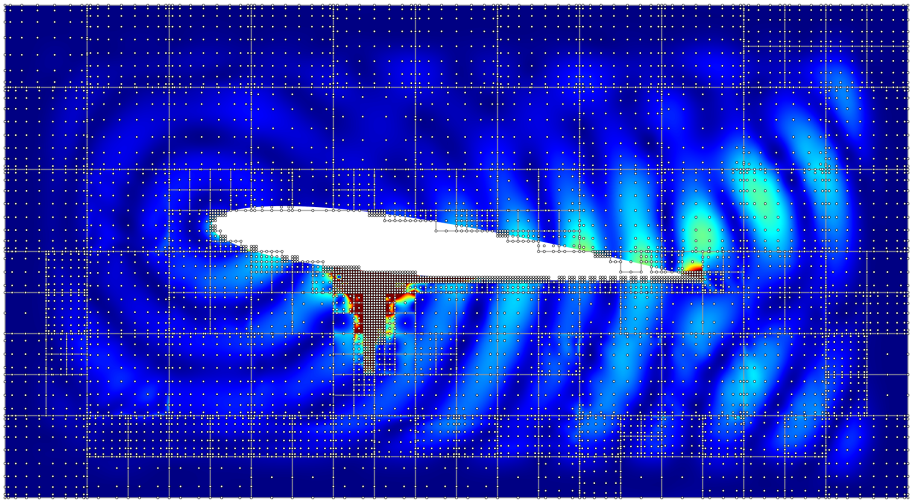
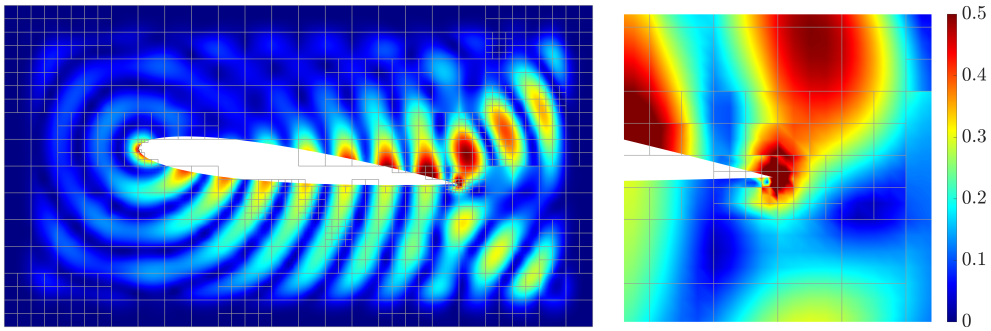


Figure 5.34: *Electromagnetic energy density field at $t = 20T$ showing inestability issues when using explicit hp-adaptation scheme.*

Aside from the instability, the mesh produced by this first adaptation strategy exhibits promising characteristics. In general, it is composed of large elements with high interpolation degrees, which is the most suitable approach when the solution is smooth. In the vicinity of the singularity, a more aggressive refinement is also observed, focusing on the reduction of element size. The overestimation of the error identified in Section 5.6.1.2 is also evident in this problem, where excessive refinement is applied in some regions along the boundary of the profile.

The simulations employing the multigrid-based adaptation strategy $\mathfrak{A}_2$ achieve SW convergence in all cases. When using this procedure, no instabilities like those observed with $\mathfrak{A}_1$ were detected; instead, fields like the one shown in Fig. 5.35 are obtained, exhibiting smooth behavior throughout the domain except in the vicinity of the geometry-induced singularity.



(a) *Global view* ***(b)*** *Detailed view around trailing edge*

Figure 5.35: *Electromagnetic energy density distribution at $t = 9.5T$ computed using the multigrid hp-adaptation scheme*

The final mesh obtained in the simulation with $\mu_{\Delta t} = f_{40}$ is shown in Figs. 5.36 and 5.37, with similar results observed for the other adaptation frequencies. As desired, $\mathfrak{A}_2$ performs h-refinement at both ends of the geometry, placing in these areas the smallest elements allowed in the discretization. In general, it can be seen that the strategy tends to use elements of smaller size and degree compared to $\mathfrak{A}_1$. This behavior is understood to result from the fact that larger elements would lead to excessively high error levels in the reference mesh or would violate the constraints imposed by Ω and the PML.

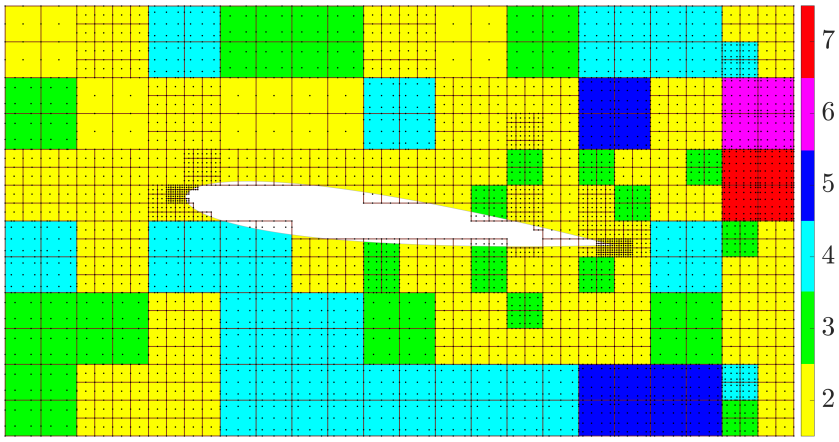
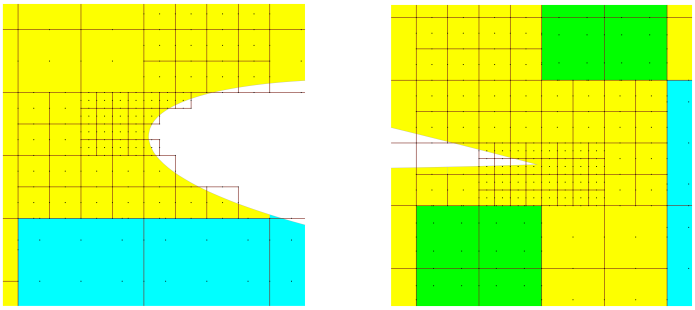


Figure 5.36: *Global view of the final mesh computed using multigrid hp-adaptation scheme*

The use of lower-degree elements may be contributing to the prevention of the instabilities previously discussed. We also believe that $\mathfrak{A}_2$ may be avoiding configurations prone to such behavior. By comparing different discretizations, it could be



(a) *Automatic hp-refinement arround leading edge*

(b) *Automatic hp-refinement arround trailing edge*

Figure 5.37: *Detailed views of the final mesh computed using multigrid hp-adaptation scheme*

detecting the onset of instability and modifying the mesh accordingly in those regions. This behavior would be highly interesting and deserves further investigation.

The results obtained for the SW are shown in Fig. 5.38. We observe that $\mathfrak{A}_2$ is able to significantly reduce the discrepancy between the curve obtained with the non-refined mesh f_0 and the reference solution. The improvement is consistent and does not appear to depend on the adaptation frequency. In fact, in all cases, the error of the f_0 analysis, initially 8%, is reduced to values below 2%. This improvement is limited by the positioning of the profile with respect to the mesh and the imposed discretization constraints, rather than by the effectiveness of the adaptation process.

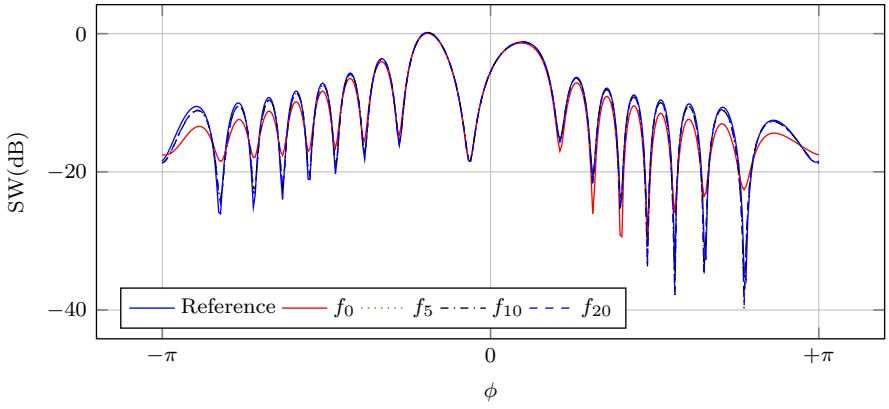


Figure 5.38: *Scattered width for the NACA0012 profile computed with multigrid-guided hp-adaptation*

Although it does not affect the accuracy of the results, the definition of the parameter $\mu_{\Delta t}$ does influence the duration of the simulations. Table 5.11 reports the time instant at which SW convergence is achieved in each simulation, with convergence

being checked every $0.5T$. It can be seen that higher adaptation frequencies delay the onset of the stationary phase, thus prolonging the total simulation time. The impact on the computational cost is twofold: on the one hand, the number of iterations of the time integration loop increases, and on the other, the adaptation process is triggered more frequently. It is also interesting to note that the simulations with adaptation terminate at a final time t_f close to that of the reference solution, which was $t_f = 8T$, even though, as previously observed, the adaptation process causes the solution to reach stationarity at a later time.

$\mu_{\Delta t}$	f_0	f_{10}	f_{20}	f_{40}
$t_f(T)$	6.5	8.5	8.5	9.5

Table 5.11: *Simulation time to meet tolerance on SW computation*

Regarding the distribution of computational time across the different meshes, the simulation on the calculation meshes accounts for approximately 47% of the total computation time, about 28% is spent on the reference meshes, and the remaining 25% on the coarser meshes. These percentages highlight the benefits of refining in p rather than in h; although the small difference observed in the computational time of the coarser meshes is favored by the tendency of $\mathfrak{A}_2$ to employ low interpolation degrees. In any case, the combined cost of simulating the two auxiliary meshes is comparable to that of solving the calculation mesh. We consider such a trade-off acceptable, since this additional cost allows for obtaining improved solutions, assessing the reliability of the final result, and avoids the need to rerun the same simulation multiple times while manually optimizing the discretization.

From this analysis, we conclude that the multigrid-based *hp*-adaptation strategy enables the optimization of the discretization in scattering problems involving singularities induced by geometric features of the domain. For this type of problem, the use of low refinement frequencies is preferable, as it reduces computation times while maintaining accuracy. The strategy based on the explicit error indicator has shown promising behavior, but it has also exhibited instability issues that require further investigation.

Chapter 6

Closure

6.1. Conclusions

In this thesis, we have introduced the cgDG method for solving problems governed by systems of hyperbolic partial differential equations, with a particular focus on the electromagnetic problem. The numerical analyses presented throughout this document demonstrate the capabilities of our methodology. Leveraging our approach, we have been able of consistently, accurately, and efficiently reproduce the evolution of wave-like fields in both stationary and transient regimes, for both 2D and 3D configurations. The key techniques embedded in cgDG that have enabled these outstanding results are:

- *High-order discretization using Cartesian elements.* This work represents the first time that high-order approximations have been applied in the context of cgFEM. The p-version of cgFEM has enabled the construction of highly efficient meshes at negligible cost. On one hand, the shape similarity between Cartesian mesh elements significantly accelerates computations on interior elements, while high-order interpolation allows for coarser meshes that maximize the computational gain per degree of freedom. On the other hand, the geometric independence of the mesh from the domain makes its construction trivial.

- *Discontinuous Galerkin formulation of the weak form.* A proper choice of numerical fluxes allows for a more faithful capture of the problem's physics, thus improving the solution. However, the main benefits of the DG approach stem from the decoupling of the global problem into element-wise subproblems. Within this framework, we have exploited the Cartesian mesh structure to drastically reduce computational time and memory requirements. At the same time, we have eliminated mesh conformity constraints, gaining maximum flexibility to adapt it to the specific features of each problem.
- *Integration of the exact geometric model into the FE analysis pipeline.* The first source of error in most numerical analyses stems from domain simplification to match discretization constraints. In contrast, cgDG preserves the exact geometric definition throughout the process, using CAD-based strategies for representation and evaluation. We have proposed preprocessing techniques that equip the models with certain features suited to its subsequent cgDG analysis, but always maintaining the original exact definition of geometry.
- *High-order integration with geometric fidelity.* Accurate integration over cut elements has traditionally been one of the key challenges in nonconforming methods. Thanks to the design of advanced numerical integration techniques, we have decoupled modeling and integration errors from discretization error. This thesis proposes procedures that decompose the portion of the physical domain within these elements, often very complex, into subdomains of simple shapes for which high-order quadratures are available. These procedures are aware of the exact domain definition and yield mappings from integration to physical space that either reproduce the geometry exactly or provide arbitrarily accurate approximations. Except in extreme cases, the mappings are bijective, ensuring positive weights and integration points located entirely within the physical domain.
- *Stabilization via extrapolation.* The stabilization strategies developed herein allow the stability of the time integrator to be decoupled from the size of element-domain overlapping at contour elements. We have proposed several stabilization techniques, each with its own characteristics, all based on extending the support of certain basis functions. These strategies are not only simple and effective but also confined to the modification of the approximation basis, making them compatible with any governing physics.
- *hp-adaptation of the mesh.* To fully exploit cgDG's capabilities, we have considered non-uniform meshes where element sizes and interpolation degrees are selected locally according to the solution's behavior. While these meshes were initially predefined, the final chapter of the thesis provides all necessary tools and a fully developed procedure for automatic *hp*-adaptation driven by error estimation. For both manual and automatic strategies, the DG formulation has allowed us to employ mesh configurations tailored solely to the solution, eliminating all inter-element constraints on size or degree.

