HERMESH

A Geometrical Domain Composition Method in Computational Mechanics

HERMESH

A Geometrical Domain Composition Method in Computational Mechanics

Ane Beatriz Eguzkitza
Degree in Physical Sciences

A Thesis presented for the degree of
Doctor in Department of Computer Architecture

Directed by Dr. Guillaume Houzeaux
Tutor: Dr. Jose Maria Cela Espín

Universitat Politècnica de Catalunya

2014

A Amatxu...

Resumen

En este trabajo presentamos el metodo **HERMESH** al que hemos catalogado como un *método de composición de dominios* puesto que a partir de mallas independientes se obtiene una solución global del problema como la unión de los subproblemas que forman las mallas independientes. Como resultado, la malla global mantiene el mismo número de grados de libertad que la suma de los grados de libertad de las mallas independientes, las cuales se acoplan en las interfases internas a través de nuevos elementos a los que nos referimos como *elementos de extensión*. Por este motivo decimos que el método de composición de dominio es geométrico. El resultado de la malla global es una malla que no es conforme en las interfases entre las distintas mallas debido a las nuevas conectividades generadas sobre los nodos existentes.

Los requerimientos de partida fueron que el método se implemente de forma implícita, sea válido para cualquier PDE y no implique ningún esfuerzo adicional ni perdida de eficiencia para el funcionamiento paralelo del código de altas prestaciones en el que ha sido implementado. Creemos que estas propiedades son las principales aportaciones de esta tesis dentro del marco de acoplamiento de mallas en mecánica computacional.

A partir de estas premisas, hemos conseguido una herramienta automática e independiente de la topología para componer mallas. Es capaz de acoplar sin necesidad de intervención del usuario, mallas con solapamiento parcial o total así como mallas disjuntas con o sin "gap" entre ellas. También hemos visto que ofrece cierta flexibilidad en relación al tamaños relativos entre las mallas siendo un método válido como técnica de remallado local.

Presentamos una descripción detallada de la implementación de esta técnica, llevada a cabo en un código de altas prestaciones de mecánica computa-

cional en el contexto de elementos finitos, **Alya**. Se demostrarán todas las propiedades numéricas que ofrece el métodos a través de distintos problemas tipo benchmark y el método de la solución manufacturada.

Finalmente se mostrarán los resultados en problemas complejos resueltos con el método HERMESH, que a su vez es una prueba de la gran flexibilidad que nos brinda.

Abstract

With this thesis we present the **HERMESH** method which has been classified by us as a *a composition domain method*. This term comes from the idea that HERMESH obtains a global solution of the problem from two independent meshes as a result of the mesh coupling. The global mesh maintains the same number of degrees of freedom as the sum of the independent meshes, which are coupled in the interfaces via new elements referred to by us as extension elements. For this reason we enunciate that the domain composition method is *geometrical*. The result of the global mesh is a non-conforming mesh in the interfaces between independent meshes due to these new connectivities formed with existing nodes and represented by the new *extension elements*.

The first requirements were that the method be implicit, be valid for any partial differential equation and not imply any additional effort or loss in efficiency in the parallel performance of the code in which the method has been implemented. In our opinion, these properties constitute the main contribution in mesh coupling for the computational mechanics framework.

From these requirements, we have been able to develop an automatic and topology-independent tool to compose independent meshes. The method can couple overlapping meshes with minimal intervention on the user's part. The overlapping can be partial or complete in the sense of overset meshes. The meshes can be disjoint with or without a gap between them. And we have demonstrated the flexibility of the method in the relative mesh size.

In this work we present a detailed description of HERMESH which has been implemented in a high-performance computing computational mechanics code within the framework of the finite element methods. This code is called **Alya**. The numerical properties will be proved with different benchmark-type

problems and the manufactured solution technique.

Finally, the results in complex problems solved with HERMESH will be presented, clearly showing the versatility of the method.

Acknowledgements

Seré incapaz de agradecer todo lo que me han dado estos años. Ni con las 200 páginas de esta tesis lo lograría. Pero he pensado que mejor escribir algo.

Gracias José María Cela, por aquella primera entrevista en que me preguntabas porqué quería hacer este doctorado y me advertías que tenía que ser capaz de hablar con un informático. Gracias a Frederic Magoulès por acogerme en su centro y reservarme una mesa. Gracias por, sin saberlo, darme la inspiración para el nombre del método propuesto. Gracias a la Fundación Iberdrola, en especial a Luís Prieto, por tenernos fe. Gracias al tribunal de la pre-lectura, Jorge Vidal y Josep Ramón Herrero, por ofrecerse a validar esta tesis. A Philip por sus noches sin dormir corrigiendo el inglés.

Los compañeros de oficina. Bueno, de las oficinas, que he recorrido unas cuantas... Gracias a aquella primera generación que nos dedicabamos a romper los conductos de ventilación o dónde se montaban partidas de juegos de rol en red.... Rogeli, con su ordenata ferrari, Raul, nuestro Raulitooooo...sigue deleitandonos con tus pasteles!, Albert, ese txavalote!, Xavi(Junior), quantes hores intenses i ens tornem a trobar, Félixxxxx, o millor Felisssssssss, Miquel Català, el xicot perfecte... Luego vino la nueva ola, ya no rodeada de mis computines... amb tú, Cristina, tota una caixa de consells, Xavi, fins que et vem tenir amb nosaltres, gràcies pels nostres trapis de música, Alexis amb les teves pantunfles japos, Chiara porque aunque no tengamos el gusto de tenerte cada día, tu aurea nos acompaña con el mejor de los ambientes! (y nunca olvideré el cambio de mesa), Cristóbal por las risas viendo pasar a Mr. Chuches y los grandes momentos en la sala. Pooo, el hombre adjunto! gracias por tus consejos de optimización! Àlex la teva predisposició a fer un cop de mà és inmensa! Matias el cuerdo, por tu gran *sabor fairy!*, Miguel, siempre ahí con un gran saludo, Edgar, ja no ens podem fer copets i Jordi, el noi de la bici amb clase!.
Gràcies Arnau per confiar en els nostres patchis! Aivalaostia, patchi! Herbert, el justiciero y siempre de graaaan ayuda! Toniiiiii, què faríem sense el Toniii!!!!!!!

Fer, mil gracias por esa primera gran conversación de ascensor en que me transmitías seguridad y por todo lo que sigue! Sol, gracias por esa mesa que iluminó a esta tesis con luz del sol. Guille, nuestro artista poniendo una nota de color. Mil gracias por ayudarme con la tapa de esta tesis. Y Roman..., el hombre de las mallas! HERMESH nació contigo! Gracias a todos por hacer tan agradables los días por aquí.

Aunque sin la ayuda de todos los que me han apoyado fuera de estas paredes, no creo que hubiera tenido fuerzas para llegar hasta aquí. Gràcies a les *petsui!* per no entendre mai el que faig i continuar preguntant-m'ho! Gemmeta, sempre a dins i aprenent juntes. Lluís, un placer crecer juntos y seguir compartiendo nuestras ilusiones. Marta que nos iluminas con tu sonrisa... y con Uma! la niña de nuestros ojos! Arturo y Javiera, siempre sinónimo de experiencia sensorial. Conchi y Lluís, qué grandes conversaciones con gin-tonic en mano comprendiendo el mundo y...que es bueno acabar algo... A mi familia, la pH! Rubitaaaa, me has acompañado desde que empecé con todo esto y sin tí nada hubiera sido igual. David, sempre amenitzant amb el teu gran humor, compartint els tupers a la gespa del campus, els teus gran tiberis a la terrassa però sobretot compartint la nostre amistat i... a la Pati! que me abrió la puertas a las costuritas y a aprender de otra manera de ver el mundo. Ana, que como la historia es circular, nos encontrábamos en mis inicios de esta aventura, y ahora la vida nos ha vuelto a unir. (Qué bonito!) Triky&Bea, siempre acogiendonos con la puerta abierta allí donde esteis, nuestro otro hogar.

Mariano y Guillaume, os pondría como conclusiones de esta tesis, en tanto que es lo mejor que he sacado de ella. No hay palabras para agradeceros todos estos años. Desde Cuba (*yatusabes...maifriend...*, con el mundo interior de Dudu...) hasta hoy y aquí.

Mariano, porque "tú y yo no nos peleamos". Gracias por todos los partidos, los bar-reños, las pelis, el plà de barri y en definitiva, por todos los momentos compartidos. Gracias por escucharme, incluso con mis discursos inconexos y compartir a los tuyos... Silvina! gracias por esos chats llenos de energía.

Guillaume, me lo has enseñado todo. Aunque esto te haya podido costar en algunos casos, ..., gracias por la paciencia. Si, gracias por la paciencia. Gracias por toda la atención y hospitalidad. Por confiar en mí, por tu sinceridad y siempre dispuesto a llamar en mis momentos de crisis! Tu ilusión ha sido un gran motor en este trabajo. Compartir todos estos años contigo ha sido más que esta tesis. Tantos cafes, tantas fotos, tantos calders, viajes ... Conocer a Ana y Mateo!. Ana..., gracias por la cuenta que te trae. Has sido una pieza clave en esta tesis!. Mateo, gracias por hacer que aun mantenga mis toques con el balón! A Christian y Michèle, nuestra familia en París.

Y aunque seguro que me he dejado muchos gracias tengo que ir ya acabando

porque por cierto,...*tengo una tesis que acabar!*... Gracias a mi familia; la de Bilbao, Elvi, Óscar, Berta y Jon que siempre me han esperado a que baje con la mantita del avión. Jon, gracias por acabar haciendonos una visita en Barcelona y siempre tenernos un 4Kilos para compartir con los catalanes. I la d'aquí; gràcies Cristina i Jaume. No només per donar-me lo millor que m'ha pasat mai, el vostre fill. Sinó per tot el que directament m'heu donat des del primer dia que ens vem conéixer, que el situem oficialment al Taktika Berri (és clar que si! començant mooolt bé, com no podia ser d'una altre manera, a un vasc!). Aquesta és una mostra de la vostre inmensa sensibilitat, feta palesa en les grans converses sempre al voltant d'un bon àpat! Heu fet que la meva feina tingui encara més valor. Per com heu cregut en ella. I ens heu donat totes les comoditats i ajudes per a que l'esforç sigui més fàcil de dur. Un esforç que al teu costat, Cristian s'atenua. Gràcies per donar-me aquests sogres. Les teves lliçons subtils del dia a dia han fet que tot aquests anys amb tu siguin els més intensos de la meva vida. Llegar a casa y que estés allí dan sentido a todo esto, sobre todo, sentido del humor. Gracias por hacer de esto, una filosofía. Gracias por enseñarme dónde está lo más importante de este trabajo y de la vida. Aprender a tu lado que no hay ningún problema que justifique un beso menos. Aunque a veces no lo demuestre. Gracias por la comprensión, por la ilusión de compartir nuestra vida y de afrontar nuevos retos. Y sin Bingen no te hubiera conocido... gran metáfora... Bingen,... ¿cuándo viene Binen? El "doctorado" más importante que habré hecho no es este y te lo debo a tí. Es el doctorado de "cómo ser feliz". Me has enseñado a superar todas las trabas que me han ido surgiendo. De hecho seguro que no hubiera llegado hasta aquí sino hubieras pedido una hermanita. Y me has llegado hasta enseñar que no dependa de tí, aunque creo que no obtuve *cum laude*...). Como tampoco lo he conseguido con todo el ejemplo que me has dado tú, Mamá, mi reina. Esta tesis te la dedico. Porque aunque sólo sepas decirme que no entiendes nada, todo esto es gracias a tí. Tu fuerza, tu integridad, tu dedicación, tus comiditas, tus llamadas diarias ... Todo me ha llevado hasta aquí. Como eres bruja, nunca se te ha escapado nada. Y siempre has estado allí para decirme que no sea así. Os quiero.