6.2. Future works

This work has opened several lines of research for the cgDG technology. Now we will summarize the most important:

- **Study of alternative high-order element families:** In this thesis, only nodal high-order elements have been considered. However, it would be interesting to study the behavior of other interpolation bases. It is known that the stability of the numerical scheme is closely related to the approximation basis used. Therefore, it seems reasonable to think that by choosing an appropriate approximation basis at the element level, both the conditioning of the systems and the stability of the time integrators could be improved. On the other hand, the use of hierarchical elements would also offer advantages in analyses with adaptive discretization.

- **Extension of 3D integration strategies to more complex geometries:** A series of strategies for the precise evaluation of integrals in curved domains within FEAVox have been presented. However, the current implementation is limited to integrals over domains with a single geometry definition. To include the general case, where the curved boundary would be defined in segments, it has been proposed to use restricted Delaunay-style tetrahedralizations with Steiner points, which would use geometric and intersection information to obtain subdivisions of the integration domain that conform to the geometry. Procedures would need to be established to position the Steiner points and the new edges that would appear on the boundary of the geometry. However, once this decomposition is obtained, the integrals can be evaluated using the procedures described in this thesis.

- **Optimization of integration quadratures for nonlinear problems:** Although efficiency was always considered during the development of procedures for the design of quadratures, reliability, generality, and accuracy were prioritized. In this project, we have limited ourselves to linear problems where the operators could be precomputed, and the computational load was mainly associated with the time evolution of the solution. However, for nonlinear problems where operators are updated at each iteration of the time loop, the efficiency of the quadratures will be crucial. By applying moment fitting strategies, the obtained quadratures could be optimized following the described procedures and adapted to each particular case. The cost of the preprocessing phase would increase, but the total computation time should be significantly reduced.

- **Application of stabilization techniques in other contexts:** The poor conditioning of the systems due to the degrees of freedom associated with intersected elements is an inherent problem in FDM methods. Stabilization strategies like

those presented in this project can be adapted for application in other contexts, even with continuous formulations. Eliminating the degrees of freedom that cause poor conditioning of the problem should improve the properties of the system to be solved without significantly affecting the accuracy of the solution.

- **Implementation of mesh adaptation procedures in three-dimensional problems:** The numerical results obtained for 2D problems made it clear that, to get the most out of cgDG, the mesh needs to be adapted. So far, efforts in the 3D case have focused on achieving a scheme that is as robust and flexible as possible. The next stage of development should address the improvement of the method's efficiency, where mesh adaptation will play a crucial role. The error indicators and automatic adaptation strategies developed for the 2D case can easily be reformulated to account for the additional dimension. Similarly, the 3D implementation can inherit algorithms and structures that have already been designed and validated for the efficient handling of information in 2D problems.

- **Application of the proposed methodology to other problems:** The DG formulation has shown great potential for solving conservation laws. The behavior of cgDG must be studied when considering other physics, such as the Navier-Stokes equations, which contain nonlinear terms, diffusivity, etc. Since the strategies proposed to overcome the initial limitations of cgDG are independent of the problem formulation, the methodology is expected to adapt satisfactorily without requiring major modifications.

Appendices

Chapter A

Transfinite mappings: Bubble functions and projections

In this appendix, the necessary functions to evaluate the proposed mapping in section 4.2.2.3 are defined. The goal was to construct a bijective transformation ϕ from a space associated with each reference subdomain $\mathcal{R}$ with coordinates $\boldsymbol{\tau}$ to subdomains Ω_i in the physical space such that the boundaries Γ_i^ϕ of Ω_i^ϕ, which represent a portion Γ_i of the boundary Γ of the computational domain Ω, are coincident. The proposed formulation was:

$$\boldsymbol{\phi} = \boldsymbol{\phi}_p(\boldsymbol{\tau}) + \Delta\boldsymbol{\phi}_c(\boldsymbol{\tau}) \tag{A.1}$$

where $\boldsymbol{\phi}_p$ represents the polynomial transformation developed in section 4.2.2.2 and $\Delta\boldsymbol{\phi}_c$ is a correction that deforms the polynomial domain $\Omega_i^{\phi_p}$, ensuring that the boundary $\Gamma_{i,c}^\phi$ is equivalent to Γ_i. If we consider a single boundary in Γ for each subdomain i, we can modify the notation by using Γ_c^ϕ and Γ_c instead of $\Gamma_{i,c}^\phi$ and Γ_i (this was the notation used in section 4.2.2.2).

The correction is constructed by combining bubble functions $\boldsymbol{\psi}_c$ and interpolation functions σ_c as follows:

$$\Delta\boldsymbol{\phi}_c(\boldsymbol{\tau}) = \boldsymbol{\psi}_c(\boldsymbol{\tau})\,\sigma_c(\boldsymbol{\tau}) \tag{A.2}$$

σ_c is a linear interpolation that is easily obtained as the sum of the interpolation functions associated with the vertices in c.

The definition of the bubble functions $\boldsymbol{\psi}_c$ is much more complex. Ultimately, we want to evaluate the differences between points located on $\Gamma_c^{\phi_p}$ and their counterparts on Γ_c. This pairing can be done following the same strategy proposed in section 4.2.2.2, such that, if we denote points in a reference system associated with c as $\boldsymbol{\eta}$, the bubble functions could be defined as:

$$\boldsymbol{\psi}_c(\boldsymbol{\tau}) = \Gamma_c(\boldsymbol{\eta}) - \Gamma_c^{\phi_p}(\boldsymbol{\eta})$$

The problem now arises in the definition of the projection $\boldsymbol{\tau} \to \boldsymbol{\eta}$. Here we recall the properties that will be sought when defining the transformation:

- the transformations are continuous
- the Jacobians of the transformations are non-negative
- the transformations are identical to the entity for points on the boundary of interest
- the projection of a point on an edge of the subdomain connected to the target boundary will be located at the vertex connecting them
- the projection of a point on a face of the subdomain connected to the target boundary will be located on the shared edge, and can be obtained by knowing only the position of the point within the face that contains it.

In the following sections, all the necessary tools to formulate these transformations are presented. First, in appendix A.1, the topologies of all the reference subdomains are established, and the coordinate systems associated with each of them are introduced. This information is used in appendix A.1.1, where linear interpolation functions associated with each subdomain are defined. Finally, in appendix A.1.2, all the projections are defined, and explicit formulas for their calculation are provided based on the interpolation functions presented in appendix A.1.1.

A.1. Integration reference subdomains

The integration strategy we propose works with different types of subdomains. Fig. A.1 presents the reference element associated with each subdomain type, as well as the vertex numbering we have used and their coordinates in the reference space. Table A.1 and Table A.2 list the components that form each element, coded according to the vertices they connect. The reference system for a generic edge ab will have its origin at a, and its base will be the vector $a \to b$. For facets, the local coordinate

system will be defined by the first three vertices of the coding, such that both for the triangular facet *abc* and for the quadrangular facet *abcd*, it would be defined with its origin at a and the base of the space would be the vectors $\{a \to b, a \to c\}$. In this way, the coordinate systems on the components correspond to those of the reference domains of lower dimension than the starting domain.

Shape	Edge
Triangle	$\{12, 23, 13\}$
Quadrilateral	$\{12, 24, 34, 13\}$
Tetrahedron	$\{12, 23, 13, 14, 24, 34\}$
Prism	$\{12, 23, 13, 14, 25, 36, 45, 56, 46\}$
Pyramid	$\{12, 24, 34, 13, 15, 25, 35, 45\}$
Hexahedron	$\{12, 24, 34, 13, 15, 26, 37, 48, 56, 68, 78, 57\}$

Table A.1: *Edges codes per cell type following vertex notation shown in Fig. A.1.*

Shape	Triangular facets	Quadrangular facets
Tetrahedron	$\{123, 124, 134, 234\}$	
Prism	$\{123, 456\}$	$\{1245, 1346, 2356\}$
Pyramid	$\{125, 135, 245, 345\}$	$\{1234\}$
Hexahedron		$\{1234, 5678, 1256, 3478, 1357, 2468\}$

Table A.2: *Facet codes per cell type following vertex notation shown in Fig. A.1.*

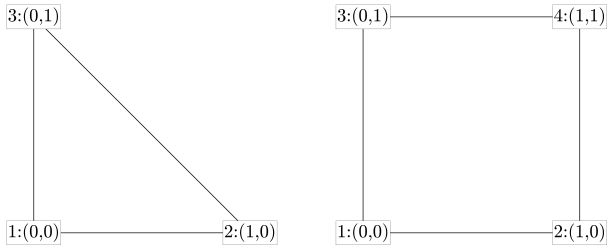
A.1.1. Linear interpolation functions for integration subdomains

Before proceeding to explain the projections, it is appropriate to define the linear interpolation functions associated with the vertices of the different reference elements. Taking as a starting point the shape functions for a one-dimensional element $I = \{\xi : \xi \in [0, 1]\}$ defined as:

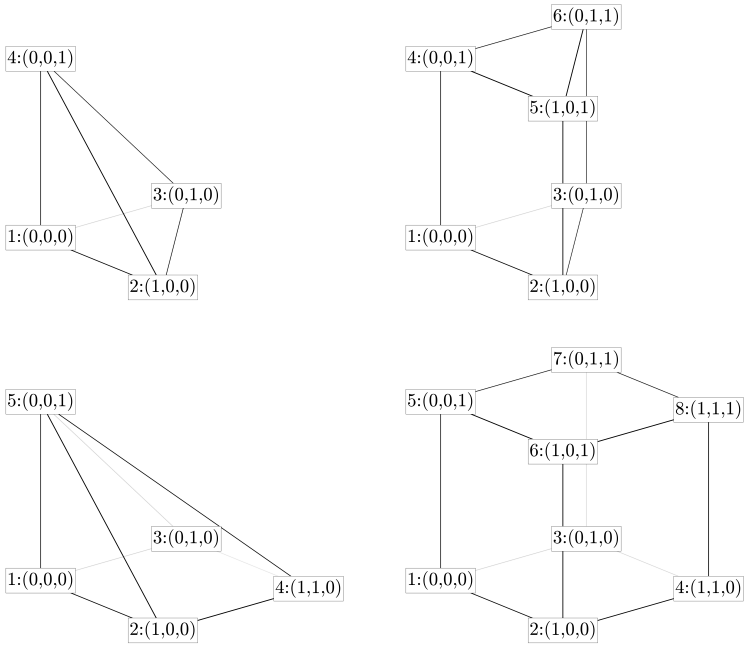
$$\chi_1^{\mathrm{I}}(\xi) = 1 - \xi \quad , \quad \chi_2^{\mathrm{I}}(\xi) = \xi$$

we can construct the interpolation functions for the reference quadrilateral element $\mathrm{Sq} = \{\boldsymbol{\xi} : \boldsymbol{\xi} \in [0, 1]^2\}$:

$$\chi_v^{\mathrm{Sq}}(\boldsymbol{\xi}) = \chi_{v_1,v_2}^{\mathrm{Sq}}(\xi_1, \xi_2) = \chi_{v_1}^{\mathrm{I}}(\xi_1)\, \chi_{v_2}^{\mathrm{I}}(\xi_2)$$



(a) *2D reference domains*



(b) *3D reference domains*

Figure A.1: *Vertex codification for all reference domains*

where the subscripts $v_1, v_2 \in \{1, 2\}$ relate v according to $v = v_1 + 2(v_2 - 1)$ and represent the index of the one-dimensional shape function in each spatial direction. Regarding the reference triangular element $\mathrm{Tr} = \{\boldsymbol{\xi} : \ \xi_1 \geq 0, \ \xi_2 \geq 0, \ \xi_1 + \xi_2 \leq 1\}$, the value of the shape functions for a given point is equivalent to the expression of the same point's position in barycentric coordinates:

$$\chi_1^{\mathrm{Tr}}(\boldsymbol{\xi}) = 1 - \xi_1 - \xi_2 \quad , \quad \chi_2^{\mathrm{Tr}}(\boldsymbol{\xi}) = \xi_1 \quad , \quad \chi_3^{\mathrm{Tr}}(\boldsymbol{\xi}) = \xi_2.$$

The shape functions constructed for the reference tetrahedron $\mathrm{Te} = \{\boldsymbol{\xi} : \ \xi_1 \geq 0, \ \xi_2 \geq 0, \ \xi_3 \geq 0, \ \xi_1 + \xi_2 + \xi_3 \leq 1\}$ are obtained by evaluating the barycentric coordinates of the point, in a way similar to the triangular case:

$$\chi_1^{\mathrm{Te}}(\boldsymbol{\xi}) = 1 - \xi_1 - \xi_2 - \xi_3 \quad , \quad \chi_2^{\mathrm{Te}}(\boldsymbol{\xi}) = \xi_1 \quad , \quad \chi_3^{\mathrm{Te}}(\boldsymbol{\xi}) = \xi_2 \quad , \quad \chi_4^{\mathrm{Te}}(\boldsymbol{\xi}) = \xi_3.$$

The interpolation functions for the reference prism $\mathrm{Pr} = \{\boldsymbol{\xi} : \ \xi_1 \geq 0, \ \xi_2 \geq 0, \ \xi_1 + \xi_2 \leq 1, \ \xi_3 \in [0,1]\}$ are obtained as the product of the shape functions for the triangular and one-dimensional elements:

$$\chi_v^{\mathrm{Pr}}(\boldsymbol{\xi}) = \chi_{v_1,v_2}^{\mathrm{Pr}}(\boldsymbol{\zeta}_3, \xi_3) = \chi_{v_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_3)\, \chi_{v_2}^{\mathrm{I}}(\xi_3)$$

where $v = v_1 + 3(v_2 - 1)$ holds, with $v_1 \in \{1,2,3\}$ and $v_2 \in \{1,2\}$ being the indices of the shape functions for the original elements, and the coordinates $\boldsymbol{\zeta}_i$ are defined as $\boldsymbol{\zeta}_i = \{\xi_j \in \boldsymbol{\xi} : i \neq j\}$.

The interpolation functions for the linear pyramid are constructed following the procedure using the Duffy transformation [171]. The strategy consists of obtaining the image of a tensorial polynomial basis after performing the transformation from a cubic domain of unit size to the reference pyramid $\mathrm{Py} = \{\boldsymbol{\xi} : \ \xi_1 \geq 0, \ \xi_2 \geq 0, \ \xi_3 \geq 0, \ \xi_1 + \xi_3 \leq 1, \ \xi_2 + \xi_3 \leq 1, \}$. By applying the constraints that ensure continuity between adjacent elements, the shape functions are derived:

$$\chi_v^{\mathrm{Py}}(\boldsymbol{\xi}) = \chi_{v_1,v_2,v_3}^{\mathrm{Py}}(\boldsymbol{\zeta}_2, \boldsymbol{\zeta}_1, \xi_3) = \frac{\chi_{v_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)\, \chi_{v_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}{\chi_{v_3}^{\mathrm{I}}(\xi_3)}$$

where $v_1, v_2 \in \{1,2,3\}$ and $v_3 \in \{1,2\}$, and it holds that:

$$\{v_1, v_2, v_3\} = \begin{cases} \{v - 2(\mathrm{ceil}\left(\frac{v}{2}\right) - 1), \mathrm{ceil}\left(\frac{v}{2}\right), 1\} & \text{if} \quad v \leq 4 \\ \{3,3,2\} & \text{if} \quad v = 5 \end{cases}$$

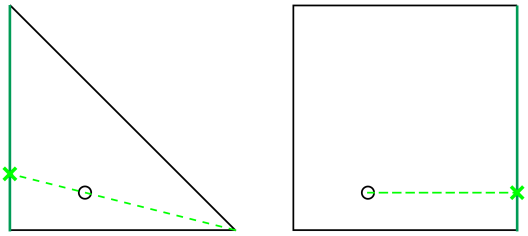
Finally, the shape functions for the hexahedral element $\mathrm{He} = \{\boldsymbol{\xi} : \boldsymbol{\xi} \in [0,1]^3\}$ are:

$$\chi_v^{\mathrm{He}}(\boldsymbol{\xi}) = \chi_{v_1,v_2,v_3}^{\mathrm{He}}(\xi_1, \xi_2, \xi_3) = \chi_{v_1}^{\mathrm{I}}(\xi_1)\, \chi_{v_2}^{\mathrm{I}}(\xi_2)\, \chi_{v_3}^{\mathrm{I}}(\xi_3)$$

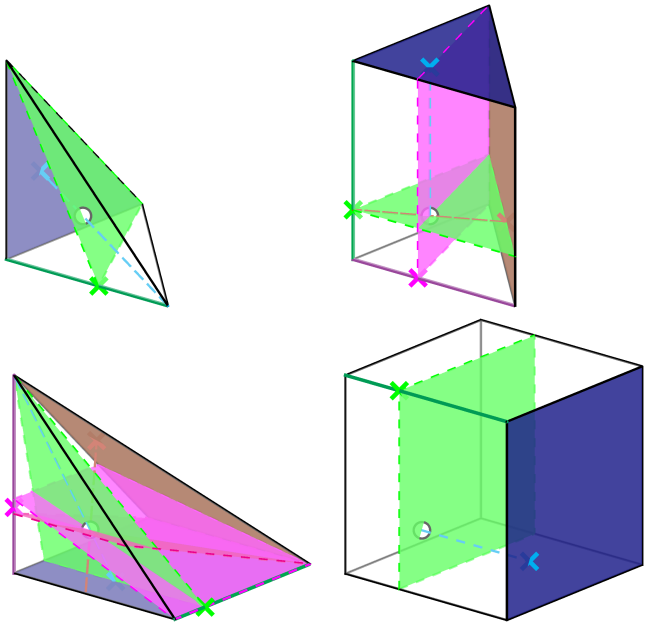
with $v = v_1 + 2(v_2 - 1) + 4(v_3 - 1)$ and $v_1, v_2, v_3 \in \{1,2\}$.

A.1.2. Point projection into local coordinate systems

The projections onto the components of each element will be calculated from the image of the point in the coordinate system associated with the reference element and the shape functions just presented. Fig. A.2 shows an example of each type of projection. Throughout the discussion, each projection will be related to the colors used to represent them in the figure.



(a) *2D reference domains*



(b) *3D reference domains*

Figure A.2: *Projections (crosses) of a point (white circle) for all reference domains. Similar colors relate the geometries involved in a certain projection: geometries used to project share the projection's color while target components of the reference domains are colored slightly different.*

Next, the strategy followed for the calculation of edge projections is identified. For each type of element and edge, the physical problem to be solved and the necessary formulation to evaluate the local coordinate η of the projection resulting from solving the problem are presented:

- Triangle: The intersection of the edge with the line passing through the opposite vertex of the triangle and the evaluation point is calculated. [Green in Fig. A.2].

$$\eta = \frac{\chi_b^{\mathrm{Tr}}(\boldsymbol{\xi})}{\sum_{l\in\{a,b\}} \chi_l^{\mathrm{Tr}}(\boldsymbol{\xi})}$$

- Quadrilateral: The evaluation point is projected onto the edge in the direction perpendicular to the edge. [Green in Fig. A.2].

$$\eta = \begin{cases} \xi_1 & if \quad b-a=1 \\ \xi_2 & if \quad b-a=2 \end{cases}$$

- Tetrahedron: The intersection of the edge with the plane containing the opposite edge and the evaluation point is calculated. [Green in Fig. A.2].

$$\eta = \frac{\chi_b^{\mathrm{Te}}(\boldsymbol{\xi})}{\sum_{l\in\{a,b\}} \chi_l^{\mathrm{Te}}(\boldsymbol{\xi})}$$

- Prism: For horizontal edges, their intersection with the plane containing the opposite vertical edge and the evaluation point is calculated. [Green in Fig. A.2]. For vertical edges, the perpendicular plane containing the evaluation point will be considered for the intersection. [Magenta in Fig. A.2]:

$$\eta = \begin{cases} \frac{\chi_{b_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_3)}{\sum_{l\in\{a,b\}} \chi_{l_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_3)} & if \quad b-a<3 \\ \xi_3 & if \quad b-a=3 \end{cases}$$

- Pyramid: For horizontal edges, the intersection with the plane containing the top vertex, the evaluation point, and the Cartesian direction perpendicular to the edge is evaluated. [Green in Fig. A.2]. For the remaining edges, their intersections with the planes containing the evaluation point, one of the vertices connected to the edge via a horizontal edge, and the horizontal Cartesian direction perpendicular to the same edge are calculated. The two resulting intersections are interpolated based on their proximity to the ends of the axis to obtain the final projection. [Magenta in Fig. A.2]

$$\eta = \begin{cases} \frac{\chi_{b_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)}{\sum_{l\in\{a,b\}} \chi_{l_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)} & if \quad a_2=b_2 \\ \frac{\chi_{b_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}{\sum_{l\in\{a,b\}} \chi_{l_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)} & if \quad a_1=b_1 \\ \frac{\chi_{b_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)}{\sum_{l\in\{a,b\}} \chi_{l_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)}\theta_1(\boldsymbol{\xi}) + \frac{\chi_{b_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}{\sum_{l\in\{a,b\}} \chi_{l_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}\theta_2(\boldsymbol{\xi}) & if \quad b=5 \end{cases}$$

where we define the interpolation functions as:

$$\theta_1(\boldsymbol{\xi}) = \frac{\chi_{a_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}{\chi_{a_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2) + \chi_{a_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}, \quad \theta_2(\boldsymbol{\xi}) = \frac{\chi_{a_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)}{\chi_{a_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2) + \chi_{a_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}$$

- Hexahedron: The intersection of the edge with the perpendicular Cartesian plane containing the evaluation point is calculated. [Green in Fig. A.2].

$$\eta = \begin{cases} \xi_1 & if \quad b-a=1 \\ \xi_2 & if \quad b-a=2 \\ \xi_3 & if \quad b-a=4 \end{cases}$$

Proceeding in the same way when dealing with facets; each point $\boldsymbol{\xi}$ in the reference element will have an associated image $\boldsymbol{\eta}$ in the local coordinate system of the facet $\mathcal{F}$. The physical problem considered in each case and the resulting mappings are:

- Tetrahedron: The intersection of the facet with the line containing the opposite vertex and the evaluation point is calculated. [Blue in Fig. A.2]

$$\boldsymbol{\eta} = \left(\frac{\chi_b^{\mathrm{Te}}(\boldsymbol{\xi})}{\sum_{l\in\{a,b,c\}} \chi_l^{\mathrm{Te}}(\boldsymbol{\xi})}, \frac{\chi_c^{\mathrm{Te}}(\boldsymbol{\xi})}{\sum_{l\in\{a,b,c\}} \chi_l^{\mathrm{Te}}(\boldsymbol{\xi})} \right)$$

- Prism: For triangular facets, the projection of the evaluation point in the vertical direction is calculated. [Blue in Fig. A.2]. For quadrilateral facets, the intersection with the line contained in a horizontal plane, containing the evaluation point and intersecting the opposite edge, is evaluated. [Blue in Fig. A.2].

$$\boldsymbol{\eta} = \begin{cases} (\xi_1, \xi_2) & if \quad \mathcal{F} \in \{123, 456\} \\ \left(\frac{\chi_{b_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_3)}{\sum_{l\in\{a,b\}} \chi_{l_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_3)}, \xi_3 \right) & if \quad \mathcal{F} \in \{1245, 1346, 2356\} \end{cases}$$

- Pyramid: For the square facet, its intersection with the line passing through the opposite vertex and the evaluation point is calculated. For triangular facets, the intersection of the pyramid with the plane containing the opposite edge and the evaluation point is obtained. The projection is calculated so that its relative position in the intersection segment of the plane with the facet is the same as the relative position of the intersection point concerning the horizontal segment contained in the cutting plane.

$$\boldsymbol{\eta} = \begin{cases} \boldsymbol{\eta} = \left(\frac{\xi_1}{\chi_1^{\mathrm{I}}(\xi_3)}, \frac{\xi_2}{\chi_1^{\mathrm{I}}(\xi_3)} \right) & if \quad \mathcal{F} = 1234 \\ \left(\frac{\xi_1}{\chi_1^{\mathrm{I}}(\xi_3)} \frac{\chi_{a_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}{\sum_{l\in\{a,c\}} \chi_{l_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}, \frac{\chi_{c_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)}{\sum_{l\in\{a,c\}} \chi_{l_2}^{\mathrm{Tr}}(\boldsymbol{\zeta}_1)} \right) & if \quad \mathcal{F} \in \{125, 345\} \\ \left(\frac{\xi_2}{\chi_1^{\mathrm{I}}(\xi_3)} \frac{\chi_{a_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)}{\sum_{l\in\{a,c\}} \chi_{l_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)}, \frac{\chi_{c_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)}{\sum_{l\in\{a,c\}} \chi_{l_1}^{\mathrm{Tr}}(\boldsymbol{\zeta}_2)} \right) & if \quad \mathcal{F} \in \{135, 245\} \end{cases}$$

- Hexahedron: The intersection of the facet with the line perpendicular to the face containing the evaluation point is calculated.

$$\boldsymbol{\eta} = \begin{cases} (\xi_1, \xi_2) & if \quad \mathcal{F} \in \{1234, 5678\} \\ (\xi_1, \xi_3) & if \quad \mathcal{F} \in \{1256, 3478\} \\ (\xi_2, \xi_3) & if \quad \mathcal{F} \in \{1357, 2468\} \end{cases}$$

Chapter B

NURBSLab

Along the main body of this document, we have emphasised the very important role that the accurate representation of the domain plays in the context of the cgDG method. To maximize the robustness of the method and leverage the potential of high order approximations we had proposed a set of strategies many of them using the exact definition of the geometry, the NURBS surfaces and curves typicaly used in CAD softwares to model the boundary of the solids.

When the development of the cgDG began, we rely on the computational tools previously developed within the Institute of Mechanical and Biomechanical Engineering at the Polytechnic University of Valencia for working with NURBS. However, the strategies that were being developed in the context of this thesis soon required calculations that exceeded the capabilities of those tools. We looked for alternatives that made the work. The *Curve Fitting Toolbox* from MATLAB® provides a few functions to work with piecewise polynomial curves in the context of data interpolation with univariate functions. Some packages published by the community targeted this topic in a more geometrical fashion. The *NURBS Toolbox* library implemented by M. Spink [143] stands as the most widespread one. It appears in the development of isogeometric analysis methods [172] and mainly focuses on the generation of NURBS leaving the computation of derived entities to a secondary role while no functionalities to evaluate the interaction between entities are provided. The MATLAB® library has been unsupported for many years while its Octave counterpart [173] continues to update but only minor changes are included (current last version from 2021). Another more recent alternative is the package *NURBS* available in [174]. This library

provides some of the basic functionalities for evaluating and modifying NURBS being its main feature that it was implemented in an object-oriented fashion. As far as the authors are aware, there are no comprehensive, efficient, and general-use libraries available online that provide a good range of tools for working with NURBS within an application implemented in MATLAB® .

The *NURBSLab* toolbox was created to fill this gap. It is a library built form scratch with a wide variety of functions that allow handling NURBS curves and surfaces without leaving the MATLAB® environment. Among the capabilities implemented, we can find sampling, decomposition, reparameterization, visualization, projection evaluation, and intersection calculation. To facilitate its use within any application, the construction of specific data structures within the library has been avoided, and it works directly with the control points and knot vectors that define the NURBS, which are requested as independent variables. In this way, there is no need to transform the original application's data structures or duplicate information. The user of the library will be able to utilize its functionalities by retrieving the specific inputs from their data structure and applying minimal transformations.