Content

Chapter 1

Overview

1.1 Introduction

Composing or decomposing? Our objective is to try to reproduce physical phenomena present in our everyday life as successfully as possible. This entails resolving partial differential equations (PDE) in complex geometries with their proper boundary conditions. When we want to solve actual problems with their naturally complex physical phenomena and geometry, we turn to the strategy of composing meshes or decomposing our domain. To obtain an accurate solution to problems of this kind, we must take into account different aspects, such as the geometry of each subdomain, the physics of the different phenomena to be modelled, the numerical methods used to solve each subdomain or the boundary conditions to be imposed in the interfaces between subdomains.

Although Nature is nowadays as complex as it was two centuries ago, the current computing power allows one to study it at much greater depth. For example, we can deal with coupled multi-physic problems or with larger and more complex geometry systems. The reason behind this is the enormous potential of the computational performance. In 1958, IBM's researchers, John Cocke and Daniel Slotnick discussed the use of parallelism in numerical methods for the very first time. Today, this discussion still goes on.

Nowadays, many computational mechanics codes allow to take the advantage of massively parallel supercomputers. Some are purely MPI, some hybrid (MPI/OpenMP), and some others make use of accelerators like GPGPU's. The range of number of cores in which these codes are efficient is still increasing, not only due to a better performance of the connection network but also to

the treatment of the bottlenecks in the computational mechanics codes (e.g. parallel I/O). Within this context of constant increase in performance, applied scientists are becoming increasingly greedy in terms of mesh size, multi-physics coupling and computational domain. Old challenges, such as solving multi-physics problems or complex geometry domains, are now possible, and not only thanks to the growth of computational resources but also to mesh or subdomain composed tools, which allow us to solve this kind of problem more efficiently.
The origin of the subdomain composed tools dates back to two centuries ago, in 1869, when Schwarz solved the Laplace equation in a domain composed of a circumference and a rectangle, which individual solution were already known as it is shown in Figure 1.1. Since then, several methods have been developed to couple independent parts of the computational domain, obtaining a unique and continuous solution.

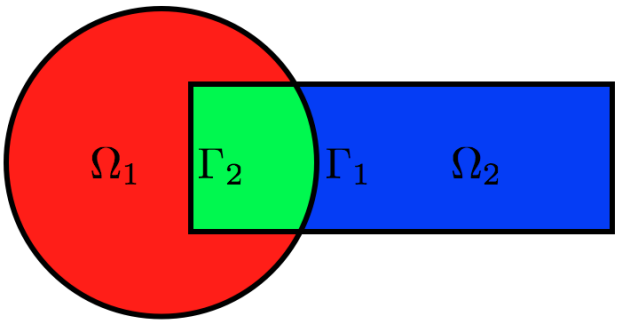


Figure 1.1: Original problem proposed by Schwarz.

More specifically, there are different reasons for the interest in the coupling of meshes. If one is interested in to assemble components obtained from different sources; this is a typical case in industry where components are designed in different departments of the company. Or if one needs to simplify the meshing of complex geometries by dividing it into different meshable pieces. Another reason is to perform local refinement to adapt to local mesh requirements. Or to couple multi-physics problems and/or subdomains in relative motion. And finally if one wants to optimize the relative positions of some components without having to remesh the whole computional domain. All of these situations could appear simultaneously and are in fact quite frequently found in actual problems. The objective of this work is to be able to deal with the majority of them in a more useful or practical way.

From the studies of Schwarz mentioned before, the so-called domain decomposition methods (DDM) appeared in order to allow one to solve computational mechanics problems using parallel algorithms in sequential or parallel machines. The idea in which all these methods are based on is the supposition that a given computational domain can be decomposed into subdomains with or without overlapping. So, the problem is reformulated in terms of each subdomain obtaining new, smaller subproblems which are solved independently although they are coupled by the unknown solution in the interface between them. However the expression "domain decomposition" has different meanings in the PDE's community:

- In parallel computation, it means that the data of the computational problem is distributed among parallel tasks. In this case, the DD can be independent from the numerical method.
- From the point of view of computational mechanics, DD means separating the physical model into regions which can be modelled with different physical phenomena imposing continuity conditions in the interfaces.
- In an algebraic sense, DD divides the solution of a large linear system equation into smaller problems. The solution of each problem is used to design a preconditioner for the algebraic system. In this context, DD only means the manner in which we solve the linear system obtained in the discretization procedure. It is well known that, when we solve computational mechanics problems, other issues appear in the resolution of the linear system if we use very fine meshes in complex domains or with discontinuities in the coefficients and with boundary layers. We have to take into account that in such cases the number of degrees of freedom can be large and parallel computing will be necessary. Direct methods to solve the linear system in parallel do not represent a good strategy because this kind of method requires a lot of communication between the processors. Iterative methods are well suited to solve the linear system but in the context of fine meshes the system could be ill-conditioned and the number of iterations excessive. A good preconditioner for the system reduces the number of iteration and one way to design such a preconditioner lies in the use of DD or multigrid algorithms.

The list comprising the different domain decomposition methods and their associated shortcomings is significantly large and constantly evolving. It is enough to observe the 962,000 academic entries that appear in Google when introducing *domain decomposition methods*. This huge variety of models is due to the different motivations related to these methods. As we have explained before, on the one hand, nowadays we find huge problems to be solved which we have to decompose in order to be able to solve and overcoming the memory problems. On the other hand, we also deal with multi-physics problems.

Finally, we should also take into account certain limitations related to the meshing process, which are perhaps solved if this process is done in parallel or in different parts, obtaining non-conforming meshes. In any case, the final objective shared by all of these techniques is to *divide and conquer*. This is probably the reason why the methods are known as domain decomposition even if what we want to do is couple or compose subproblems of diverse origin. The strategy for composing them depends on conformity of the global mesh, the numerical method, the overlapping, the continuity condition of the solution in the interface or whether the coupling is implicit or iterative. In fact, to the best of the author's knowledge, no complete classification which takes all of these factors into account has been found. The terminology used in the bibliography is sometimes confusing, due to the multiplicity of methods and associated names. For the sake of clarity, we will classify the different strategies into two main classes. On the one hand, there are the so-called Domain Decomposition Methods (DDM); and on the other hand, we introduce, with obvious meaning and after hours of debates, the concept of Domain Composition Methods (DCM)... Figure 1.2 illustrates the difference between both.

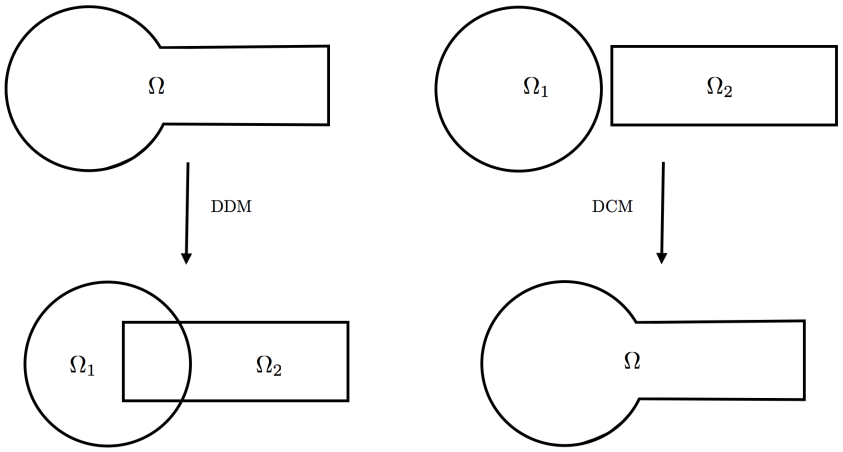


Figure 1.2: Domain Decomposition Method vs Domain Composition Method

The first class, DDM, consists in dividing an existing mesh into submeshes, usually referred to as subdommains. The application of these methods consists mainly in parallelizing the solution strategy or devising algebraic solvers and preconditioners based on local solutions: Primal/Dual Schur, Schwarz, block LU, coupling via transmission conditions and so on. The other class, DCM, has the opposite objective: to join subdomains meshed independently, likely to be non-conforming, and obtain a global solution on this composite mesh.

It should be mentioned that in some applications, certain strategies used in DCM can be used for the same purpose as DDM (multi-block strategies for parallelization), and similar methodologies can be used in both classes (coupling via transmission conditions, see Houzeaux & Codina (2003a)). The role of all DDM and DCM consists in obtaining a global solution on a composite mesh. The DCM category can in turn be divided into two families, depending on whether the coupling is applied directly to the mesh or to the local solutions obtained on the submeshes. These families can be referred to as *mesh-based* and *formulation-based*. The methods proposed in the literature for these two families include:

- Mesh-based (joining, gluing, assembly):
  - mesh conforming;
  - HERMESH (current work).
- Formulation-based (local solution coupling):
  - transmission conditions;
  - constraints imposition;
  - mesh-free interpolations.

The objective of the next section is to present a broad review of this context and in Figure 1.3 showing a sketch of it. This work aims to make a contribu-

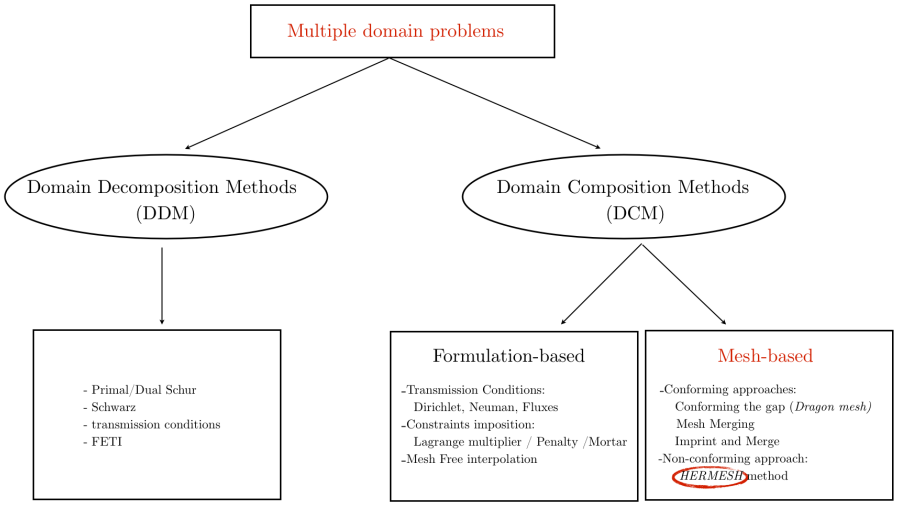


Figure 1.3: Alternatives in the resolution of multiple domain problems

tion in the composite meshes field. The method proposed has been called the

HERMESH method and allows one to couple meshes in an implicit, versatile and automatic way. We regard these properties as a great step forward. We will also show that the method becomes naturally parallel, retaining the parallel performance of the original code. The developments carried out in this thesis have been successfully applied in different contexts. The method has been tested to solve the temperature equation, the Navier-Stokes problems (NS), the Reynolds Average Navier Stokes equations (RANS) coupled with a hyperbolic equation of level set and the solid mechanics problems. We hope that this thesis will serve as a contribution to the state of the art of the area of high performance computing, in mesh coupling problems.