The following section lists the functions that constitute the core of the library. This documentation includes the purpose of each function as well as its input and output variables, ordered according to their position in the function call. For the input and output arguments, in addition to a brief explanation of their meaning, the expected data types and dimensions are provided.

B.1. Non-Uniform Rational B-Splines. Fundamentals

B.2. Core functions

nurbs_BasicConstants

Computes the basic constants of a NURBS curve (d=1) or surface (d=2) as defined in [141]

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}.

Output arguments:

p: degree of the NURBS in each direction. {1xd matrix}

n: n(i)+1 is the number of control points in the ith direction. {1xd matrix}

m: m(i)+1 is the size of the knot vector in the ith direction. {1xd matrix}

nurbs_BasisFunction

Compute the B-Spline basis functions at a given set of points

Input arguments:

params: coordinates of the N points in parametric space of dimension d. {dxN matrix}

knots: knot vector. {1x2 cell if d=1; 1x(m+1) matrix if d=2}.

order: order of the B-Spline. {1xd matrix}

span(optional): for each point in params, the span considered to evaluate the basis functions. {dxN matrix}

Output arguments:

Basis: basis functions of the B-Spline evaluates at the given point coordinates {prod(*order*)xN matrix}

subPw: indices of the control points asociated to the basis functions. The number of basis functions asociated to a point if less than the NURBS order, the NaNs are returned. {prod(*order*)xN matrix}

nurbs_CheckCurveSuperposition

Discover if two NURBS curves overlap in non-homogeneous space to some extent.

Input arguments:

uPw: control points of the first curve in homogeneous coordinates. {4x(n+1) matrix}.

uKnots: knot vector of the first curve. {1x(m+1) matrix}.

vPw: control points of the second curve in homogeneous coordinates. {4x(n+1) matrix}.

vKnots: knot vector of the second curve. {1x(m+1) matrix}.

TOL (default=$1e-6$*)*: distance under wich two segment of the curves, at least this same distance long (if *SAMPLESIZE* is ignored), are considered superposed. {scalar}

SAMPLESIZE (optional): number of points that are sampled in each curve span to evaluate superposition. {integer}

Output arguments:

flag: if the curves are disjoint or they intersect at isolated points it is equal to 0; if the curves completely overlap it is equal to 1; if the curves partialy overlap it is equal to 2. {integer}

uSpan: ranges in parametric space of the first curve that overlap. The spans are sorted in ascending order regarding parameter values and span limits are also sorted in ascending. {2xN matrix}

vSpan: ranges in parametric space of the second curve that overlap. The spans and their end points are arranged such that their images in non-homogeneous space match with the first curve. {2xN matrix}

nurbs_ComputeDefinitionInterval

Compute the ranges in parametric space for which the NURBS surface (d=2) or curve (d=1) is defined.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}.

Output arguments:

iniParam: lower limit of the definition interval {1xd matrix}

endParam: upper limit of the definition interval {1xd matrix}

nurbs_ComputeFundamentalForms

Compute the quadratic differential forms of a NURBS surface at a set of points. The first fundamental form characterizes the interior geometry of the surface and is computed as: $d\mathbf{r}^2 = Edu^2+2Fdudv+Gdv^2$; while the second fundamental form characterizes its local structure and is computed as: $2\delta = Ldu^2+2Mdudv+Ndv^2$

Input arguments:

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector of the surface. {1x2 cell}.

uv: coordinates of N points in the parametric space of the surface. {2xN matrix}

Output arguments:

E: du^2 coefficients of the first fundamental form. {1xN matrix}

F: *dudv* coefficients of the first fundamental form. {1xN matrix}

G: dv^2 coefficients of the first fundamental form. {1xN matrix}

L: du^2 coefficients of the second fundamental form. {1xN matrix}

M: *dudv* coefficients of the second fundamental form. {1xN matrix}

N: dv^2 coefficients of the second fundamental form. {1xN matrix}

nurbs_ComputeGaussianCurvature

Compute the Gaussian curvature of a NURBS surface at a set of points.

Input arguments:

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector of the surface. {1x2 cell}.

uv: coordinates of N points in the parametric space of the surface. {2xN matrix}

Output arguments:

K: Gaussian curvature at each point. {1xN matrix}

nurbs_ComputeLengthOnCurve

Compute the length of the NURBS curve segment that ranges between two points in its parametric space.

Input arguments:

Pw: control points of the curve in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the curve. {1x(m+1) matrix}

iniU(default=lower bound of the definition interval): parametric coordinates of the segment starting point. {scalar}

endU(default=lower bound of the definition interval): parametric coordinates of the segment ending point. {scalar}

z01(optional): coordinates of the G sampling points of an integration quadrature in [0,1] reference space. {1xG scalar}

w01(optional): weight of the integration quadrature with sampling points *z01*. {1xG scalar}

Output arguments:

len: length of the curve segment. {scalar}

nurbs_ComputeLoopOnSurfaceBoundary

Compute the NURBS definition of the set of curves comprising the boundary of a NURBS surface.

Input arguments:

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the surface. {1x2 cell}

Output arguments:

cPw: control point of the boundary curves both in homogeneous coordinates and in the parametric coordinates of the surface {2x4 cell}

cKnots: knot vector of the curves. {2x4 cell}

OF: set of flag that dictates if the curve should be traversed in reverse direction to close a continuous loop that delimits the surface leaving the support of the surface on the left when curves in parametric space are considered. {1x4 boolean matrix}

nurbs_ComputeMeanCurvature

Compute the mean curvature of a NURBS surface at a set of points.

Input arguments:

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector of the surface. {1x2 cell}.

uv: coordinates of N points in the parametric space of the surface. {2xN matrix}

Output arguments:

K: mean curvature at each point. {1xN matrix}

nurbs_ComputePrincipalCurvatures

Compute the principal curvatures of a NURBS surface at a set of points.

Input arguments:

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector of the surface. {1x2 cell}.

uv: coordinates of N points in the parametric space of the surface. {2xN matrix}

Output arguments:

k1: first principal curvature at each point. {1xN matrix}

k2: second principal curvature at each point. {1xN matrix}

nurbs_ConstantParameterCurveInSurface

Compute the NURBS definition of the curve that arises when a NURBS surface is evaluated setting any of its coordinates in parametric space to a certain value.

Input arguments:

params: parametric coordinates of the surface defined as follows value si provided for the fixed coordinate while the free coordinate is set to NaN. {2x1 matrix}

sPw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

sKnots: knot vector of the surface. {1x2 cell}

Output arguments:

cPw: control points of the curve in homogeneous coordinates {4x(n+1) matrix}

cKnots: knot vector of the curve. {1x(m+1) matrix}

nurbs_CurveDimension

Determine the dimension of the manifold in non-homogeneous space the curve lays in.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}

TOL: maximum distance in non-homogeneous coordinates that the points in the curve may distance from the manifold while being considered laying in it. {scalar}

Output arguments:

dim: minimum dimension of the manifold that contains the whole curve. {scalar}

nurbs_Derivative

Compute the NURBS definition of the first order derivatives of a NURBS surface (d=2) or curve (d=1) in homogeneous space.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}

Output arguments:

dPw: control points of the derivatives in homogeneous coordinates. {1x2 cell if d=2; 1xn matrix if d=1}

dKnots: knot vectos of the derivatives. {1x2 cell if d=2; 1x(m-1) matrix if d=1}

nurbs_EvaluateCurveSurfaceComposition

Given a NURBS curve whose image is in the parametric space of a NURBS surface, evaluate the images of the composition for a set o points.

Input arguments:

params: coordinates of N points in the parametric space of the curve. {dxN matrix}

cPw: control points of the curve in homogeneous coordinates. {4x(n+1) matrix}

cKnots: knot vector of the curve. {1x(m+1) matrix}

sPw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

sKnots: knot vector of the surface. {1x2 cell}

deval: flag that makes this function returns the images of the first derivate of the composition when true. {boolean}

Output arguments:

xyz: Images of the points in non-homogeneous coordinates. {3xN matrix}

nurbs_EvaluateDifferentiabilityAtKnots

Compute the number of continuous derivatives of the NURBS curve (d=1) or surface (d=2) at every knot value.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}.

Output arguments:

diff: number of continuous derivatives for every knot value. The size of the output equals the number of different knots values U in each parametric direction. {1x2 cell if d=2; 1x(m+1) matrix if d=1}.

nurbs_Evaluation

Evaluate NURBS curve (d=1) or surface (d=2) at a given set of points.

Input arguments:

params: coordinates of N points in parametric space. {dxN matrix}

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}

Output arguments:

coords: if number of output arguments is 1, images of the points in homogeneous coordinates {4xN matrix}; if number of output arguments is 2, images of the points in non-homogeneous coordinates {3xN matrix}

w(optional): fourth component of the points in homogeneos coordinates {1xN matrix}

nurbs_EvalutateNDerivativeOfCurvePolyComposition

Evaluate the α-derivative of the reparameterized NURBS curve at a given set of points.

Input arguments:

alpha:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector. {1x(m+1) matrix}.

params: coordinates of N points in parametric space. {1xN matrix}.

Output arguments:

coords: if number of output arguments is 1, images of the points in homogeneous coordinates {4xN matrix}; if number of output arguments is 2, images of the points in non-homogeneous coordinates {3xN matrix}

w(optional): fourth component of the points in homogeneos coordinates {1xN matrix}

nurbs_FindAppropriateQuadratureToComputeLength

Provide the lowest order Gaussian integration quadrature to accurately compute the length of a NURBS curve.

Input arguments:

Pw: control points of the curve in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the curve. {1x(m+1) matrix}

Output arguments:

z01: coordinates in [0,1] reference space of the G sampling points that build the integration quadrature. {1xG scalar}

w01(optional): weight of the integration quadrature with sampling points *z01*. {1xG scalar}

nurbs_FindLocalExtrema

Find all the points in a NURBS surface (d=2) or curve (d=1) that are local maxima or minima in any cartesian direction.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}

guess(optional): parametric coordinates of the N points used as initial guess of the Newton-Raphson solver. {dxN matrix}

TOL: convergence of the Newton-Raphson solver is achieved when the projection of the vectors that are tangent to the NURBS on the target Cartesian direction is less than this value. {scalar}

MAXITER(default=20): maximum number of iterations of the Newton-Raphson solver . {scalar}

Output arguments:

XYZ: coordinates of the local maxima and minima in the non-homogeneous space. {3xM matrix}

UV: coordinates of the local maxima and minima in the parametric space of the NURBS. {dxM matrix}

IntPlanes: flags identifying the Cartesian directions for which the point is not a local maximum or minimum. {3xM boolean matrix}

nurbs_FindPointsOnSurface

Find the parametric coordinates of a the set of points in a NURBS surface.

Input arguments:

XYZ: non-homogeneous coordinates of N points in the surface. {3xN matrix}

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the surface. {1x2 cell}

UV0(optional): parametric coordinates of the points used as initial guess for the Newton-Raphson solver. {2xN matrix}

NRTOL(default=1e-10): convergence of the Newton-Raphson solver is achieved when the distance in non-homogeneous space between the target point and the current guess is less than this value. {scalar}

MAXITER(default=20): maximum number of iterations of the Newton-Raphson solver . {scalar}

GEOTOL(default=1.1e-10): when the distance between the output of the Newton-Raphson solver and the surface boundary is less than this value the approximate solution is moved to the boundary of the surface. {scalar}

Output arguments:

pXYZ: coordinates of the approximate solutions in the non-homogeneous space. {3xN matrix}

UV: coordinates of the approximate solutions in the parametric space of the surface. If an approximate solution coincides with a collapsed point of the surface, the undetermined coordinate is set to NaN. {2xN matrix}

NonConv: flags storing the non-convergence of the Newton-Raphson solver for all points. {1xN matrix}

nurbs_FindProjectedPointsOnCurves

Given two NURBS curve, find pairs of points such that there is no point in their neighbourhoods that is closer to its pair in the other curve.

Input arguments:

Pw1: control points of the first curve in homogeneous coordinates. {4x(n+1) matrix}

knots1: knot vector of the first curve. {1x(m+1) matrix}

Pw2: control points of the second curve in homogeneous coordinates. {4x(n'+1) matrix}

knots2: knot vector of the second curve. {1x(m'+1) matrix}

t10(optional): parametric coordinate of the points used as initial guess of the Newton-Raphson solver for the first curve. {1xN matrix}

t20(optional): parametric coordinate of the points used as initial guess of the Newton-Raphson solver for the second curve. {1xN matrix}

NRTOL(default=1e-10): convergence of the Newton-Raphson solver is achieved when the distance between the projected point generated in two consecutive iterations is less than this value. {scalar}

MAXITER(default=20): maximum number of iterations of the Newton-Raphson solver . {scalar}

Output arguments:

XYZ1: coordinates in the non-homogeneous space of the closest in the first curve. {3xM matrix}

XYZ2: coordinates in the non-homogeneous space of the closest in the second curve. {3xM matrix}

T1: coordinate of the closest points in the parametric space of the first curve. {1xM matrix}

T2: coordinate of the closest points in the parametric space of the second curve. {1xM matrix}

nurbs_FindPtsInSurfaceNormalToVector

Find the points in a NURBS surface at which the surface is perpendicular to a certain direction.

Input arguments:

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the surface. {1x2 cell}

UV0(optional): parametric coordinates of the points used as initial guess for the Newton-Raphson solver. {2x1 matrix}

vector: target direction. {3x1 matrix}

NRTOL(default=1e-10): convergence of the Newton-Raphson solver is achieved when the projection of the vector direction on the plane tangent to the surface at the current guess is less than this value. {scalar}

MAXITER(default=20): maximum number of iterations of the Newton-Raphson solver . {scalar}

Output arguments:

XYZ: coordinates of the approximate solutions in the non-homogeneous space. {3xN matrix}

UV: coordinates of the approximate solutions in the parametric space of the surface. If an approximate solution coincides with a collapsed point of the surface, the undetermined coordinate is set to NaN. {2xN matrix}

nurbs_FindSpan

Identify the knot spans asociated to each point from a given set.

Input arguments:

params: coordinates of the N points in parametric space of dimension d. {dxN matrix}

knots: knot vector. {1x2 cell if d=1; 1x(m+1) matrix if d=2}

order: order of the B-Spline. {1xd matrix}

TOLERANCE(default=$1e-5$*)*: distance in parametric space such that every point closer to a knot this value is considered to share knots coordinates. {scalar}

Output arguments:

span: knot spans in which the points lay. {dxN matrix}

nurbs_IntersectSurfaceWithCartesianAxes

Find all the intersections of a NURBS surface with a set of cartesian axes.

Input arguments:

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the surface. {1x2 cell}

UV0(optional): parametric coordinates of the N points used as initial guess for the Newton-Raphson solver. {2xN matrix}

dir: the non-zero direction of the Cartesian axes. {1xN matrix}

pt0: non-homogeneous coordinates of N points the Cartesian axes pass through. {3xN matrix}

NRTOL(default=1e-10): convergence of the Newton-Raphson solver is achieved when the distance between the current guess and its projection on the cartesian axis than this value. {scalar}

MAXITER(default=20): maximum number of iterations of the Newton-Raphson solver . {scalar}

GEOTOL(default=1.1e-10): when the distance between the output of the Newton-Raphson solver and the surface boundary is less than this value the approximate solution is moved to the boundary of the surface. {scalar}

Output arguments:

XYZ: coordinates of the intersections in the non-homogeneous space. {3xM matrix}

UV: coordinates of the intersections in the parametric space of the surface. If an intersection coincides with a collapsed point of the surface, the undetermined coordinate is set to NaN. {2xM matrix}

AxisID: pointer to the Cartesian axis provided as input that intersects the surface at each point. {1xM matrix}

nurbs_IsPeriodic

Discover if a NURBS curve (d=1) or surface (d=2) is periodic in any direction.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}.

TOL: distance in non-homogeneous space . {scalar}.

Output arguments:

flag: result of assessing periodicity in each direction of the parametric space. {1xd boolean matrix}

nurbs_IsPointOnSurfaceBoundary

Discover if a set of points lays on the boundary of a NURBS surface.

Input arguments:

pts: coordinates of N point in non-homogeneous space (d=3) or surface parametric space (d=2). {dxN matrix}

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the surface. {1x2 cell}

TOL: maximum distance from the surface boundary in non-homogeneous space that is considered to characterised point relative locations. {scalar}

Output arguments:

flag: results of individually assesing the position of the points with respect to the surface boundary. {1xN boolean matrix}

nurbs_NDerivateAtPoints

Evaluate derivatives of NURBS curve (d=1) or surface (d=2) at a given set of points.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}.

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}.

deriv: order of the derivative in each direction {1xd matrix}

params: coordinates of N points in parametric space. {dxN matrix}.

Output arguments:

Xdp: value of the *deriv*-derivative of the NURBS at point coordinates. {3xN matrix}

nurbs_Plot

Display the non-homogeneous image and control points of a NURBS curve (d=1) or surface (d=2).

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}

SampleSize: number of point to sample in each parametric dimension. {1xd matrix}

NewFigure: if it is True, the NURBS will be plotted in a new figure. {boolean}

nurbs_ProjectPointsOnCurve

Find the set of points (projections) in a NURBS curve that are the closest with respect to a set of point in the non-homogeneous space.

Input arguments:

XYZ: set of N point to project. {3xN matrix}

Pw: control points of the curve in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the curve. {1x(m+1) matrix}

u0(optional): parametric coordinate of the points used as initial guess fo the Newton-Raphson solver. {1xN matrix}

NRTOL(default=1e-10): convergence of the Newton-Raphson solver is achieved when the distance between the projected point generated in two consecutive iterations is less than this value. {scalar}

MAXITER(default=20): maximum number of iterations of the Newton-Raphson solver . {scalar}

GEOTOL(default=1.1e-10): when the distance between a projected point and an end of the curve is less than this value the projected point is moved to the end of the curve. {scalar}

Output arguments:

pXYZ: coordinates of the projected points in the non-homogeneous space. {3xN matrix}

U: coordinate of the projected points in the parametric space of the curve. {1xN matrix}

NonConv: flags storing the non-convergence of the Newton-Raphson solver for all points. {1xN matrix}

nurbs_ProjectPointsOnSurface

Find the set of points (projections) in a NURBS surface that are the closest with respect to a set of point in the non-homogeneous space.

Input arguments:

XYZ: set of N point to project. {3xN matrix}

Pw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the surface. {1x2 cell}

UV0(optional): parametric coordinates of the points used as initial guess for the Newton-Raphson solver. {2xN matrix}

NRTOL(default=1e-10): convergence of the Newton-Raphson solver is achieved when the distance between the projected point generated in two consecutive iterations is less than this value. {scalar}

MAXITER(default=20): maximum number of iterations of the Newton-Raphson solver . {scalar}

GEOTOL(default=1.1e-10): when the distance between a projected point and the surface boundary is less than this value the projected point is moved to the boundary of the surface. {scalar}

Output arguments:

pXYZ: coordinates of the projected points in the non-homogeneous space. {3xN matrix}

UV: coordinates of the projected points in the parametric space of the surface. If a projection coincides with a collapsed point of the surface, the undetermined coordinate is set to NaN. {2xN matrix}

NonConv: flags storing the non-convergence of the Newton-Raphson solver for all points. {1xN matrix}

nurbs_RefineCurveKnotVector

Knot refinement of a NURBS curve using Oslo algorithm, see [141].

Input arguments:

Pw: control points of the original curve in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the original curve. {1x(m+1) matrix}

U: set of new knots for insertion. {1xN matrix}

nTimes: number of times each knot is inserted. {1xN matrix}

Output arguments:

Pw: Control points of the new curve in homogeneous coordinates. {4x(n'+1) matrix}

knots: knot vector of the new curve. {1x(m'+1) matrix}

nurbs_RemoveKnotsInCurve

Compute the NURBS curve that results from taking some knots away from a NURBS curve while trying to keep unchanged the non-homogeneous image of the curve.

Input arguments:

Pw: control points of the original curve in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the original curve. {1x(m+1) matrix}

U: values of the knots to remove. {1xN matrix}

nTimes: number of times to remove. {1xN matrix}

Output arguments:

Pw: control points of the new curve in homogeneous coordinates. {4x(n'+1) matrix}

knots: knot vector of the new curve. {1x(m'+1) matrix}

flag: true value if all knots have been successfully removed with minor discrepancies between the images of the original and new curves. {boolean}

nurbs_ReparameterizeCurveUsingPolynomialMapping

Compute the NURBS definition of a curve that has been reparameterised using a polynomial mapping.

Input arguments:

Pw: control points of the original curve in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the original curve. {1x(m+1) matrix}

polyCoefs: coefficients of the polynomial function of degree q that maps the new parameterisation into the old one. The coefficients are sorted such that the last value in the list is the independent term of the transformation. {1x(q+1) matrix}

s0: coordinates of the start point in the new parameterization. {scalar}

sEnd: coordinates of the start point in the new parameterization. {scalar}

Output arguments:

Pw: control points of the reparameterised curve in homogeneous coordinates. {4x(n'+1) matrix}

knots: knot vector of the reparameterised curve. {1x(m'+1) matrix}

nurbs_SplitEntityAtPt

Split NURBS curve (d=1) at a certain point or surface (d=2) following constant parameter lines that intersect at a certain point.

Input arguments:

Pw: control points in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector. {1x2 cell if d=2; 1x(m+1) matrix if d=1}

u: first coordinate in parametric space of the point where to split. {scalar}

v(only if d=2): second coordinate in parametric space of the point where to split. {scalar}

Output arguments:

Pw: control points of the components in homogeneous coordinates. {1x(2^{d}) cell}

knots: knot vector of the components. {1x(2^{d}) cell}

nurbs_StraightLineInSurface

Compute the NURBS definition of the curve that arises when a NURBS surface is traversed folloing the straight path in the parametric space that connect two points.

Input arguments:

sPw: control points of the surface in homogeneous coordinates. {4x(n+1) matrix}

sKnots: knot vector of the surface. {1x2 cell}

uv1: coordinates of the start point in the parametric space of the surface. {2x1 matrix}

uv2: coordinates of the end point in the parametric space of the surface. {2x1 matrix}

Output arguments:

cPw: control points of the curve in homogeneous coordinates {4x(n'+1) matrix}

cKnots: knot vector of the curve. {1x(m'+1) matrix}

nurbs_TurnUnclampedIntoClamped

Modify NURBS curve (d=1) or surface (d=2) definition to express it as a clamped NURBS. The input and output NURBS are de same but the first and the last knots in the knot vector of the output NURBS have multiplicity equal to the order of the NURBS.

Input arguments:

Pw: control points of the unclamped NURBS in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the unclamped NURBS. {1x2 cell if d=2; 1x(m+1) matrix if d=1}

Output arguments:

Pw: control points of the clamped NURBS in homogeneous coordinates. {4x(n+1) matrix}

knots: knot vector of the clamped NURBS. {1x2 cell if d=2; 1x(m+1) matrix if d=1}

Chapter C

New trace inverse estimate for polynomials

In Section 5.3.3, a modification to the analysis scheme proposed in [167, 168] was introduced, enabling the derivation of error bounds for polynomial approximations more suitable for evaluation in the cgDG context. Specifically, it was proposed to replace the field $\mathbf{m}_i^\circ(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_i^0)$ considered in the original work with a new field $\mathbf{m}_i = \mathbf{m}_i^\circ \odot \mathfrak{n}$, which exhibits advantageous properties for defining $\mathbf{x}_i^0$. This appendix extends the analysis using this new field and briefly presents its application in deriving a new trace-inverse estimate for polynomial functions.

The geometric setting for this analysis is similar to that used in Section 5.3.3 and is illustrated in Fig. C.1. A partition of the domain Ω_i delimited by $\mathbf{x}_i^0$ and $\partial\Omega_i$ into r subdomains $\{\Omega_i^j\}$ is considered. Each subdomain in the partition is associated with two domains: $\tilde{\Omega}_i^j$ and $\underline{\Omega}_i^j$. The first, $\tilde{\Omega}_i^j$, identifies the triangular domain enclosed by connecting the vertices of Ω_i^j using straight segments, while $\underline{\Omega}_i^j$ denotes the largest domain similar to $\tilde{\Omega}_i^j$ entirely contained within Ω_i^j. The boundary of $\tilde{\Omega}_i^j$ opposite to $\mathbf{x}_i^0$ is denoted as $\partial\tilde{\Omega}_i^j$, with $\tilde{\mathbf{n}}_i^j$ representing its outward normal and $\tilde{\mathfrak{n}}_i^j$ the vector obtained by taking the absolute value of each component of $\tilde{\mathbf{n}}_i^j$. Corresponding entities are similarly defined with respect to $\underline{\Omega}_i^j$.

Following the same reasoning as in [168], the following inequality is obtained:

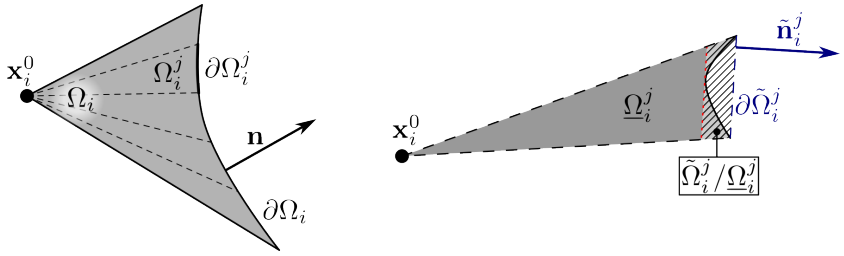


Figure C.1: *Geometric setting in the inference of trace estimates*

$$\|v\|^2_{\partial\Omega_i} \leq \sum_{j=1}^{r} \frac{(1+\delta_j)(p+1)(p+2)}{2} \frac{|\partial\underline{\Omega}_i^j|}{|\underline{\Omega}_i^j|} \|u\|_{\Omega_i^j} \tag{C.1}$$

Applying the divergence theorem to $\nabla \cdot \tilde{\mathbf{m}}_i^j$ and considering that $\underline{\Omega}_i^j$ is contained within Ω_i^j, with this difference becoming arbitrarily small as r increases, the following bound is obtained for the quotient:

$$\frac{|\partial\underline{\Omega}_i^j|}{|\underline{\Omega}_i^j|} \leq \frac{(1+\delta_j)\|\tilde{\mathbf{n}}_i^j\|_1}{\min_{\mathbf{x}\in\partial\Omega_i^j}(\tilde{\mathbf{m}}_i^j \cdot \mathbf{n})} \tag{C.2}$$

Substituting this result into (C.1) and taking the limit as $r \to \infty$ — which implies $\delta_j \to 0$ — one obtains:

$$\|v\|^2_{\partial\Omega_i} \leq \frac{\sqrt{2}(p+1)(p+2)}{\min_{\mathbf{x}\in\partial\Omega_i}(\mathbf{m}_i \cdot \mathbf{n})} \|u\|_{\Omega_i} \tag{C.3}$$

Bibliography

[1] Yee KS. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation* 5 1966; **14**:302–307, doi:10.1109/TAP.1966.1138693.

[2] Taflove A, Hagness SC. *Computational electrodynamics: the finite-difference time-domain method; 3rd ed.*, vol. 80. Artech House, 2005.

[3] Yee KS, Chen JS. The finite-difference time-domain (fdtd) and the finite-volume time-domain (fvtd) methods in solving maxwell's equations. *IEEE Transactions on Antennas and Propagation* 1997; **45**:354–363, doi:10.1109/8.558651.