The thesis is organized as follows. In the rest of this chapter, we present our own classification of the existing methods which deal with the coupling of the parts of the domain from different points of view. Roughly, this may be a matter of minimizing the cost of solving a large system of equations, or of issues related to the physics of the problem or the complexity of the geometry. Finally, we present the main objectives of this work.

The second chapter will present the equations of computational mechanics in which we have applied the HERMESH method, showing its versatility. The first will be the advection-diffusion-reaction equation, where we will also explain a new development carried out by the same authors to determine a new expression for stabilization parameter τ in the context of two-scale variational models. Afterwards we will present the incompressible and stabilized Navier-Stokes equations with its respective time discretization as well as the algebraic split strategy used to solve the linearized system. We will also talk about the turbulence phenomenon describing the numerical approaches as well as the level set equation. Finally, there will be a description of the solid mechanics equations.

The third chapter is divided in two parts. Firstly it is dedicated to the presentation of the first steps that were carried out during the development of the methodology proposed in this work and which led us to the technique we now call the HERMESH method. Next, we describe this proposed method as a new strategy for coupling independent meshes. The coupling is carried through the creation of new connectivities based on geometrical criteria that are discussed in this chapter. A detailed description for two- and three-dimensional cases is made. We also talk about smoothing aspects that have been implemented to obtain better solutions of the method. We discuss the parallel performance as well as implementation issues and possible improvements. Finally, some numerical properties are discussed through different examples.

The next two chapters are dedicated to the description of the applications of the HERMESH method. Chapter four describes the Chimera-type problems

solved with our proposed method. The nomenclature and implementation issues are presented in this chapter. The strategy to construct a manifold surface in the hole-cutting process is detailed as well as the coupling in the apparent interfaces created in this hole-cutting step is shown. Finally we prove the numerical properties of HERMESH for this kind of problems through different examples. These properties are: independence of flow direction, mesh convergence and mass conservation. Chapter five presents the other kind of application, the mesh gluing strategy carried out with HERMESH method. We are able to join independent non-matching meshes in an automatic way regardless of their relative positions. This tool is based on different definitions of the interface connectivity information resulting in the four combinations illustrated in the chapter. We prove through different examples some numerical properties as the follows: We obtain error zero when the solution lives in the finite element space. HERMESH can obtain the same solution as one domain problem in a particular case. It is also proved that the mesh convergence is order two.

The sixth chapter presents real applications that are solved with the HERMESH method. The simulation of the free surface flow around a ship hull with the RANS equations together with the Spalart-Allmaras turbulence model and level set equation is run. We have also applied the method to the simulation of a neuron within a solid mechanics framework. The HERMESH method is a very useful strategy for simulating the flow in a wind farm to optimize the position of the turbines since for each configuration of the computational domain we do not have to remesh. Another advantage is related to being able to capture the wake generated for each turbine. It is possible with a fine patch mesh which contains the disk used to simulate the turbine. The rest of the computational domain could be meshed with a coarse and structured background mesh, saving a lot of computational resources. The simulation of the wind farm entails solving the Navier-Stokes equations coupled with a $k - \varepsilon$, and the boundary layer is solved with a wall law. Finally, we present two biomechanical applications solved with the HERMESH method. One corresponds to the large and small airways simulation. The other one is the flow passing through a bypass in a stenosed artery. So both applications belong to the context of the incompressible Navier-Stokes equations.

The last chapter is devoted to some conclusions and future work.

1.2 State of the art in multiple domains problems

We are going to make a broad review of all of these methods.

1.2.1 Origins

As noted before, in 1869 Schwarz proposed the idea of dividing the domain to solve the Laplace equation. His idea was the origin of the family of methods called DDM. With this technique the computational domain Ω where the boundary value problem is set is subdivided into two or more subdomains on which discretized problems of smaller dimension are to be solved. There are two ways of subdividing the computational domain: overlapping and disjoint subdomains.

Overlapping: Schwarz methods. In particular, the way of dividing the Schwarz domain was with overlapping subdomains. His idea was to solve the Laplace equation in a domain composed of simple domains, as is illustrated in Figure 1.1. In particular, he proposed solving the PDE in a circle with boundary conditions taken from the interior square and solving the PDE in the square with boundary conditions taken from the circle interior. And iterating.

Let $\Omega = \Omega_1 \cup \Omega_2$ with $\Omega_1 \cap \Omega_2 \neq \emptyset$ and solve the next problem:

$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega & (1.1)\\ u &= 0 \quad \text{on} \quad \partial\Omega & (1.2) \end{aligned}$$

such that,

$$\begin{cases} -\Delta(u_1^{n+1}) = f & \text{in} \quad \Omega_1 \\ u_1^{n+1} = 0 & \text{on} \quad \partial\Omega_1 \cap \partial\Omega \\ u_1^{n+1} = u_2^n & \text{on} \quad \Gamma_1 \end{cases}$$

$$\begin{cases} -\Delta(u_2^{n+1}) = f & \text{in} \quad \Omega_2 \\ u_2^{n+1} = 0 & \text{on} \quad \partial\Omega_2 \cap \partial\Omega \\ u_2^{n+1} = u_1^{n+1} & \text{on} \quad \Gamma_2 \end{cases}$$

The part of the subdomain boundary Ω_i, $\partial\Omega_i$, which is not part of the global physical boundary, $\partial\Omega$, is referred to as the *artificial internal boundary*, Γ_i. This problem was called the *alternating Schwarz algorithm*. The problem in Ω_1 has to be solved before the problem in Ω_2, so the algorithm is sequential by nature.

One century later, Lions (1988) introduced the Parallel Schwarz method for the purpose of parallel computing. It consists of the following formulation, based on the problem presented above:

$$\begin{cases} -\Delta(u_1^{n+1}) = f & \text{in} \quad \Omega_1 \\ u_1^{n+1} = 0 & \text{on} \quad \partial\Omega_1 \cap \partial\Omega \\ u_1^{n+1} = u_2^n & \text{on} \quad \Gamma_1 \end{cases}$$

$$\begin{cases} -\Delta(u_2^{n+1}) &= f & \text{in} & \Omega_2 \\ u_2^{n+1} &= 0 & \text{on} & \partial\Omega_2 \cap \partial\Omega \\ u_2^{n+1} &= u_1^n & \text{on} & \Gamma_2 \end{cases}$$

It inherently becomes parallel. The differences stems from the way of updating the artificial Dirichlet boundary condition on Γ_i. These methods can be generalized to an arbitrary number of subdomains.
Both methods presented in a continuous sense were discretized to obtain computational tools. The *Multiplicative Schwarz Method* (MSM) is the algebraic equivalent to the original alternating method and the *Additive Schwarz Method* (ASM), developed by Dryja and Widlund in 1988, is the algebraic equivalent to the Parallel Schwarz method. At the same time, as shown in Figure 1.4, there exists a system solver analogue for both methods. MSM can be seen as block Gauss-Seidel and ASM as block Jacobi; this algorithm can be seen as an iterative solver.

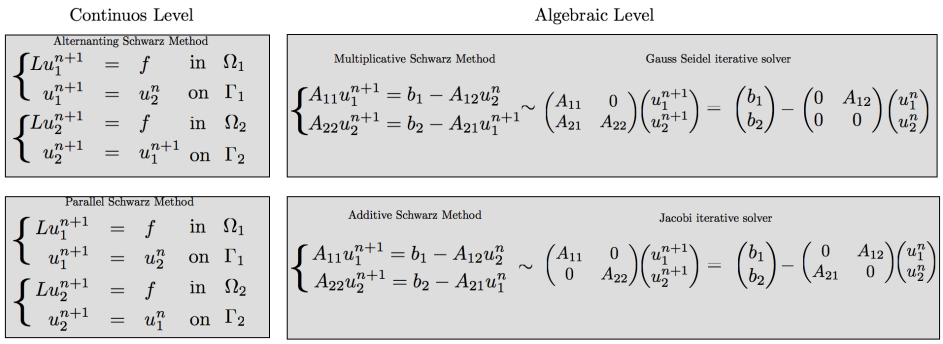


Figure 1.4: Continuous and algebraic forms of Schwarz methods.

In both cases we have two elliptic boundary value problems with a Dirichlet condition, in Ω_1 and Ω_2 and two sequences $u_1^{(n+1)}$, $u_2^{(n+1)}$ that must converge to the restrictions of the solution u of the original problem. The rate of the convergence depends on the overlapping region. Roughly speaking, overlapping DDM operate by means of an iterative procedure, where the PDE is repeatedly solved within every subdomain. Each subdomain with the artificial internal boundary condition is provided by its neighbouring subdomains The convergence of the solution on these internal boundaries ensures the convergence of the solution in the whole domain. But the problem is that this convergence is deteriorated when the number of subdomains increases. This problem can be relaxed with the so called coarse grid corrections, introduced in Bramble, Pasciak, & Schatz (1986), which incorporate a more rapid global information propagation mechanism.
As is well known, when real problems, $Lu = f$ in Ω with $u = g$ in $\partial\Omega$, are solved with a numerical method such as the finite difference, finite volume or

finite element method, the resulting system of linear equations $Au = b$ is very large and so we solve it in an iterative way. A stationary iterative method could be applied and is formulated in the following way:

$$u^{n+1} = u^n + P(b - Au^n) \tag{1.3}$$

where P represents a preconditioner. If the domain Ω is divided in p overlapping subdomains, Ω_i an equivalent algebraic residual formulation of the overlapping DDM can be done. First of all a restriction operator, R_i, has to be designed such that. $R_i : \Omega \longrightarrow \Omega_i$ and is defined as:

$$(R_i^t)_{g,l} = \begin{cases} 1, & \text{if g is the global number of local node l in}\Omega_i; \\ 0, & \text{otherwise.} \end{cases} \tag{1.4}$$

That is, this matrices are formed by the identity on the corresponding subdomains and extended by zeros. Their action restricts a vector in the global discrete domain to a vector in the local discrete domain by choosing the entries corresponding to the interior nodes of the subdomain. Their transpose is an extension matrix that prolongs a vector in the discrete domain to one in the global domain by adding zeros for the nodal values which are not in the considered subdomain. A prolongation operator is also necessary and corresponds to R_i^t. With this definition, the restriction of the matrix A in each subdomain is computed by:

$$A_i = R_i^t A R_i, \tag{1.5}$$

and the formulation of the overlapping ASM is computed as follows:

$$u^{n+1} = u^n + \sum_{i=1}^{p} R_i^T A_i^{-1} R_i (b - Au^n), \tag{1.6}$$

This residual form of the algorithm helps to state that the ASM could be seen as a preconditioner in a Richardson iteration, with the preconditioner written as follows:

$$P = \sum_{i=1}^{p} R_i^T A_i^{-1} R_i \tag{1.7}$$

For the simple case of two subdomains, considered in Figure 1.4; the expression of the Richardson iteration is done by:

$$\begin{pmatrix} u_1^n \\ u_2^n \end{pmatrix} = \begin{pmatrix} u_1^{n-1} \\ u_2^{n-1} \end{pmatrix} + (R_1^t A_{11}^{-1} R_1 + R_2^t A_{22}^{-1} R_2)(b - Au^{n-1})$$

the restriction operators are:

$$\begin{array}{rcl} R_1 & = & [I \quad 0] \\ R_2 & = & [0 \quad I] \end{array} \tag{1.8}$$

such that applied to the global matrix A:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

we obtain the same expression given in Figure 1.4:

$$(R_1^t A_{11}^{-1} R_1 + R_2^t A_{22}^{-1} R_2) = \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & A_{22}^{-1} \end{pmatrix} = \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{pmatrix}$$

Incorporating the idea mentioned before about the coarse grid correction, the preconditioner will be written as follows:

$$\sum_{i=0}^{P} R_i^T A_i^{-1} R_i \tag{1.9}$$

where A_0^{-1} represents the solver for the global coarse grid problem. Strategies of this kind could be seen as a variant of the two-level multigrid methods, originally introduced by Bank & Dupont (1980). The generalization to many levels was done by Dryja, Sarkis, & Widlund (1996). The first studies related to the development of preconditioners based on DDM strategies are found in Golub (1983); Chan (1987). A key point in this issue is to achieve the optimality property when the number of subdomains grows. Dryja and Widlund developed a special additive Schwarz method, see Dryja & Widlund (1994). In this paper the authors demonstrate that the condition number of the preconditioned operator for the two-level additive Schwarz algorithm is bounded from above by $C(1 + (H/\delta))$ where H is the diameter of the subregion and δ is the overlap between subdomains and the constant C is independent from H, h and δ. But in this case, the size of the overlap has to be generous. Other numerical results, Gropp & Smith (1994), demonstrated that this is not a restriction so, with a minimal overlap, these methods perform quite well. Some years later, a variant of the additive Schwarz algorithm was developed by Cai and Sharkis and called Resctrictive Additive Schwarz (RAS) algorithm. The original paper was Cai & Sarkis (1999) and was further analyzed in Sarkis & Koobus (2000). This variant converges faster than the classical Additive Schwarz and this is due to RAS algorithm is convergent everywhere whereas ASM is not convergent in the overlap.