[4] Hachemi ME, Hassan O, Morgan K, Rowse DP, Weatherill NP. Hybrid methods for electromagnetic scattering simulations on overlapping grids. *Communications in Numerical Methods in Engineering* 8 2003; **19**:749–760, doi:10.1002/cnm.625.

[5] Sevilla R, Fernández-Méndez S, Huerta A. Nurbs-enhanced finite element method (nefem). *International Journal for Numerical Methods in Engineering* 10 2008; **76**:56–83, doi:10.1002/NME.2311.

[6] Sevilla R, Fernández-Méndez S, Huerta A. Comparison of high-order curved finite elements. *International Journal for Numerical Methods in Engineering* 8 2011; **87**:719–734, doi:10.1002/nme.3129.

[7] Dawson M, Sevilla R, Morgan K. The application of a high-order discontinuous galerkin time-domain method for the computation of electromag-

netic resonant modes. *Applied Mathematical Modelling* 3 2018; **55**:94–108, doi:10.1016/j.apm.2017.10.030.

[8] Xie ZQ, Sevilla R, Hassan O, Morgan K. The generation of arbitrary order curved meshes for 3d finite element analysis. *Computational Mechanics* 3 2013; **51**:361–374, doi:10.1007/s00466-012-0736-4.

[9] Sevilla R, Rees L, Hassan O. The generation of triangular meshes for nurbs-enhanced fem. *International Journal for Numerical Methods in Engineering* 2016; **108**:941–968, doi:10.1002/nme.5247.

[10] Zou X, Lo SB, Sevilla R, Hassan O, Morgan K. The generation of tetrahedral meshes for nurbs-enhanced fem. *Engineering with Computers* 12 2024; **40**:3949–3977, doi:10.1007/S00366-024-02004-Z.

[11] Reed WH, Hill TR. Triangular mesh methods for the neutron transport equation. *Technical Report*, Los Alamos Scientific Lab., N. Mex.(USA) 1973.

[12] Cockburn B, Karniadakis GE, Shu CW. The development of discontinuous galerkin methods. *Discontinuous Galerkin Methods*, Cockburn B, Karniadakis GE, Shu CW (eds.), Springer Berlin Heidelberg: Berlin, Heidelberg, 2000; 3–50, doi:10.1007/978-3-642-59721-3_1.

[13] Hesthaven JS, Warburton T. Nodal high-order methods on unstructured grids: I. time-domain solution of maxwell's equations. *Journal of Computational Physics* 9 2002; **181**:186–221, doi:10.1006/jcph.2002.7118.

[14] Sevilla R, Hassan O, Morgan K. The use of hybrid meshes to improve the efficiency of a discontinuous galerkin method for the solution of maxwell's equations. *Computers and Structures* 6 2014; **137**:2–13, doi:10.1016/j.compstruc.2013.01.014.

[15] Courant R, Friedrichs K, Lewy H. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development* 3 1967; **11**:215–234, doi:10.1147/rd.112.0215.

[16] Gottlieb S, Shu CW, Tadmor E. Strong stability-preserving high-order time discretization methods. *SIAM Review* 1 2001; **43**:89–112, doi:10.1137/S003614450036757X.

[17] Gottlieb S. On high order strong stability preserving runge-kutta and multi step time discretizations. *Journal of Scientific Computing* 11 2005; **25**:105–128, doi:10.1007/BF02728985.

[18] Grote MJ, Mitkova T. Explicit local time-stepping methods for maxwell's equations. *Journal of Computational and Applied Mathematics* 10 2010; **234**:3283–3302, doi:10.1016/j.cam.2010.04.028.

[19] Krivodonova L. An efficient local time-stepping scheme for solution of nonlinear conservation laws. *Journal of Computational Physics* 11 2010; **229**:8537–8551, doi:10.1016/j.jcp.2010.07.037.

[20] Piperno S. Symplectic local time-stepping in non-dissipative dgtd methods applied to wave propagation problems. *ESAIM: Mathematical Modelling and Numerical Analysis* 9 2006; **40**:815–841, doi:10.1051/m2an:2006035.

[21] Hochbruck M, Pažur T. Implicit runge–kutta methods and discontinuous galerkin discretizations for linear maxwell's equations. *SIAM Journal on Numerical Analysis* 1 2015; **53**:485–507, doi:10.1137/130944114.

[22] Kennedy CA, Carpenter MH. Diagonally implicit runge-kutta methods for ordinary differential equations. a review. *Technical Report*, Langley Research Center 2016.

[23] Ascher UM, Ruuth SJ, Spiteri RJ. Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics* 11 1997; **25**:151–167, doi:10.1016/S0168-9274(97)00056-1.

[24] Kanevsky A, Carpenter MH, Gottlieb D, Hesthaven JS. Application of implicit–explicit high order runge–kutta methods to discontinuous-galerkin schemes. *Journal of Computational Physics* 8 2007; **225**:1753–1781, doi:10.1016/j.jcp.2007.02.021.

[25] Dolean V, Fahs H, Fezoui L, Lanteri S. Locally implicit discontinuous galerkin method for time domain electromagnetics. *Journal of Computational Physics* 1 2010; **229**:512–526, doi:10.1016/j.jcp.2009.09.038.

[26] Descombes S, Lanteri S, Moya L. Locally implicit time integration strategies in a discontinuous galerkin method for maxwell's equations. *Journal of Scientific Computing* 7 2013; **56**:190–218, doi:10.1007/s10915-012-9669-5.

[27] Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: I. a stabilized lagrange multiplier method. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(41-44):2680–2686, doi:10.1016/j.cma.2010.05.011.

[28] Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: Ii. a stabilized nitsche method. *Applied Numerical Mathematics* 4 2012; **62**:328–341, doi:10.1016/j.apnum.2011.01.008.

[29] Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient finite element methodology based on cartesian grids: Application to structural shape optimization. *Abstract and Applied Analysis* 2013; **2013**:1–19, doi:10.1155/2013/953786.

[30] Tur M, Albelda J, Nadal E, Ródenas JJ. Imposing dirichlet boundary conditions in hierarchical cartesian meshes by means of stabilized lagrange multipliers. *International Journal for Numerical Methods in Engineering* 5 2014; **98**:399–417, doi:10.1002/nme.4629.

[31] Tur M, Albelda J, Marco O, Ródenas JJ. Stabilized method of imposing dirichlet boundary conditions using a recovered stress field. *Computer Methods in Applied Mechanics and Engineering* 11 2015; **296**:352–375, doi:10.1016/j.cma.2015.08.001.

[32] Tur M, Albelda J, Navarro-Jimenez JM, Rodenas JJ. A modified perturbed lagrangian formulation for contact problems. *Computational Mechanics* 4 2015; **55**:737–754, doi:10.1007/s00466-015-1133-6.

[33] Navarro-Jiménez JM, Tur M, Fuenmayor FJ, Ródenas JJ. On the effect of the contact surface definition in the cartesian grid finite element method. *Advanced Modeling and Simulation in Engineering Sciences* 12 2018; **5**:12, doi:10.1186/s40323-018-0108-5.

[34] Navarro-Jiménez JM, Tur M, Albelda J, Ródenas JJ. Large deformation frictional contact analysis with immersed boundary method. *Computational Mechanics* 10 2018; **62**:853–870, doi:10.1007/s00466-017-1533-x.

[35] Muñoz D, Ródenas JJ, Nadal E, Albelda J. 3d topology optimization with h-adaptive refinement using cartesian grids finite element method (cgfem). *EngOpt 2018 Proceedings of the 6th International Conference on Engineering Optimization*, Rodrigues H, Herskovits J, Mota Soares C, Araújo A, Guedes J, Folgado J, Moleiro F, Madeira JFA (eds.), Springer International Publishing: Cham, 2019; 778–788, doi:10.1007/978-3-319-97773-7_68.

[36] Muñoz D, Albelda J, Ródenas JJ, Nadal E. Improvement in 3d topology optimization with h-adaptive refinement using the cartesian grid finite element method. *International Journal for Numerical Methods in Engineering* 3 2021; :nme.6652doi:10.1002/nme.6652.

[37] Giovannelli L, Ródenas JJ, Navarro-Jimenez JM, Tur M. Element stiffness matrix integration in image-based cartesian grid finite element method. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8641 LNCS, Springer Verlag, 2014; 304–315, doi:10.1007/978-3-319-09994-1_31.

[38] Giovannelli L, Ródenas JJ, Navarro-Jiménez JM, Tur M. Direct medical image-based finite element modelling for patient-specific simulation of future implants. *Finite Elements in Analysis and Design* 11 2017; **136**:37–57, doi:10.1016/j.finel.2017.07.010.

[39] Gutiérrez-Gil J, Nadal E, León OM, González-Estrada OA, Albelda J, Rodenas JJ. Application of the cartesian grid finite element method to image-based bone remodelling process. *5th ECCOMAS Young Investigators Conference*, 2019; 1–2.

[40] Timmer H, Stern J. Computation of global geometric properties of solid objects. *Computer-Aided Design* 1980; **12**(6):301–304, doi:10.1016/0010-4485(80)90093-7.

[41] Cattani C, Paoluzzi A. Boundary integration over linear polyhedra. *Computer-Aided Design* 1990; **22**(2):130–135, doi:10.1016/0010-4485(90)90007-Y.

[42] Mirtich B. Fast and accurate computation of polyhedral mass properties. *Journal of graphics tools* 1996; **1**(2):31–50, doi:10.1080/10867651.1996.10487458.

[43] Rashid M, Gullett P. On a finite element method with variable element topology. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**(11-12):1509–1527, doi:10.1016/S0045-7825(00)00175-4.

[44] Dasgupta G. Integration within polygonal finite elements. *Journal of Aerospace Engineering* 2003; **16**(1):9–18, doi:10.1061/(ASCE)0893-1321(2003)16:1(9).

[45] Duczek S, Gabbert U. Efficient integration method for fictitious domain approaches. *Computational Mechanics* 2015; **56**:725–738, doi:10.1007/s00466-015-1197-3.

[46] Lasserre J. Integration on a convex polytope. *Proceedings of the American Mathematical Society* 1998; **126**(8):2433–2441, doi:10.1090/S0002-9939-98-04454-2.

[47] Lasserre J. Integration and homogeneous functions. *Proceedings of the American Mathematical Society* 1999; **127**(3):813–818, doi:10.1090/S0002-9939-99-04930-8.

[48] Mousavi S, Sukumar N. Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. *Computational Mechanics* 2011; **47**:535–554, doi:10.1007/s00466-010-0562-5.

[49] Chin EB, Lasserre JB, Sukumar N. Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra. *Computational Mechanics* 2015; **56**:967–981, doi:10.1007/s00466-015-1213-7.

[50] Chin EB, Sukumar N. An efficient method to integrate polynomials over polytopes and curved solids. *Computer Aided Geometric Design* 2020; **82**:101 914, doi:10.1016/j.cagd.2020.101914.

[51] Gunderman D, Weiss K, Evans JA. Spectral mesh-free quadrature for planar regions bounded by rational parametric curves. *Computer-Aided Design* 2021; **130**:102 944, doi:10.1016/j.cad.2020.102944.

[52] Xiao H, Gimbutas Z. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers & mathematics with applications* 2010; **59**(2):663–676, doi:10.1016/j.camwa.2009.10.027.

[53] Hubrich S, Di Stolfo P, Kudela L, Kollmannsberger S, Rank E, Schröder A, Düster A. Numerical integration of discontinuous functions: moment fitting and smart octree. *Computational Mechanics* 2017; **60**:863–881, doi:10.1007/s00466-017-1441-0.

[54] Mousavi S, Xiao H, Sukumar N. Generalized gaussian quadrature rules on arbitrary polygons. *International Journal for Numerical Methods in Engineering* 2010; **82**(1):99–113, doi:10.1002/nme.2759.

[55] Joulaian M, Hubrich S, Düster A. Numerical integration of discontinuities on arbitrary domains based on moment fitting. *Computational Mechanics* 2016; **57**:979–999, doi:10.1007/s00466-016-1273-3.

[56] Maday Y, Nguyen NC, Patera AT, Pau GS. A general multipurpose interpolation procedure: the magic points. *Communications on Pure and Applied Analysis* 2009; **8**(1):383–404, doi:10.3934/cpaa.2009.8.383.

[57] Legrain G. A nurbs enhanced extended finite element approach for unfitted cad analysis. *Computational Mechanics* 2013; **52**(4):913–929, doi:10.1007/s00466-013-0854-7.

[58] Legrain G. Non-negative moment fitting quadrature rules for fictitious domain methods. *Computers & Mathematics with Applications* 2021; **99**:270–291, doi:10.1016/j.camwa.2021.07.019.

[59] Garhuom W, Düster A. Non-negative moment fitting quadrature for cut finite elements and cells undergoing large deformations. *Computational Mechanics* 2022; **70**(5):1059–1081, doi:0.1007/s00466-022-02203-9.

[60] Parvizian J, Düster A, Rank E. Finite cell method: h-and p-extension for embedded domain problems in solid mechanics. *Computational Mechanics* 2007; **41**(1):121–133, doi:10.1007/s00466-007-0173-y.

[61] Divi SC, Verhoosel CV, Auricchio F, Reali A, Van Brummelen EH. Error-estimate-based adaptive integration for immersed isogeometric analysis. *Computers & Mathematics with Applications* 2020; **80**(11):2481–2516, doi:10.1016/j.camwa.2020.03.026.

[62] Dréau K, Chevaugeon N, Moës N. Studied x-fem enrichment to handle material interfaces with higher order finite element. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(29-32):1922–1936, doi:10.1016/j.cma.2010.01.021.

[63] Marco O, Sevilla R, Zhang Y, Ródenas JJ, Tur M. Exact 3d boundary representation in finite element analysis based on cartesian grids independent of the geometry. *International Journal for Numerical Methods in Engineering* 8 2015; **103**:445–468, doi:10.1002/NME.4914.