Although these Schwarz methods had been developed originally for elliptic problems, there are a lot of studies to apply them to parabolic problems. See Lions (1988) where the dependence of the convergence rate on the time step and mesh size is discussed. Lions analyses the variational form interpretation of the problem and mentions some references of previous studies about the convergence of the solution of classical boundary value problem for harmonic functions in an iterative way. In Cai (1994) the time is discretized to obtain

a sequence of steady problems to which the domain decomposition methods are applied. On the other hand, in Gander (2006) the authors formulate algorithms directly for the original problem without discretization in time, which means that time dependent subproblems are solved. In Gerardo-Giorda, Nobile, & Vergara (2010) a convergence analysis of Robin/Robin strategy for the fluid-structure problem is presented as well as optimal values of parameters in the Robin transmission conditions are determined.

To finalise this brief review of the Schwarz methods, it is worth noting some remarks. The additive methods converge more slowly than multiplicative ones and are more restrictive in terms of the smoothness required by subdomains although ASM presents better properties in parallel computations. In order to combine the best properties of both methods in Cai (1993) a hybrid method is proposed. But there is no general and universal method because the multiple proposed combinations depend on the problem to be solved. Both methods present the drawback of needing overlapping subdomains to converge since only the continuity of the unknown is imposed, and not on the flux of the unknown (called Neumann condition). When there is no overlap, the convergence is not possible. In Lions (1989) a new version of the Schwarz algorithm with Robin conditions (which means a linear combination of Dirichlet and Neumann condition) in the interface is presented. Methods of this kind are referenced as optimized Schwarz methods, do not depend on the overlapping, and the converge is faster. In Houzeaux (2003) an algorithm of domain decomposition in overlapping subdomains based on the imposition of the Dirichlet/Robin condition is developed.

Non-overlapping. Also formulated as an iterative strategy, the idea is to introduce proper coupling transmission conditions in the interface between subdomains. Such methods are known as **Iteration-by-subdomains** and were first proposed by P.L. Lions, see Lions (1989). Non-overlapping DDM necessarily use two different transmission conditions on the interface, in such a way that both the continuity of the unknown and its first derivatives are achieved on the interface (for ADR equations). See Houzeaux (2003).
Many possibilities could be formulated as we can see in the following formulation. Let $\Omega = \Omega_1 \cup \Omega_2$ with $\Omega_1 \cap \Omega_2 = \emptyset$ and Γ the interface between subdomains. Then the problem is formulated as:

$$\begin{cases} -\Delta(u_1^{n+1}) = f & \text{in} \quad \Omega_1 \\ u_1^{n+1} = 0 & \text{on} \quad \partial\Omega_1 \cap \partial\Omega \\ \alpha_1 u_1^{n+1} + \beta_1 \dfrac{\partial u_1^{n+1}}{\partial n_1} = \alpha_1 u_2^n + \beta_1 \frac{\partial u_2^{n+1}}{\partial n_1} & \text{on} \quad \Gamma \end{cases} \tag{1.10}$$

$$\begin{cases} -\Delta(u_2^{n+1}) & = f & \text{in} & \Omega_2 \\ u_2^{n+1} & = 0 & \text{on} & \partial\Omega_2 \cap \partial\Omega \\ \alpha_2 u_2^{n+1} + \beta_2 \frac{\partial u_2^{n'}}{\partial n_2} & = \alpha_2 u_1^n + \beta_2 \frac{\partial u_1^{n'}}{\partial n_2} & \text{on} & \Gamma \end{cases} \tag{1.11}$$

with n_i being the outward unit vector normal to the interface Γ and n' could be the iteration $n+1$ for the parallel version of the algorithm or n for the sequential option. Depending on the kind of transmission conditions, $(1.10)_3$ and (1.11_3) different strategy types appear:

- Dirichlet/Neuman(D/N): with $\beta_1 = \alpha_2 = 0$;
- Robin/Robin(R/R): with $\alpha_1, \alpha_2, \beta_1, \beta_2 \neq 0$;
- Robin/Neuman(R/N): with $\alpha_2 = 0$.

One of the first works devoted to developing variations of the Schwarz methods is Bjorstad & Widlund (1986). In particular, the authors analyse a D/N method. Other similar studies are carried out in Marini & Quarteroni (1988). Also, Widlund (1988) shows an extension of this method considering global coarse solvers. In Funaro, Quarteroni, & Zanolli (1988) the authors propose a relaxation parameter for the iterative process. The first studies in the N/N methods correspond to the reference Bourgat, Glowinski, Tallec, Vidrascu et al. (1988). Other works are Tallec (1993); Berselli & Saleri (2000); Achdou, Tallec, Nataf, & Vidrascu (2000) among others. In Carlenzoli & Quarteroni (1995) the adaptive D/N was introduced, meaning that a Dirichlet transmission condition at inflow and Neumann transmission condition at outflow are imposed as well as adaptive R/N method. These methods are reviewed in Gastaldi, Gastaldi, & Quarteroni (1996). Robin/Robin combination is studied in Lions (1989); Nataf (1995); Lube, Otto, & Muller (1998);

These methods could be transformed to other iterative processes solving the problem on the interface. The Steklov-Poincaré methods and the Schur complement methods involve the construction of lower dimensional systems for degrees of freedom defined on the interfaces of subdomains; the former is at the differential level, and the latter at the algebraic level. As in the case of Schwarz methods, any combination of iteration-by-subdomains methods described before can be reinterpreted as a preconditioned Richardson method, but in this case, for the solution of the Steklov-Poincaré interface equation. For example in Toselli & Widlund (2005) or Marini & Quarteroni (1989) such an equivalent is treated in D/N algorithm. For the N/N case, this issue is treated in Tallec (1993). For a detailed study of these issues see Quarteroni & Valli (1999). Mandel developed the so-called *Balancing Domain Decomposition* methods (BDD, Mandel (1993)) in which the coarse problem is introduced. The way to construct an efficient preconditioner for the resulting interface operators is a core problem in the study of non-overlapping DDM. The construction

of this preconditioner has been investigated from various strategies and to various models in the literature. This kind of preconditioner consists of two ingredients: one coarse solver and a set of local solvers. So one has to consider *multi-level* methods to ensure faster communication between subdomains. The idea behind these techniques is to combine a smoothing step in which the high error frequencies are attenuated and a correction step at the coarse level. A review of algebraic multilevel methods can be found in Axelsson (2003). The difference between algebraic multigrid, also called multi-level technique, and geometric multigrid is that the former method does not take into account any information about the mesh associated with the discretization. The second class of methods uses the finest mesh to construct the coarse space to speed up convergence. A good review of these methods applied to CFD techniques is found in Wesseling & Oosterlee (2001). A general review about algebraic multigrid is seen in Stüben (2001) and the references therein. A general survey of preconditioning techniques for large linear systems can be found in Benzi (2002).

All of these methods presented have been traditionally called domain decompositions methods. With the continuing growth of computational resources, these methods have constitued the paradigm for evolving parallel computational techniques. In fact, the origin of both, DDM and DCM is the same: Schwarz methods. This thesis adresses the problem consisting of coupling independent meshes by means of DCM. This strategy is described in detail in the following section.

1.2.2 Domain Composition Methods

In this work we are going to focus on the DCM class. This technique is useful for different reasons:

- To assemble different components obtained from different sources, this case being typical in industries in which components are designed in different departments of the company.
- To simplify the meshing of complex geometries by dividing it into different meshable pieces.
- To perform local refinement to adapt to local mesh requirements.
- To couple multi-physics problems.
- To couple subdomains in relative motion without having to remesh the whole computational domain (typically the Chimera method).
- To optimize the relative positions of some components without having to remesh the whole computational domain, tipically in the Chimera problems.

Any of these reasons leads in general to non-matching meshes. The work presented in this thesis as well as the rest of this section will be focused on this framework. As we have shown before, the DCM class has been divided into two families, depending on whether the coupling is applied directly to the mesh or to the local solutions obtained on the submeshes. These families have been referred to as *mesh-based* and *formulation-based* as is shown in the scheme of Figure 1.2 and will be explained in the rest of the section.

Formulation-Based

We have considered three different methodologies belonging to this first family:

- transmission conditions coupling;
- constraint imposition methods;
- mesh-free interpolation.

This section is devoted to dealing with them.

Transmission conditions. This family corresponds to the methods explained in section 1.2.1. It is worth to mention that in the DCM context, the continuity could be imposed through non-matching meshes. The coupling is achieved via transmission conditions (in the finite element context) or by imposing the continuity of fluxes (in the finite volume context). In a finite element context, the transmission conditions make it possible to obtain continuous variables and continuous fluxes across the interfaces. These conditions can be of Dirichlet or Neumann/Robin type, imposing continuity of the solution and its flux, respectively.
The main drawbacks of iteration-by-subdomains methods are listed as follows:

- As pointed out in Houzeaux & Codina (2003b) at the continuous level, one can show the equivalence between the two-domain and one-domain formulation for overlapping subdomains. The same equivalence is demonstrated for disjoint subdomains in Quarteroni & Valli (1999). However, the equivalence is no longer true at the discrete level.
- They introduce an additional iterative loop.
- The convergence depends on the local characteristics of the equation.
- The Neuman or Robin condition depends on the type of equation.

In fact, as we noted before, this kind of methods has been applied mainly to the parallelization strategies.

Constraint Impositions. This family of methods couples the solutions obtained on disjoint subdomains by constraining the governing equations with certain continuity conditions. These continuity constraints can be imposed via Lagrange multipliers or by using penalty methods.
Lagrange multiplier methods are functions defined on the interfaces between adjacent subdomains for imposing continuity in the primal unknowns. The coupling is explicitly in the variational form, in the continuous or discrete sense. The first works found in this issue are Dihn, Glowinski, & Périaux (1984) or Dorr (1989). Although these methods do not imply the iterative procedure, an increasing number of degrees of freedom is necessary in order to add the new unknowns corresponding to Lagrange multipliers. The Lagrange spaces have to be properly chosen if we want to obtain a well-posed linear system or we have to apply some stabilization technique, see Brezzi, Franca, Marini, & Russo (1997b). Farhat and Roux introduced another variant of these methods, the so-called Finite Element Tearing and Interconnecting (FETI) methods, Farhat & Roux (1991). In particular, these methods were developed for elasticity problems in conforming subdomains although they have also been extended to advection-diffusion-reaction problems, see Toselli (2001), and in non-conforming subdomains, see Stefanica & Klawonn (1999). The primal or original unknowns are eliminated by solving a local Neuman subproblem with the result of obtaining the equation for the Lagrange multipliers or dual unknowns. A strategy inspired in the elasticity problems lies in the 3-field method, by Marini & Brezzi (1994). This was introduced in the domain decomposition framework by Brezzi and Marini in 1994 and offers the possibility to discretize different subdomains with different formulations. The original problem is reformulated introducing one Lagrange multipliers in each subdomain. This method enables one to independently solve two local problems corresponding to each subdomain so one can work with different meshes or different discretizations. In order to by-pass the inf-sup conditions of the standard three-field formulation, in Rapin & Lube (2004) a stabilized three-field formulation is presented. The authors use finite elements with SUPG stabilization in the interior of the subdomain so that almost arbitrary discrete function spaces could be used in each subdomain.

These methods are also useful in an algebraic sense and could also be useful as a solver strategy. In the reference Rivera, Heniche, Glowinski, & Tanguy (2010) we can find a strategy to parallelize an ILU-type iterative finite element solver for the Stokes problem using the Lagrange multiplier to communicate between the processors, minimizing the number of communications. The solutions of the Lagrange multipliers represent the jump in normal derivatives through the interfaces. In that work, some penalty parameters are introduced in order to solve the problem of the existence of zeros in the diagonal of the resulting matrix.

We want to devote special attention to the non-matching meshes, since

our method can couple this kind of meshes. As it will be emphasized during this thesis, there are a lot of reasons to treat non-matching meshes. The idea presented before in the context of three-field formulation, consisting in allowing the use of different orders of interpolation in the different subdomains, is not the only motive to deal with non-matching meshes. It is also useful when PDE's with heterogeneous media or Adaptive Mesh Refinement (AMR) are solved. When complex geometries have to be meshed, one of the most useful strategies is to divide the geometry into independent parts obtaining a non-matching global mesh. In contact problems, the necessity to tie two independent and non-matching meshes also appears. Many examples could be mentioned in this context. As a first result of coupling different discretizations we find the work of Bernardi & Maday (1990). The authors propose and compare two types of couplings with the Poisson equation. Later, the so-called mortar methods, Belgacem & Maday (1994), appeared to treat the coupling in non-matching grids, developed originally to disjoint subdomains. Mortar methods impose the continuity in the interfaces in a weak form using the Lagrange Multipliers space which is orthogonal to the jump in the unknown interface. These methods couple a mesh discretized with spectral elements with another mesh discretized with finite elements. The earliest works can be found Belgacem & Maday (1997); Belgacem (1999). In the following references the mortar method is extended to overlapping subdomains, Cai, Dryja, & Sarkis (1999); Achdou (2002); Kim & Widlund (2006). In Becker, Hansbo, & Stenberg (2003) a mortar method based on the classical Nitsche method is developed. This last approach solved a Dirichlet problem without introducing boundary conditions in the finite element space definition. In Hansbo, Hansbo, & Larson (2003); Hansbo (2005) a consistent finite element method for overlapping meshes based on Nitsche's method is analysed. In Massing, Larson, & Logg (2013) the geometric computations which are necessary to assemble the discrete system associated to the Nitcshe methods, and other methods like contact mechanics or extended finite elements methods, among others, are discussed. In Bernardi, Maday, & Rapetti (2005) the authors describe the characteristics of mortar spectral element and mortar finite element methods. The coupling is done optimally in the sense that the error is bounded by the sum of the approximation errors in different subdomains to be coupled. But one of the main drawback of these methods lies in the fact that the resulting matrix becomes non-positively definite. An alternative strategy to mortar methods is found in Boillat (2003). The authors couple independent meshes discretized with linear finite elements through the use of a penalty technique. The scheme is compared to the Nitsche method and the penalty parameter, with which a quasi-optimal convergence is obtained, is proportional to the mesh size.
In the adaptive mesh refinement (AMR) context mentioned before the conformity of the mesh is lost, which leads to the so-called *hanging nodes*. To solve these problems the Lagrange Multipliers method has been widely used. See,

for example, Berger & Colella (1989); Bernardi & Maday (2000); Trangenstein & Kim (2004); Fard, Hulsen, Meijer, Famili, & Anderson (2012). In Glowinski, He, Lozinski, Rappaz, & Wagner (2005) we can find the resolution of this problem with a Schwarz algorithm presented in 1.2.

Mesh-free methods. Let us mention other kinds of methods that belong to the formulated-based family and correspond to the use of mesh-free techniques to locally connect non-conforming meshes. The idea is to pass the information from one side to another, using the mesh-free framework in the gap between the subdomains. This strategy to couple both kinds of discretization methods is already studied in the work of Belytschko, Organ, & Krongauz (1995), where the advantages of each of the methods are used, meaning that the authors couple finite elements near the Dirichlet boundaries and element-free Galerkin in the interior of the domain. Another application of this strategy is found in the refinement process, as we can see in the work of Liu, Uras, & Chen (1997). The reference Huerta & Mendez (2000) presents a unified formulation of a mixed interpolation general to any spatial order and this formulation is applied for coupling and enrichment purposes. A general presentation of these methods is in Belytschko, Krongauz, Organ, Fleming, & Krysl (1996) and an excellent review is found in Li & Liu (2002). There are different strategies to interpolate the information between subdomains when mesh-free methods are used. In Kim (2002) we find the moving-least-square (MLS) interpolation (Dolbow & Belytschko (1998)) to construct shape functions used in Interface Element Method (IEM). The same strategy for hexahedral meshes in three dimensions with curved surfaces is explained by the same author in Kim (2008). The continuity and compatibility conditions are satisfied at the non-matching interface and no additional degrees of freedom are generated. The finite element meshes to be joined could present gaps and penetrations on the curved interfaces because the interface elements eliminate these irregularities since they divide the finite elements into several polygons. But rational-type shape functions coming from MLS-approximations present a difficulty in numerical integration. In Tian & Yagawa (2007) the authors couple non-matching meshes for the elasticity problem with three different interpolations: compact support radial basis function (CS-RBF), radial point interpolation method (RPIM) and moving Kriging interpolation method (MKIM). The basic idea of the coupling via mesh-free methods is to define in the interface between finite element meshes a new mesh-free type of mesh with new elements called meshless gluing elements. In this new mesh the integration area does not coincide with the interpolation area. A new interface also appears between the meshless-type mesh and original mesh, but there is conformity between the nodes, meaning that the interpolation is continuous C^0 , so the continuity is set automatically. A general review of the existing methods in this kind of hybrid formulations is the following reference:

Rabczuk, Xiao, & Sauer (2006).
Mesh-free methods to couple non-conforming meshes presents some advantages over the methods based on Lagrange multipliers. On the one hand, mesh-free methods are easier to implement for any dimension. On the other hand, in an algebraic sense, the resulting matrix is positive definite and without adding more degrees of freedom. The main drawback of this strategy is determining the influence domain of the elements of the interface since the shape functions do not maintain the same properties of finite element shape functions. To overcome this issue in Cho, Jun, Im, & Kim (2005), an alternative method is presented in such a way that a correspondence is set between a master element and one element of the interface which naturally determines the influence domain.
In the AMR context, one also finds some applications of this technique, see Lim, Im, & Cho (2007), where the authors illustrate the application for automatic mesh refinement around a stress concentration zone in a T-shape structure in three-dimensions. It is worth mentioning that, in this context, another hybrid combination could be found in the form of Continuous Galerkin with Discontinuous Galerkin Method; for instance, in Badia & Baiges (2013) for incompressible flows. A continuous Galerkin variational multiscale formulation is combined with an equal-order symmetric interior penalty discontinuous Galerkin with upwind for the convective term.

Mesh-based

The mesh-based methods belonging to this family only take geometrical aspects into consideration to connect independent meshes. These methods are usually given a general term: meshing assembly models. This type of methods are divided into two main groups:

- Conforming approaches:
  - creating a gap and meshing it with a conforming mesh;
  - mesh merging;
  - imprinting and merging.
- Non-conforming approach: HERMESH method.

All of these techniques represent an alternative to the above-mentioned strategies of "formulated based" and consist in modifying certain aspects of the meshes to be coupled in order to obtain conforming or quasi-conforming meshes. Let us analyse both approaches.

Conforming approaches. The first group obtains one conforming mesh from two independent meshes. There are serveral techniques to do this. Among them, we find the so-called DRAGON (Direct Replacement of Arbtrary Grid Overlapping by Non-strucutred grid) meshes, see Liou & Kao (1994). The meshes are originally formed from two structured meshes joined together by one non-structured mesh. The non-structured mesh is formed a posteriori from a hole-cutting process. The hole is filled in with this non-structured mesh which connects both structured meshes in a conforming manner.
The second strategy corresponds to the mesh merging technique and it is found in the work of Staten, Shepherd, Ledoux, & Shimada (2010), for hexahedra meshes. The author called the technique Mesh Matching. It locally modifies the topology of the hexahedra elements on one or both sides of the interface surfaces, allowing the meshes to be merged into a conforming mesh across the interface. In the finite volume framework, we can mention the code developed by EDF R&D, *Code_Saturne*, which uses non-conforming meshes. Adjacent boundaries of non-conforming meshes are split into their intersecting subsets, resulting in their forming a conforming mesh of polyhedra with an arbitrary number of faces per cell, although they exclude polygons with holes. The idea is to build new faces corresponding to the intersections of the initial faces to be joined. The terminology used by the authors for this process is *conforming joining*.
We also find a mesh merging technique to create a conforming mesh coming from two independent meshes and consisting in merging the meshes with the intersection between them. This kind of algorithm performs what are, in a way, Boolean operations between groups of surface meshes. In Lo & Wang (2004) we find an algorithm based on tracing the neighbours of intersecting triangles (TNOIT) to determine the intersection lines. Once this intersection is done, the nodes on the intersection lines are repositioned, elements cut by the intersection line are removed and the gap is meshed by the proper connection of nodes between the intersection line and the surface boundary. A similar technique is found in Cebral, Löhner, Choyke, & Yim (2001) to join a carotid artery with stenosis or to merge surface models representing the internal and external carotid arteries of a normal subject.
Finally, we mention the geometric operation of imprinting technique, although this is done at the geometry level and not at the mesh level. The idea is to compute the intersection graph between two objects which create a coincident topology where both objects intersect. After the imprinting operation, mesh mirror technique or merge strategy is applied in order to obtain a conforming mesh between two objects. These techniques are explained in White, Saigal, & Owen (2004) or Clark, Hanks, & Ernst (2008) where non-manifold interfaces between volumes are created from an assembly model for conformal meshing. This technique, like many others, is used in the CUBIT project (Blacker, Bohnhoff, & Edwards (1994)). It is a full-featured software toolkit for the robust

generation of two- and three-dimensional finite element meshes and geometry preparation. Its main goal is to reduce the time for generating meshes, particularly large hex meshes of complicated, interlocking assemblies. One of the functions of this code is to import two independent and non-conforming meshes and modifying the meshes locally to make them conforming. The result is a single mesh that is stitched together at the locally modified region.

Non-conforming approach. The other class of strategies based on mesh modifications is the method presented in this work: the so-called HERMESH method. It creates new connectivities between independent meshes with the existing nodes. The method has been explained in some references as Eguzkitza, Houzeaux, Aubry, & Vázquez (2013a); Houzeaux, Eguzkitza, Aubry, Owen, & Vázquez (2013); Eguzkitza, Houzeaux, Calmet, Vázquez, Soni, Aliabadi, Bates, & Doorly (2013b) and it will be described in detail in this manuscript.