[64] Antolin P, Wei X, Buffa A. Robust numerical integration on curved polyhedra based on folded decompositions. *Computer Methods in Applied Mechanics and Engineering* 2022; **395**:114 948, doi:10.1016/j.cma.2022.114948.

[65] Thiagarajan V, Shapiro V. Adaptively weighted numerical integration over arbitrary domains. *Computers & Mathematics with Applications* 2014; **67**(9):1682–1702, doi:10.1016/j.camwa.2014.03.001.

[66] Scholz F, Mantzaflaris A, Jüttler B. First order error correction for trimmed quadrature in isogeometric analysis. *Advanced Finite Element Methods with Applications: Selected Papers from the 30th Chemnitz Finite Element Symposium 2017 30*, Springer, 2019; 297–321, doi:10.1007/978-3-030-14244-5_15.

[67] Eriksson K, Estep D, Hansbo P, Johnson C. Introduction to adaptive methods for differential equations. *Acta numerica* 1995; **4**:105–158, doi:10.1017/S0962492900002531.

[68] Bangerth W, Rannacher R. *Adaptive finite element methods for differential equations*. Springer Science & Business Media, 2003, doi:10.1007/978-3-0348-7605-6.

[69] Ainsworth M, Oden JT. *A Posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, 2011, doi:10.1016/S0045-7825(96)01107-3.

[70] Chamoin L, Díez P. *Verifying Calculations-Forty Years On: An Overview of Classical Verification Techniques for FEM Simulations*. Springer, 2015, doi:10.1007/978-3-319-20553-3.

[71] Hairer E, Wanner G, Nørsett SP. *Solving Ordinary Differential Equations I: Nonstiff Problems*, chap. Runge-Kutta and Extrapolation Methods. Springer Berlin Heidelberg: Berlin, Heidelberg, 1993; 129–353, doi:10.1007/978-3-540-78862-1_2.

[72] Díez P, Huerta A. A unified approach to remeshing strategies for finite element h-adaptivity. *Computer Methods in Applied Mechanics and Engineering* 1999; **176**(1):215–229, doi:https://doi.org/10.1016/S0045-7825(98)00338-7.

[73] Roy CJ, Oberkampf WL. A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing. *Computer Methods in Applied Mechanics and Engineering* 2011; **200**(25):2131–2144, doi:https://doi.org/10.1016/j.cma.2011.03.016.

[74] Phillips TS, Roy CJ. Richardson extrapolation-based discretization uncertainty estimation for computational fluid dynamics. *Journal of Fluids Engineering* 2014; **136**(12):121 401, doi:10.1115/1.4027353.

[75] Zienkiewicz OC, Zhu JZ. A simple error estimator and adaptive procedure for practical engineerng analysis. *International journal for numerical methods in engineering* 1987; **24**(2):337–357, doi:10.1002/nme.1620240206.

[76] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1331–1364, doi:10.1002/nme.1620330702.

[77] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1365–1382, doi:10.1002/nme.1620330703.

[78] Blacker T, Belytschko T. Superconvergent patch recovery with equilibrium and conjoint interpolant enhancements. *International Journal for Numerical Methods in Engineering* 1994; **37**(3):517–536, doi:10.1002/nme.1620370309.

[79] Boroomand B, Zienkiewicz OC. Recovery by equilibrium in patches (rep). *International journal for numerical methods in engineering* 1997; **40**(1):137–164, doi:10.1002/(SICI)1097-0207(19970115)40:1<137::AID-NME57>3.0.CO;2-5.

[80] Ródenas J, Tur M, Fuenmayor F, Vercher A. Improvement of the superconvergent patch recovery technique by the use of constraint equations: the spr-c technique. *International Journal for Numerical Methods in Engineering* 2007; **70**(6):705–727, doi:10.1002/nme.1903.

[81] Babuška I, Rheinboldt WC. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis* 1978; **15**(4):736–754, doi:10.1137/0715049.

[82] Babuska I, Szabo BA, Katz IN. The p-version of the finite element method. *SIAM journal on numerical analysis* 1981; **18**(3):515–545, doi:10.1137/0718033.

[83] Kelly DW, De SR Gago J, Zienkiewicz OC, Babuska I. A posteriori error analysis and adaptive processes in the finite element method: Part i—error analysis. *International journal for numerical methods in engineering* 1983; **19**(11):1593–1619, doi:10.1002/nme.1620191103.

[84] Becker R, Rannacher R. An optimal control approach to a posteriori error estimation in finite element methods. *Acta numerica* 2001; **10**:1–102, doi:10.1017/S0962492901000010.

[85] Gui Wz, Babuška I. The h, p and hp versions of the finite element method in 1 dimension: Part i. the error analysis of the p-version. *Numerische Mathematik* 1986; **49**(6):577–612, doi:10.1007/BF01389733.

[86] Gui Wz, Babuška I. The h, p and hp versions of the finite element method in 1 dimension: Part ii. the error analysis of the h-and hp versions. *Numerische Mathematik* 1986; **49**:613–657, doi:10.1007/BF01389734.

[87] Gui Wz, Babuška I. The h, p and hp versions of the finite element method in 1 dimension: Part iii. the adaptive hp version. *Numerische Mathematik* 1986; **49**:659–683, doi:10.1007/BF01389735.

[88] Demkowicz L, Devloo P, Oden J. On an h-type mesh-refinement strategy based on minimization of interpolation errors. *Computer Methods in Applied Mechanics and Engineering* 1985; **53**(1):67–89, doi:10.1016/0045-7825(85)90076-3.

[89] Zhu J, Zienkiewicz O. Adaptive techniques in the finite element method. *Communications in applied numerical methods* 1988; **4**(2):197–204, doi:10.1002/cnm.1630040210.

[90] Bouillard P, Ihlenburg F. Error estimation and adaptivity for the finite element method in acoustics: 2d and 3d applications. *Computer methods in applied mechanics and engineering* 1999; **176**(1-4):147–163, doi:10.1016/S0045-7825(98)00334-X.

[91] Bautista JG, de la Llave Plata M, Couaillier V, Visonneau M, Schneider K. h-adaptation for high-order discontinuous galerkin schemes built on local multiwavelet analysis. *Computers & Fluids* 2023; **256**:105 844, doi:10.1016/j.compfluid.2023.105844.

[92] A Szabó B. Mesh design for the p-version of the finite element method. *Computer Methods in Applied Mechanics and Engineering* 1986; **55**(1):181–197, doi:10.1016/0045-7825(86)90091-5.

[93] Düster A, Rank E, Szabó B. The p-version of the finite element and finite cell methods. *Encyclopedia of computational mechanics* 2017; **3**(Part 1):137–171, doi:10.1002/9781119176817.ecm2003g.

[94] Babuška I, Guo BQ. Regularity of the solution of elliptic problems with piecewise analytic data. part i. boundary value problems for linear elliptic equation of second order. *SIAM Journal on Mathematical Analysis* 1988; **19**(1):172–203, doi:10.1137/0519014.

[95] Costabel M, Dauge M, Schwab C. Exponential convergence of hp-fem for maxwell equations with weighted regularization in polygonal domains. *Mathematical Models and Methods in Applied Sciences* 2005; **15**(04):575–622, doi:10.1142/S0218202505000480.

[96] Babuska I, Szabo B. On the rates of convergence of the finite element method. *International Journal for Numerical Methods in Engineering* 1982; **18**(3):323–341, doi:10.1002/nme.1620180302.

[97] Babuška I, Dorr MR. Error estimates for the combined h and p versions of the finite element method. *Numerische Mathematik* 1981; **37**:257–277, doi:10.1007/BF01398256.

[98] Bangerth W, Kayser-Herold O. Data structures and requirements for hp finite element software. *ACM Transactions on Mathematical Software (TOMS)* 2009; **36**(1):1–31, doi:10.1145/1486525.1486529.

[99] Demkowicz L. Fully automatic hp-adaptivity for maxwell's equations. *Computer methods in applied mechanics and engineering* 2005; **194**(2-5):605–624, doi:10.1016/j.cma.2004.05.023.

[100] Hovhannisyan N, Müller S, Schäfer R. Adaptive multiresolution discontinuous galerkin schemes for conservation laws. *Mathematics of computation* 2014; **83**(285):113–151, doi:10.1090/S0025-5718-2013-02732-9.

[101] Adjerid S, Aiffa M, Flaherty JE. Computational methods for singularly perturbed systems. *Proceedings of Symposia in Applied Mathematics*, vol. 56, American Mathematical Society, 1999; 47–84, doi:10.1090/psapm/056/1718897.

[102] Houston P, Süli E. hp-adaptive discontinuous galerkin finite element methods for first-order hyperbolic problems. *SIAM Journal on Scientific Computing* 2001; **23**(4):1226–1252, doi:10.1137/s1064827500378799.

[103] Oden JT, Patra A. A parallel adaptive strategy for hp finite element computations. *Computer methods in applied mechanics and engineering* 1995; **121**(1-4):449–470, doi:10.1016/0045-7825(94)00705-r.

[104] Mavriplis C. Adaptive mesh strategies for the spectral element method. *Computer methods in applied mechanics and engineering* 1994; **116**(1-4):77–86, doi:10.1016/s0045-7825(94)80010-3.

[105] Houston P, Senior B, Süli E. Sobolev regularity estimation for hp-adaptive finite element methods. *Numerical Mathematics and Advanced Applications*, Brezzi F, Buffa A, Corsaro S, Murli A (eds.), Springer Milan: Milano, 2003; 631–656, doi:10.1007/978-88-470-2089-4_58.

[106] Chalmers N, Agbaglah G, Chrust M, Mavriplis C. A parallel hp-adaptive high order discontinuous galerkin method for the incompressible navier-stokes equations. *Journal of Computational Physics: X* 2019; **2**:100 023, doi:https://doi.org/10.1016/j.jcpx.2019.100023.

[107] Balanis CA. *Advanced engineering electromagnetics*. Wiley, 1989.

[108] Givoli D. *Numerical methods for problems in infinite domains*, vol. 33. Elsevier, 2013, doi:10.1016/c2009-0-08826-6.

[109] Berenger JP. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics* 10 1994; **114**:185–200, doi:10.1006/jcph.1994.1159.

[110] Jin JM, Chew WC. Combining pml and abc for the finite-element analysis of scattering problems. *Microwave and Optical Technology Letters* 7 1996; **12**:192–197, doi:10.1002/(SICI)1098-2760(199607)12:4<192::AID-MOP4>3.0.CO;2-B.

[111] Bonnet F, Poupaud F. Berenger absorbing boundary condition with time finite-volume scheme for triangular meshes. *Applied Numerical Mathematics* 12 1997; **25**:333–354, doi:10.1016/S0168-9274(97)00000-7.

[112] Chew WC, Weedon WH. A 3d perfectly matched medium from modified maxwell's equations with stretched coordinates. *Microwave and optical technology letters* 1994; **7**(13):599–604, doi:10.1002/mop.4650071304.

[113] Kabakian AV, Shankar V, Hall WF. Unstructured grid-based discontinuous galerkin method for broadband electromagnetic simulations. *Journal of Scientific Computing* 6 2004; **20**:405–431, doi:10.1023/B:JOMP.0000025932.17082.18.

[114] Abarbanel S, Gottlieb D. A mathematical analysis of the pml method. *Journal of Computational Physics* 1997; **134**(2):357–363, doi:10.1006/jcph.1997.5717.

[115] xin Fan G, Liu QH. A strongly well-posed pml in lossy media. *IEEE Antennas and Wireless Propagation Letters* 2003; **2**:97–100, doi:10.1109/LAWP.2003.814776.

[116] Rabinovich D, Givoli D, Bécache E. Comparison of high-order absorbing boundary conditions and perfectly matched layers in the frequency domain. *International Journal for Numerical Methods in Biomedical Engineering* 2010; **26**(10):1351–1369, doi:10.1002/cnm.1394.

[117] Donea J, Huerta A. *Finite Element Methods for Flow Problems*. Wiley, 2003, doi:10.1002/0470013826.

[118] Hesthaven JS, Warburton T. *Nodal Discontinuous Galerkin Methods*, vol. 54. Springer New York, 2008, doi:10.1007/978-0-387-72067-8.

[119] Pietro DAD, Ern A. *Mathematical Aspects of Discontinuous Galerkin Methods*, vol. 69. Springer Berlin Heidelberg, 2012, doi:10.1007/978-3-642-22980-0.

[120] Godunov SK, Bohachevsky I. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Matematičeskij sbornik* 1959; **47(89)**(3):271–306.

[121] Lax PD. Hyperbolic systems of conservation laws ii. *Communications on Pure and Applied Mathematics* 1957; **10**:537–566, doi:10.1002/cpa.3160100406.

[122] Roe PL. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics* 10 1981; **43**:357–372, doi:10.1016/0021-9991(81)90128-5.

[123] Harten A, Lax PD, van Leer B. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM Review* 1 1983; **25**:35–61, doi:10.1137/1025002.

[124] LeVeque RJ. *Finite volume methods for hyperbolic problems*, vol. 31. Cambridge university press, 2002, doi:10.1017/cbo9780511791253.

[125] Toro EF. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Berlin Heidelberg, 2009, doi:10.1007/b79761.

[126] Arnold DN, Brezzi F, Cockburn B, Marini LD. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM journal on numerical analysis* 2002; **39**(5):1749–1779, doi:10.1137/s0036142901384162.

[127] Fernández-Méndez S, Huerta A. Imposing essential boundary conditions in mesh-free methods. *Computer methods in applied mechanics and engineering* 2004; **193**(12-14):1257–1275, doi:10.1016/j.cma.2003.12.019.