1.3 Applications of Domain Composition Methods

The Domain Composition Methods presented before have been developed to be applied in two kinds of frameworks. One corresponds to the *Mesh Gluing* problems where different components will be composed to obtain a global solution. The other context corresponds to the so-called *Chimera* problems, which also compose independent meshes but add the particularity of one overset mesh inside the other with a corresponding hole-cutting procedure. Both scenarios will be described in the following sections.

1.3.1 Mesh gluing

The idea of this kind of application is to compose independent meshes also referred to as dissimilar meshes. The issue of connecting them is one of the major problems in FEM. The meshes could be non-matching and the discrete interface between the meshes may present gaps or overlaps. Coupling non-conforming meshes help us to piece together the parts that are modelled independently, without sacrificing accuracy or efficiency, such as wing and fuselage structures that may have been modelled by different analysts in different groups or organizations. Moreover, one can also impose selective refinement only on those components where it is required. Hence, in a variety of industrial applications, it is important to employ an efficient method that uses the existing meshes to solve the global system.The origins were in aerospace engineering and was also extended to offshore and shipbuilding industries or even in the electronic field. Another advantage of this technique is found in problems with repeated structural components like blades in an wind turbine.
We find the application of mesh gluing in several softwares under different

names. In the NASA Langley Research Center, a method has been developed for analysing structures composed of two or more independent substrucutures, based on a hybrid variational formulation with Lagrange multipliers and interfaces elements, Aminpour, Ransom, & McCleary (1995). The software developed by them is *MSC Patran/Nastran*, see Schiermeier, Housner, Ransom, Aminpour, & Stroud (1996), which offers the possibility to connect dissimilar meshes and they refer to this option as *glue*. They apply the technique mostly for permitting a high level of refinement in the local region and a coarser level of refinement in the global region, that is, as a local refinement technique.
The software *Abaqus* also offers the possibilty of mesh gluing, known by the developers as *tie*, via constraint equations and using a master-slave formulation. The constraint prevents slave nodes from separating or sliding in relation to the master surface. See this manual for further details: ABAQUS (2014). In both softwares, the union between meshes could be for a shell and solid or two solids. In other commercial softwares like Altair (2014) only the contact between surfaces is valid.
Mesh gluing is also necessary to solve multiphysic problems. As an example, in fluid-structure interaction (FSI) computations, a natural decomposition is carried out by using non-overlapping meshes for the flow and the structure, also referred to as *partitioned coupling*. See Piperno, Farhat, & Larrouturou (1995); Farhat, Lesoinne, & Tallec (1998); Dureisseix (2008). In FSI realistic applications, the fluid and structure meshes are non-matching either because they have been designed by different analysts or because the fluid and structure problems have different resolution requirements, as we have explained before. In the following reference, Felippa, Park, & Farhat (2001) we find an interesting review of the use of partitioned strategies in coupled dynamical systems and a particular summary of the terminology related to this issue. Among many other definitions, the authors describe two kinds of partitioning which we want to mention:

> *Partitioning may be* ***algebraic*** *or* ***differential****. In algebraic partitioning the complete coupled system is spatially discretized first, and then decomposed. In differential partitioning the decomposition is done first and each field then discretized separately.*
> *Differential partitioning often leads to non-matched meshes, as a typical of fluid-structure interaction and handles those naturally. Algebraic partitioning was originally developed for matched meshes and substructuring, but recent work has aimed at simplifying the treatment of non-matched meshes through frames*

As the authors note, since the problem that we are considering can be studied from many angles, the terminology is far from standard. We want to refer to the general problem of coupling systems with independent meshes, whether it be coupling different structural domains, multiphysical problems or even contact problems, such as *mesh gluing*.

The Arlequin method is another tool for the multimodel and multiscale analysis of complex mechanical problems, see Dhia & Rateau (2005). This technique locally allows refinement to link structure models (also called *substructuring* or *external junction* by the authors) or to introduce an essential local modification in the models themselves (which is referred to by the authors as *internal junction*). The coupling operator is based on the Lagrange multiplier field belonging to the dual space of admissible displacement fields restricted to the gluing zone.

Different ways can be found to transfer data between non-matching meshes in FSI and could be divided in a direct way or dual way. The former is formed by techniques such as nearest neighbour interpolation, projection methods and methods based on interpolation by splines. A good reference where these methods are compared is Boer, Zuijlen, & Bijl (2007). A dual way to impose boundary conditions is by Lagrange multipliers, explained before. Another example of multiphysical problem is the work of Dureisseix & Bavestrello (2006), for thermo-viscoelasticity coupled problems where an extension of mortar technique is applied.

Techniques in contact computational finite element analysis involve tying or gluing several non-matching computational domains. In this problem, interfaces are physical ones and present the boundary between two components and could present gaps or overlapping between them. Many of the mortar methods with non-matching grids applied to this problems fail the so-called patch test, meaning that they do not exactly recover any globally linear solution of the governing equations (see Dohrmann, Key, & Heinstein (2000) for further discussion of this issue). In Parks, Romero, & Bochev (2007) this test is guaranteed using a novel Lagrange multipliers technique joining finite element models on two independently meshed subdomains in two dimensions that share a curved interface solving the Poisson equation. Another example of mortar method with curved surfaces is Flemisch, Melenk, & Wohlmuth (2005) or Puso (2004) in three dimensional problems for solid mechanics.

We find also composing meshes in aeroacoustic computation. This is the case of the work of Lee, Vouvakis, & Lee (2005) for modelling large finite antenna arrays, or the work of Peers, Zhang, & Kim (2010) in high-order finite difference schemes for multiblock computational aeroacoustic in complex geometries.

To unite the rotor and stator independent meshes to compose an electric machine we find the work of Tsukerman (1992). In this reference, the authors propose an *overlapping elements* method which is valid only if the surfaces to be connected are regular. This drawback is overcome in Krebs, Clénet, & Tsukerman (2010) where the method is extended to non-planar surfaces introducing modified shape functions in the overlapping area. The method is valid for moving meshes without any distortion in it. The method has been developed for

three-dimensional meshes with hexahedral, tetrahedral or prism elements. The idea is to project the nodes of one surface to the other and these projections create virtual nodes and virtual elements. The virtual nodes do not entail new degrees of freedom and are just used to define the nodal shape function for the scalar potential in the overlapping area with the result that it is continuous in the whole domain. In the framework of Generalized Finite Element Methods (GFEM) we find the work of Duarte, Liszka, & Tworzydlo (2007) to unite non-matching meshes. In particular, the authors present a GFEM with clustering since they cluster a set of nodes and elements which define a modified finite element partition of unity that is constant over part of the clusters.
In the solid mechanics framework, the meshless technique could be useful for problems which present cracks or large deformations. See, for example, Cho & Im (2006) where different couplings are compared in such situations. Other references are Cho et al. (2005); Tian & Yagawa (2007); Kim (2008). Fluid-structure interaction is another example of applicability in the meshless and finite element combination for mesh joining application. In fact, one of the first works related to this issue is found in this application, see Attaway, Heinstein, & Swegle (1994).

Mesh-based techniques like the work of Lo & Wang (2004) or Cebral et al. (2001) also have also been applied to solve mesh union problems.

1.3.2 Chimera

This section describes the so-called Chimera methods. As stated above, when real problems have to be solved, complex geometries could be present. The Chimera methods, also called overset grid methods, are a proper tool to simplify the meshing process. Also the simulation of objects in relative motion is the other classical problem simplified with Chimera methods. A key point is that they allow flexibility in the choice of the type of the elements as well as their orientation. The origin of the terminology *Chimera* is the mythological Greek monster. Its descriptions were varied but all of them coincided in the fact that the monster was composed of different parts of other animals. And this characteristic is what gives the name to the strategy. In the Chimera problems, the mesh is composed of independent meshes embedded, one embedded inside the other. The idea came from Steger & Benek (1983, 1987); Steger (1991); Keeling (1997). At the beginning, the composed meshes were structured meshes. Firstly, Steger applied the method to solve the Euler equations in a transonic flow in a wing plane. But due to the non-conservative algorithm applied on the interfaces, the results between one domain and the Chimera problem differed very much. In fact, this issue is one of the main difficulties of the method. The interchange of the information in the interfaces to couple the independent meshes could be the source of the error in the solution. In Liu & Shyy (1996) and the references therein we can find a good discussion of this

question in the context of Navier-Stokes equations.

The appealing characteristics of the Chimera method have enabled many applications, summarized here:

- *Simplified mesh generation.* Landmann & Montagnac (2011); Zheng & Liou (2003) At the time when mesh generators could not handle complex geometries, the Chimera method was proposed as an alternative. Different meshes around the components of the computational domain are generated in an independent way. This enables a great flexibility in the choice of elements in each mesh.

- *Local refinement.* Kao, Liou, & Chow (1994); Meakin (1995); Tang, Casey, & Sotiropoulos (2003); Pärt-Enander (1995). When more accuracy is required in some specific parts of the computational domain, the Chimera method is a capable alternative in codes that cannot handle adaptivity. Local refinement can be achieved by simply placing a refined patch mesh onto the original mesh.

- *Moving components.* Meakin (1993); Prewitt, Belk, & Shyy (2000); Blades & Marcum (2007) The Chimera method is well suited for treating problems where components are moving (e.g.. opening flat). The independent meshes are moved as rigid bodies and the solution is recoupled when suited (e.g. each time step). This recoupling can be very costly and cumbersome when taking into consideration parallel solvers on distributed memory machines. However, they make it possible to maintain the boundary layers and local refinement around the bodies in a natural way.

- *Optimization.* Nielsen & Diskin (2013) Another straightforward application is the configuration optimization. A series of objects with independent meshes can be moved around without having to remesh the whole computational domain. Then, optimization techniques can be used to find the optimum configuration of the components.

The main advantage of the Chimera method for the last two applications with respect to a complete automatic remeshing stems from the fact that the component meshes move like rigid bodies from one time step/configuration to another. The coupling only affects the vicinity of the interfaces and, if the interfaces are far enough from the body, the impact of the coupling is expected to be minimal. A good review of the literature related to the first two applications is found in Wolf (2004).

The development of this technique has been widely treated in NASA aerospace applications. A good review of the work coming from the institution is Chan (2009). Examples of this include the Chimera Grid Tools project developed by Chan (see Chan (2005)), PEGASUS 5 (see Benek, Steger, Dougherty, &

Buning (1986); Rogers, Suhs, & Dietz (2003)) or the OVERFLOW series (see Jespersen, Pulliam, & P.G.Buning (1997)). PEGASUS is completely independent from the solver but will produce interpolation information for either finite difference or finite volume flow solvers. In the DRAGON meshes presented previously, the authors use the software PEGASUS to provide boundary or fringe nodes which are reordered according to their geometric coordinates. They need to introduce additional matrices to describe the connection between the structured and unstructured grids. In most of the codes for solving this kind of problem the experience of the user is a decisive factor to the success of the simulation. The details of the computational mesh have to be known as well as some trial and error test could be necessary to decide the best configuration of the hole-cutting process. This step is the first thing to do in these problems. It consists in eliminating the elements of the background mesh located just above the embedded meshes, called patch meshes. This process generates apparent interfaces where the coupling has to be imposed. To avoid the dependency of the experience of the user in Wang, Parthasarathy, & Hariharan (2000) the authors develop an automated Chimera strategy. In Landmann & Montagnac (2011) we find a highly automated parallel Chimera method for overset grids based on the implicit hole-cutting (IHC) technique. This algorithm is based on the one presented by Lee & Baeder (2003). In Fan & Chao (2010), three enhancements for the original overset grid assembly method are proposed. The Beggar code (see Maple & Belk (1994)) is capable of solving three-dimensional inviscid or viscous compressible flows problems involving multiple moving objects. It allows patched and overlapping structured grids in a framework that includes grid assembly, flow solution, force and momentum calculation and the integration of the rigid body, with a reduced number of user inputs. DCF3D (domain connectivity function), Meakin (1991), is another code used to achieve the grid assembly task in overset meshes. It has been used with the OVERFLOW solver. Smith & Pallins (1993) shows the parallel implementation of an overset grid flow solver for a network-based heterogeneous computing environment. It was derived from OVERFLOW. Parallel performance, but only for static problems, was presented in Meakin & Wissink (1999). The first presentation of the parallel implementation of grid assembly for dynamic overset grids was by Barszcz, Weeratunga, & Meakin (1993). DCF3D was parallelized and used in connection with a parallel version of OVERFLOW on a distributed memory parallel machine. In Cai, Tsai, & Liu (2006) a multi-block overset grid method using a combination of matched and overlapping grids to simulate turbulent flows over multi-element airfoils and three-dimensional wing-body combinations is presented. In Djomehri & Biswas (2003) the role of asynchronous communication, grid-splitting and grid-grouping strategies used with the OVERFLOW flow solver is analysed. CMPGRD, see Chesshire & Henshaw (1990), is another code for accomplishing the same objective and it has been included in the OVERTURE code, by Brown, Henshaw, & Quinlan (1999). This

code is developed in Los Alamos and it is designed to solve PDE allowing the adaptive grid refinement, moving objects and overlapping meshes to be solved in parallel or sequential version. Another software valid for Strucutured, Unstrucutured, and Generalized overset Grid AssembleR, (SUGGAR, see Noack (2005)) is devoted to the simulation of moving bodies. DirtLib: Donor interpolation Recep-tor Transaction library, described in Noack (2006) by the same author of the SUGGAR code, contains all the connectivity information between subdomains required by the solver.
Although it is true that Chimera methods are useful in a variety of applications, it is worth mentioning that the hole-cutting preprocess described before also demand a great deal of computational work which must be done in an efficient manner. In Guerrero (2007) successful and efficient applications of the Chimera Method are described.
From a more theoretical point of view, it should be highlighted the analysis of the Chimera method given by Brezzi, Lions, & Pironneau (2001). In this work, the Laplace operator in a Chimera problem is analysed. The domain is the union of two subdomains, $\Omega = \Omega_1 \cup \Omega_2$ with Ω_1 and Ω_2 open and such that $\Omega = \Omega_1 \cap \Omega_2 \neq \emptyset$. In the work of Keeling (1997) the authors analyse the differences between a differential equation system coming from one domain problem and another one coming from the resoultion of two overlapping subdomains. They identify the boundary operator, called trace operator needed to impose a proper transmission conditions on the interfaces as well as its numerical approximation.

Finally, we want to mention the following reference Sanders & Puso (2012) where the Nitsche method is applied in the simulation of fluid-structure problems and the discontinuous Galerkin discretization is used for non-linear materials. As well as other approach like the next work Katz & Jameson (2008) with an mesh-less interface as an alternative to interpolation

1.4 Main objectives

As we have seen in the review of the state of the art, the idea of coupling meshes or subdomains appears in a great number of contexts or applications, each of them with their own particularities. Next, we present a list in order to summarize the main problems detected in all of these different strategies.

- Main drawbacks of *Formulation-Based* Methods:
  - Iteration-by-subdomain methods introduce an additional iterative loop, are PDE dependency and difficult to implement.
  - Difficult to choose a proper space for the Lagrange Multiplier functions.

  - Modify the number of degrees of freedom in constraint methods.
  - Problems to determine the influence domain for the shape function in meshless methods.
  - Difficult numerical integration in some approximations used in meshless methods.

- Main drawbacks of *Mesh-Based* Methods:
  - DRAGON mesh implies modifying the number of degrees of freedom as well as a powered mesh generator.
  - Difficulty associated with generating a conformal mesh between solids where solid-solid interfaces are not obvious.
  - Loose tolerances or user error from CAD packages often make conformal mesh generation tedious.
  - Strong element type dependency.

With the HERMESH method, we want to achieve a general method, easy to implement, valid for any PDE, implicit and parallel. So the first requirements were the method are:

1 Implicit;

2 versatile;

3 parallel.

These three main aspects have been respected in the HERMESH method which in turn offers the following advantages:

4 Quasi-automatism: in some cases, the coupling can be done without a priori identifications of the interfaces.

5 Topology independency: submeshes can be disjoint with and without gap or overlapping; conforming and non-conforming.

6 Mesh size and mesh type independencies; the sizes and types of the elements of the different submeshes can be different.

7 Degrees of freedom conservation: the method introduces new elements but no additional degrees of freedom, which can be a very attractive feature from the implementation point of view.

All of these features will be studied in this manuscript.

Chapter 2

Computational Mechanics Equations

Before describing our method in depth, one might propose describing the physics and the numerical schemes of all the equations solved in this work. These are the equations which govern the following problems:

- *Ship Hull*: We have four coupled equations solved in a staggered manner, two equations for the Navier-Stokes equations: momentum equation and continuity equation, one equation of the Spalart-Allmaras turbulence model and one level set equation;.
- *Neuron test*: We solve large-scale computational solid mechanics. In particular, we consider the Neo-Hookean hyperelastic constitutive model.
- *Wind Farm*: This problem entails the solution of the RANS equations along with a k-ε turbulence model specially designed for the Atmospheric Boundary Layer and a specific law of the wall, including roughness effects.
- *Large and small airways*: We solve the Navier-Stokes equations for incompressible transient flow.
- *Bypass in an estenosed artery*: We solve the Navier-Stokes equations for incompressible flow.

All these problems have been addressed successfully with the HERMESH method presented in this work. This shows its considerable versatility. In each case, the advantages of solving the problem with HERMESH are different, as is explained below.

Ship hull. In this problem we simulate the free surface flow around a ship hull with the RANS equations together with the Spalart-Allmaras turbulence model and a hyperbolic equation for the level set. The computational domain is composed of two independent meshes. On the one hand, we have generated a mesh for the hull, including the boundary layer. On the other hand, we have generated a mesh for the background in which the hull is embedded. HERMESH applied to this problem could be useful in the optimization purpose if we would want to test different keels.

Neuron test. In this case we have been able to prove that the application of the HERMESH method in solid mechanic equations is valid for a real and complex geometry. In particular, we simulate the real geometry of a neuron composed of two different materials, one corresponding to the nucleus and the other to the body of the neuron. The nucleus is assumed to be three times stiffer than the cell body. The applpication of HERMESH to this problems could be very useful when different parts of the neuron are meshed independently.

Wind farms. We present a CFD modelling strategy for wind farms aimed at predicting and optimizing the production of farms. The CFD model includes meteorological data assimilation, complex terrain and wind turbine effects. The model involves the solution of the RANS equations together with a k-ε turbulence model specially designed for the Atmospheric Boundary Layer. As the integration of the model up to the ground surface is still not viable for complex terrains, a specific law of the wall including roughness effects is implemented. The wake effects and the aerodynamic behaviour of the wind turbines are described using the actuator disk model, which consists of a volumetric force included in the momentum equations. The placement of the wind turbines and a mesh refinement for the near wakes is done by means of a Chimera method based on HERMESH coupling. It consists in having an independent patch for each turbine and coupling all of them to the background mesh which contains the topography. There is, in this case, a large difference in mesh sizes between the non-structured patch mesh and the structured background mesh. The application of the HERMESH method to the optimization of the position of the wind turbines in the wind farms saves a huge amount of time and memory computational resources. In fact, one does not have to remesh for each different configuration and it allows the combination of a coarse structured mesh for the background and a fine and non-structured mesh around the wind turbines to capture the effects of the wakes.

Large and small airways. This simulation comes from joint research between the Barcelona Supercomputing Center (BSC), Imperial College (IC) and Jackson State University (JSU). On the one hand, IC obtains a mesh for the large airways down to the trachea. On the other hand, JSU generates an arbitrary number of generations of the idealized broncho-pulmonary tree in an automatic manner. With the method presented in this work, the HERMESH method, both independent meshes will be coupled in such a way that the global mesh is ready to be treated to solve the transient flow.

Bypass in an stenosed artery. Finally, we propose applying the HERMESH method to couple the bypass with the vessel. In this way, the geometry of the bypass can be easily changed in order to test various configurations (angles and diameters), without any need to remesh the complete computational domain.

During the development of this thesis, some results have been obtained for the solution of the general Advection-Diffusion-Reaction (ADR). For this equation we have developed a new stabilization method based on a two-scale variational method. This strategy will be adapted and applied for the rest of the equations presented before, except for the solid mechanical problems, as stabilization is not needed. The stabilization method presented in this work is based on the following. A two-scale variational method is applied to the one-dimensional problem. Then, a series of approximations is assumed to analytically solve the subgrid scale equation. This enables one to devise expressions for the "stabilization parameter" τ, in the context of a variational multiscale (two-scale) method. The method is equivalent to the traditional Green method used in the litterature, although it offers another point of view. The scheme proposed is compared to existing methods by means of solving a simple numerical example.

2.1 Advection-Diffusion-Reaction Equation and Stabilization

The advection-diffusion-reaction equation we want to focus on is:

$$L(u) := -\varepsilon\Delta u + \nabla\cdot(\boldsymbol{a}u) + su = f \qquad \text{in} \qquad \Omega, \tag{2.1}$$

where Ω is a n_d-dimensional (open and bounded) domain ($n_d = 1, 2, 3$) with boundary $\partial\Omega$; ε is the diffusion constant of the medium; f is the force term; $\boldsymbol{a}$ is the advection field (not necessary solenoidal) and s is a constant source

term. This equation should be provided with boundary conditions.