[128] Lu K, Augarde CE, Coombs WM, Hu Z. Weak impositions of dirichlet boundary conditions in solid mechanics: a critique of current approaches and extension to partially prescribed boundaries. *Computer Methods in Applied Mechanics and Engineering* 2019; **348**:632–659, doi:10.1016/j.cma.2019.01.035.

[129] Babuška I. The finite element method with penalty. *Mathematics of computation* 1973; **27**(122):221–228, doi:10.2307/2005611.

[130] Zhu T, Atluri SN. A modified collocation method and a penalty formulation for enforcing the essential boundary conditions in the element free galerkin method. *Computational Mechanics* 1998; **21**(3):211–222, doi:10.1007/s004660050296.

[131] Babuška I. The finite element method with lagrangian multipliers. *Numerische Mathematik* 1973; **20**(3):179–192, doi:10.1007/bf01436561.

[132] Nitsche J. Über ein variationsprinzip zur lösung von dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 1971; **36**(1):9–15, doi:10.1007/BF02995904.

[133] Hansbo P, Larson MG. Discontinuous galerkin methods for incompressible and nearly incompressible elasticity by nitsche's method. *Computer methods in applied mechanics and engineering* 2002; **191**(17-18):1895–1908, doi:10.1016/s0045-7825(01)00358-9.

[134] Johansson A, Larson MG. A high order discontinuous galerkin nitsche method for elliptic problems with fictitious boundary. *Numerische Mathematik* 4 2013; **123**:607–628, doi:10.1007/s00211-012-0497-1.

[135] Nadal E. Cartesian grid fem (cgfem): high performance h-adaptive fe analysis with efficient error control. application to structural shape optimization. PhD Thesis, Universitat Politècnica de València 2014.

[136] Sevilla R, Fernández-Méndez S, Huerta A. Nurbs-enhanced finite element method for euler equations. *International Journal for Numerical Methods in Fluids* 7 2008; **57**:1051–1069, doi:10.1002/fld.1711.

[137] Sommariva A, Vianello M. Computing approximate fekete points by qr factorizations of vandermonde matrices. *Computers and Mathematics with Applications* 4 2009; **57**:1324–1336, doi:10.1016/j.camwa.2008.11.011.

[138] Bos L, Marchi SD, Sommariva A, Vianello M. Computing multivariate fekete and leja points by numerical linear algebra. *SIAM Journal on Numerical Analysis* 1 2010; **48**:1984–1999, doi:10.1137/090779024.

[139] Hansbo A, Hansbo P. A finite element method for the simulation of strong and weak discontinuities in solid mechanics. *Computer Methods in Applied Mechanics and Engineering* 8 2004; **193**:3523–3540, doi:10.1016/j.cma.2003.12.041.

[140] Burman E, Claus S, Hansbo P, Larson MG, Massing A. Cutfem: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering* 11 2015; **104**:472–501, doi:10.1002/nme.4823.

[141] Piegl L, Tiller W. *The NURBS Book*. Springer Berlin Heidelberg, 1997, doi:10.1007/978-3-642-59223-2.

[142] Rogers DF. *An introduction to NURBS : with historical perspective*. Morgan Kaufmann Publishers, 2001.

[143] Mick H. NURBS Toolbox by D.M. Spink. `https://www.mathworks.com/matlabcentral/fileexchange/26390-nurbs-toolbox-by-d-m-spink` 2025. MATLAB Central File Exchange. Accessed: 2025-07-16.

[144] Nguyen VP, Anitescu C, Bordas SP, Rabczuk T. Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation* 11 2015; **117**:89–116, doi:10.1016/J.MATCOM.2015.05.008.

[145] Hughes T, Cottrell J, Bazilevs Y. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 10 2005; **194**:4135–4195, doi:10.1016/j.cma.2004.10.008.

[146] Lorensen WE, Cline HE. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, Association for Computing Machinery: New York, NY, USA, 1987; 163–169, doi:10.1145/37401.37422.

[147] Lee D. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing* 5 1983; **22**:207–221, doi:10.1016/0734-189X(83)90065-8.

[148] Joe B, Simpson RB. Corrections to lee's visibility polygon algorithm. *BIT* 12 1987; **27**:458–473, doi:10.1007/BF01937271.

[149] Peterson TE. A note on the convergence of the discontinuous galerkin method for a scalar hyperbolic equation. *SIAM Journal on Numerical Analysis* 2 1991; **28**:133–140, doi:10.1137/0728006.

[150] Lasaint P, Raviart PA. On a finite element method for solving the neutron transport equation. *Mathematical Aspects of Finite Elements in Partial Differential Equations*, de Boor C (ed.). Academic Press, 1974; 89–123, doi:10.1016/B978-0-12-208350-1.50008-X.

[151] Richter GR. An optimal-order error estimate for the discontinuous galerkin method. *Mathematics of Computation* 1 1988; **50**:75, doi:10.2307/2007915.

[152] Penrose R. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society* 7 1955; **51**:406–413, doi:10.1017/S0305004100030401.

[153] Marco O, Ródenas JJ, Navarro-Jiménez JM, Tur M. Robust h-adaptive meshing strategy considering exact arbitrary cad geometries in a cartesian grid framework. *Computers and Structures* 12 2017; **193**:87–109, doi:10.1016/j.compstruc.2017.08.004.

[154] Witherden FD, Vincent PE. On the identification of symmetric quadrature rules for finite element methods. *Computers and Mathematics with Applications* 5 2015; **69**:1232–1241, doi:10.1016/J.CAMWA.2015.03.017.

[155] Jaśkowiec J, Sukumar N. High-order symmetric cubature rules for tetrahedra and pyramids. *International Journal for Numerical Methods in Engineering* 1 2021; **122**:148–171, doi:10.1002/NME.6528.

[156] Jaśkowiec J, Sukumar N. Addendum to the paper "high-order symmetric cubature rules for tetrahedra and pyramids". *International Journal for Numerical Methods in Engineering* 4 2021; **122**:1875–1883, doi:10.1002/nme.6600.

[157] Wandzura S, Xiao H. Symmetric quadrature rules on a triangle. *Computers and Mathematics with Applications* 2003; **45**:1829–1840, doi:10.1016/S0898-1221(03)90004-6.

[158] Frey PJ, George P. *Mesh Generation*. Wiley, 2008, doi:10.1002/9780470611166.

[159] Zienkiewicz O, Taylor R, Zhu JZ. *The Finite Element Method: its Basis and Fundamentals*. Elsevier, 2013, doi:10.1016/C2009-0-24909-9.

[160] Prabhune B, Suresh K. A computationally efficient isoparametric tangled finite element method for handling inverted quadrilateral and hexahedral elements. *Computer Methods in Applied Mechanics and Engineering* 2 2023; **405**:115 897, doi:10.1016/J.CMA.2023.115897.

[161] Gatto P, Demkowicz L. Construction of h1-conforming hierarchical shape functions for elements of all shapes and transfinite interpolation. *Finite Elements in Analysis and Design* 6 2010; **46**:474–486, doi:10.1016/J.FINEL.2010.01.005.

[162] Warburton T. An explicit construction of interpolation nodes on the simplex. *Journal of Engineering Mathematics* 11 2006; **56**:247–262, doi:10.1007/S10665-006-9086-6/METRICS.

[163] Kubatko EJ, Yeager BA, Ketcheson DI. Optimal strong-stability-preserving runge-kutta time discretizations for discontinuous galerkin methods. *Journal of Scientific Computing* 10 2014; **60**:313–344, doi:10.1007/S10915-013-9796-7.

[164] Houston P, Schwab C, Süli E. Discontinuous hp-finite element methods for advection-diffusion-reaction problems. *SIAM Journal on Numerical Analysis* 2002; **39**(6):2133–2163, doi:10.1137/s0036142900374111.

[165] Castillo P, Cockburn B, Schötzau D, Schwab C. Optimal a priori error estimates for the *hp*-version of the local discontinuous galerkin method for convection–diffusion problems. *Mathematics of computation* 2002; **71**(238):455–478, doi:10.1090/s0025-5718-01-01317-5.

[166] Cheng Y, Meng X, Zhang Q. Application of generalized gauss–radau projections for the local discontinuous galerkin method for linear convection-diffusion equations. *Mathematics of Computation* 2017; **86**(305):1233–1267, doi:10.1090/mcom/3141.

[167] Cangiani A, Dong Z, Georgoulis EH, Houston P. hp-version discontinuous galerkin methods for advection-diffusion-reaction problems on polytopic meshes. *ESAIM: Mathematical Modelling and Numerical Analysis* 2016; **50**(3):699–725, doi:10.1051/m2an/2015059.

[168] Cangiani A, Dong Z, Georgoulis E. *hp*-version discontinuous galerkin methods on essentially arbitrarily-shaped elements. *Mathematics of Computation* 2022; **91**(333):1–35, doi:10.1090/mcom/3667.

[169] Carstensen C, Sauter SA. A posteriori error analysis for elliptic pdes on domains with complicated structures. *Numerische Mathematik* 2004; **96**:691–721, doi:10.1007/s00211-003-0495-4.

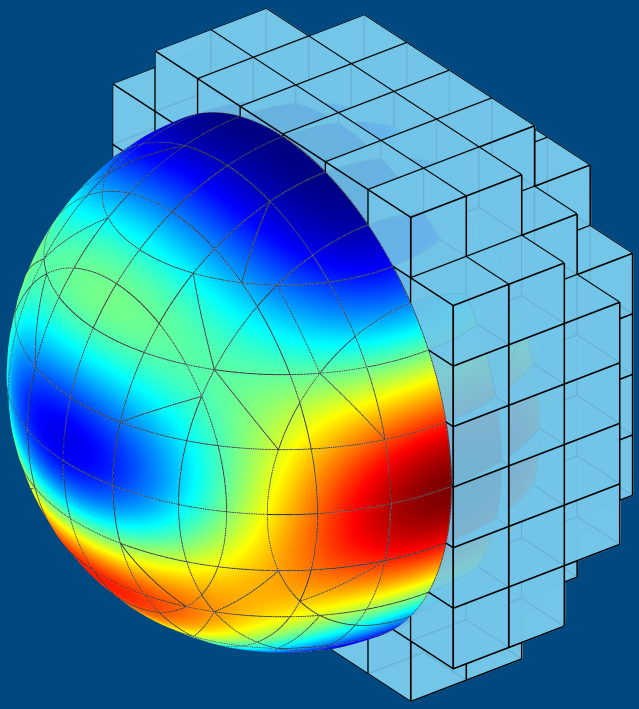
[170] Cockburn B, Shu CW. The local discontinuous galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis* 12 1998; **35**:2440–2463, doi:10.1137/S0036142997316712.

[171] Duffy MG. Quadrature over a pyramid or cube of integrands with a singularity at a vertex. *SIAM Journal on Numerical Analysis* 12 1982; **19**:1260–1262, doi:10.1137/0719090.

[172] de Falco C, Reali A, Vázquez R. Geopdes: A research tool for isogeometric analysis of pdes. *Advances in Engineering Software* 12 2011; **42**:1020–1034, doi:10.1016/J.ADVENGSOFT.2011.06.010.

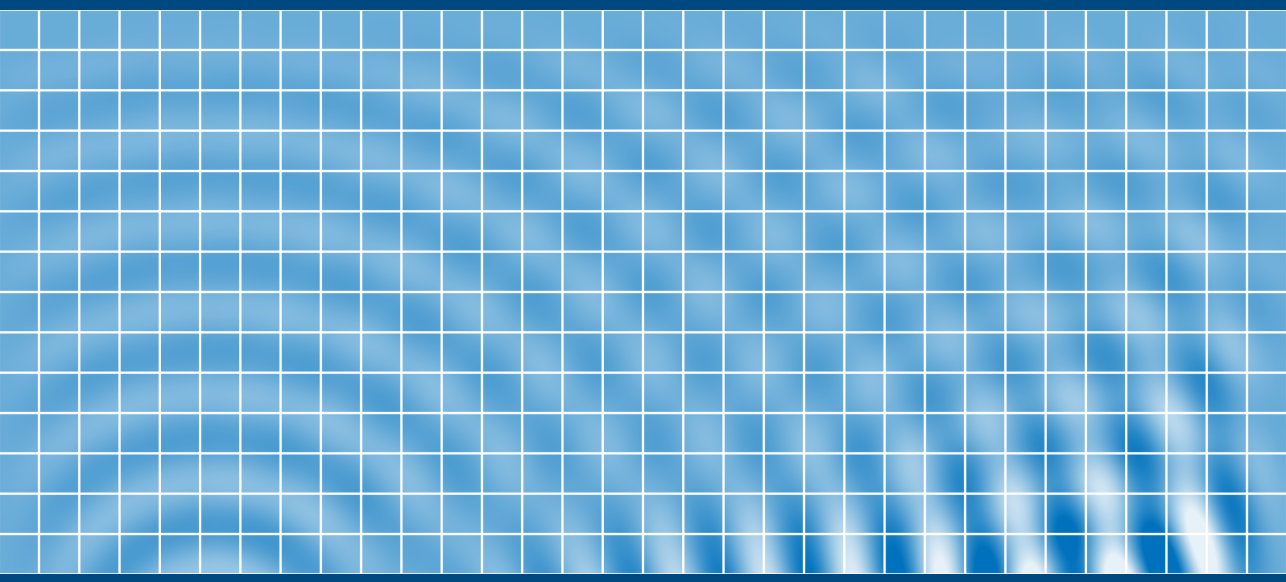
[173] Spink M, Claxton D, de Falco C, Vázquez R. The nurbs toolbox. URL `http://octave.sourceforge.net/nurbs/index.html`.

[174] Marvi MSP. Nurbs. URL `https://github.com/Mspmarvi/NURBS/releases/tag/1.0.0`.



"It is difficult to say what is impossible,
for the dream of yesterday is the hope of today
and the reality of tomorrow."

Robert H. Goddard



UNIVERSITAT POLITÈCNICA DE VALÈNCIA