Let Ω^k be a regular finite element partition of the domain Ω, with index K ranging from 1 to the number of elements. The diameter of Ω^k will be denoted by h as usual. Let us construct (at least mentally...) the functional linear spaces from the previous partition so that the resulting finite element approximation is said to be *conforming*. Given two functions u and v we define $(u, v)_w$ as the L-inner product in w (if w is omitted the scalar product is considered over Ω). Assuming homogeneous Dirichlet and Neumann boundary conditions, the discrete Galerkin formulation of the problem consists in finding u_h in the appropriate space U_h such that

$$a(u_h, v_h) = (f, v_h) \qquad v_h \in V_h, \tag{2.2}$$

where V_h is an appropriate test function space and the bilinear form a is defined as

$$a(u_h, v_h) := \varepsilon(\nabla w; \nabla v) + (\boldsymbol{a} \cdot \nabla w, v) + (sw, v). \tag{2.3}$$

Note that for the sake of clarity, some technical details have been and will be omitted (e.g. the right-hand side of Equation 2.2 should be the duality paring). As has been well-known for a long time, the Galerkin method is infradiffussive and can lead to oscillant solutions. Stabilized finite element techniques have been developed to improve the numerical stability. We will make a quick review of these methods developed until now in order to set the variational multiscale framework. We will only consider those techniques which have naturally led to the settings of the variational multiscale family and will not consider techniques such as the Characteristic-Galerkin or the Taylor-Galerkin methods. The effects of the stabilization techniques we are interested in consist of the addition of a stabilization term $s(u_h, v_h)$ to the original equation so that the stabilized system reads:

$$a(u_h, v_h) + s(u_h, v_h) = (f, v_h) \qquad v_h \in V_h, \tag{2.4}$$

We define the residual of the equation as

$$R(u_h) := f - L(u_h). \tag{2.5}$$

The first stabilization methods that were developed were called *artificial viscosity* (AV) methods, Johnson (1987). They consist in adding a viscosity-like term to the equation whose coefficient was originally a constant; sufficiently small to avoid over-diffusion and sufficiently large to stabilize the solution. These methods are not consistent and, as the additional term is of first order accuracy, the convergence of the method is not optimal.
To remedy this, second order artificial viscosity methods have been developed,

see Baruzzi, Habashi, & Hafez (1992). To correct the indiscriminate character of artificial viscosity, the *streamline upwind* (SU) was introduced in Hughes & Brooks (1979). This method is less diffusive than the AV method as the diffusion is only added in the streamline direction, but the non-consistency of artificial viscosity methods is not improved.
Simultaneously with their work on the advection-diffusion systems of equations, the Standford team (Hughes, Franca, & Balestra (1986)) devised a Petrov-Galerkin formulation for solving the Stokes problem which avoided the need to satisfy the Babuska-Brezzi stability condition, by adding a new perturbation to the continuity equation, proportional to the pressure gradient test function. Following the same ideas, the *Galerkin/Least-square* (GLS) method was presented in Hughes, Franca, & Hulbert (1989) as a "conceptual simplification" and generalization of the SUPG method for advection-diffusion equations with a negative perturbation term proportional to the whole differential operator of the test fuction.
At the same time, Douglas & Wang (1989) developed a stabilization technique, known as the *Douglas-Wang* (DW) method, for the Stokes problem changing the sign of the Laplacian of the perturbation function. This method was soon applied to the ADR equation (Franca, Frey, & Hughes (1992)) substituting the differential operator L to minus its adjoint L^*. During that time, the different expressions for the stabilization parameter were obtained in two main ways; obtaining a nodally exact solution for simple one-dimensional problems and using convergence analysis. In 1995 it was shown by Hughes (1995) that stabilized methods could be derived from a *variational multiscale* (VMS) formulation considering that the basis of residual-based, or consistent, stabilized methods is a variational multiscale analysis of the partial differential equations under consideration. This approach combines ideas of physical modelling with numerical approximation in a unified way. The numerical instabilities of the Galerkin method are due to the subgrid (unresolved) scales and their effects must be modelled at the grid (resolved) level. This method not only explains the instabilities but also clearly identifies the intrinsic time τ.
In the same sense, the bubbles method (Brezzi, Franca, Hughes, & Russo (1997a)) can be interpreted as a particular case (and implementation) of a VMS method when the coarse scale space consists of linear polynomials.
Table 2.1 shows the expression of the stabilization term $s(u_h, v_h)$ for the stabilization methods described earlier, where τ is a stabilization parameter that can depend on the element size h and the equation coefficients, and where

$$\int_{\Omega'} (\cdot) d\Omega := \sum_K \int_{\Omega_K} (\cdot) d\Omega,$$

is the integral over all the elements. Note that in both AV and SU methods, the stabilization parameter has sometimes been regarded as constant over the mesh.

Method	Stabilization term $s(u_h, v_h)$
Non-consistent	
Artificial viscosity (AV)	$\int_{\Omega'} \tau(\nabla v_h \cdot \nabla u_h)\, d\Omega$
Streamline upwind (SU)	$\int_{\Omega'} (a \cdot \nabla v_h)\tau(a \cdot \nabla u_h)\, d\Omega$
Consistent	
SU Petrov-Galerkin (SUPG)	$-\int_{\Omega'} (a \cdot \nabla v_h)\tau R(u_h)\, d\Omega$
Galerkin Least-Square (GLS)	$-\int_{\Omega'} L(v_h)\tau R(u_h)\, d\Omega$
Douglas-Wang (DW)	$\int_{\Omega'} L^*(v_h)\tau R(u_h)\, d\Omega$
Variational Multiscale	
Algebraic Models (ASGS)	$\int_{\Omega'} L^*(v_h)\tilde{u}\, d\Omega \quad$ with $\tilde{u} = \tau R(u_h)$

Table 2.1: Stabilization term $s(u_h, v_h)$ of common stabilization methods 2.4.

The variational subgrid scale (VSGS) method is a two-scale variational multiscale method which offers a general framework for stabilization methods. From a splitting of the exact solution u into a grid (coarse) scale u_h and a subgrid (fine) scale $\tilde{u}$ and a substitution of u into the continuous weak form, one can obtain a system of two weak equations for these unknowns. At this stage, simplifications must be done to solve the subgrid scale equation. In the litterature, the usual approximation consists in taking the subgrid scale as element-wise and solving for the following equation

$$L(\tilde{u}) = R(u_h) \tag{2.6}$$

in each element. The differential operator L is usually approximate by an algebraic operator $\tau \approx L^{-1}$; these methods are referred to as algebraic subgrid scale methods (ASGS). Then the expression for the subgrid scale is substituted into the grid scale equation which is solved for u_h using the Galerkin method. Figure 2.1 shows the road map of stabilization methods (this figure was strongly inspired by that of Hughes, Feijóo, Mazzei, & Quincy (1998)).

The different ASGS methods proposed in the litterature differ essentially in two points. First, the choice for the subgrid scale space; then, the manner in which the differential operator is approximated, that is how τ is chosen. The main characteristics of the method proposed here consists of three main assumptions. The first one consists in considering a one-dimensional problem. The second one consists of the choice of the subgrid scale: the bubble space. The third one consists in taking the right-hand side of the subgrid scale equation (the residual $R(u_h)$) constant and solving the subgrid scale analytically.

The first part of this chapter is organized as follows. We derive a family of weak forms, depending on the integration by parts of the advection term. This leads to a different natural condition of Robin type, and different coercivity

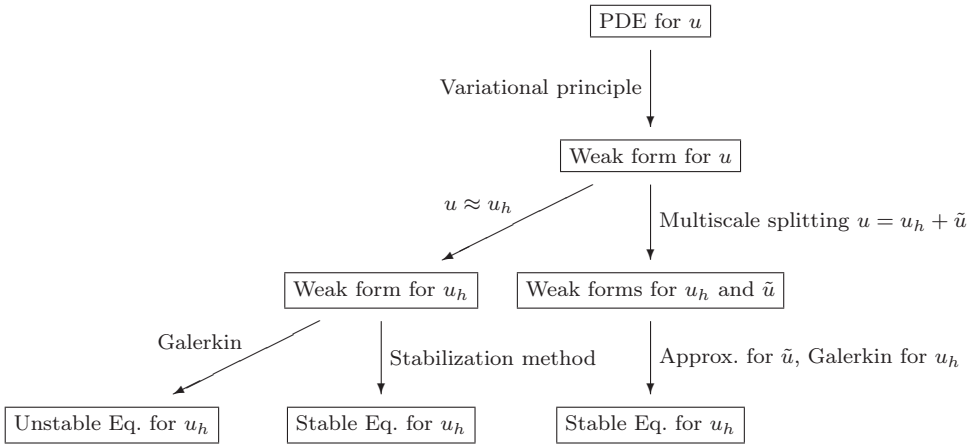


Figure 2.1: Road map of stabilization methods

properties of the bilinear operator. In the framework of subgrid scale models, it is therefore legitimate to ask if the Robin transmission condition should include the subgrid scale effects. Next, we present the ASGS model and, in particular, derive the expressions for the stabilization parameter τ. To finalise this first part of this chapter, we aim to solve some numerical examples to demonstrate the behavior of the expression comparing it with others proposed in the litterature, in the full range of physical parameters.

Weak formulations. We derive a family of weak forms, by integrating the diffusion term by parts as well as a fraction $(1-\flat)$ of the advective term, where $\flat \in [0,1]$. The corresponding formulation will be referred to as $(1-\flat)$-weak formulation. We split the boundary $\partial\Omega$ into two components denoted Γ_D and Γ_N, so that $\partial\Omega = \overline{\Gamma_D \cup \Gamma_N}$. We introduce the following definitions:

$$\begin{aligned} H^1_{\Gamma_D}(\Omega) &:= \{v \in H^1(\Omega) \mid v_{|\Gamma_D} = 0\}, \\ V^0 &:= H^1_0(\Omega), \\ V &:= H^1_{\Gamma_D}(\Omega). \end{aligned}$$

We endow $H^1(\Omega)$ with usual scalar product $(\cdot,\cdot)_1$ and associated norm $\|\cdot\|_1$.

Let us consider our differential problem (2.1). We restrict ourselves to solutions in $H^1(\Omega)$, which obliges us to choose the force term so that

$$f \in H^1(\Omega)'.$$

In order to be able to show the existence and uniqueness of the solution, we must assume that all the remaining terms present in the last equation are bounded, i.e.

$$s \in L_\infty(\Omega), \nabla \cdot \boldsymbol{a} \in L_\infty(\Omega), \boldsymbol{a} \in L_\infty(\Omega)^{n_d}. \tag{2.7}$$

We first note that according to the Gauss theorem, we rewrite the advective term as follows:

$$\begin{aligned}\int_\Omega(\boldsymbol{a}\cdot\nabla w)v\,d\Omega &= (1-\flat)\int_\Omega(\boldsymbol{a}\cdot\nabla w)v\,d\Omega+\flat\int_\Omega(\boldsymbol{a}\cdot\nabla w)v\,d\Omega\\ &= (1-\flat)\int_\Omega(\boldsymbol{a}\cdot\nabla w)v\,d\Omega+\flat\int_{\Gamma_N}(\boldsymbol{a}\cdot\boldsymbol{n})wv\,d\Gamma\\ &\quad -\flat\int_\Omega(\boldsymbol{a}\cdot\nabla v)w\,d\Omega-\flat\int_\Omega(\nabla\cdot\boldsymbol{a})wv\,d\Omega.\end{aligned} \tag{2.8}$$

where $\flat$ corresponds to the part of convective term which is integrated by parts. Therefore, the weak formulation reads: find $u\in V$ so that

$$a(u,v)=l(v)\qquad\forall\, v\in V, \tag{2.9}$$

where

$$\begin{aligned}a(w,v) &:= \varepsilon(\nabla w,\nabla v)+(1-\flat)(\boldsymbol{a}\cdot\nabla w,v)-\flat(w,\boldsymbol{a}\cdot\nabla v)\\ &\quad +([s+(1-\flat)\nabla\cdot\boldsymbol{a}]w,v),\\ l(v) &:= (g,v)_{\Gamma_N}+(f,v)_\Omega,\end{aligned} \tag{2.10}$$

where the natural condition consists in prescribing g on Γ_N defined as

$$g:=\varepsilon\frac{\partial u}{\partial n}-\flat(\boldsymbol{a}\cdot\boldsymbol{n})u. \tag{2.11}$$

Although strictly algebraically equivalent, the formulations stemming from the different values of $\flat$ have two major differences: their respective natural conditions and their variational properties. The former has just appeared explicitly, as the natural boundary condition is given by Equation (2.11). The latter becomes evident when we study the existence and uniqueness of the solutions. From Lax-Milgram lemma, problem (2.9) has a unique solution if $a(w,v)$ is both continuous and coercive for any $w,v\in V$. It can be shown that for any $\flat\geq 0$ the bilinear form is continuous. However, coercivity is subjected to the following conditions, as see Quarteroni & Valli (1994); Houzeaux & Codina (2002):

$$s+\frac{1}{2}\nabla\cdot\boldsymbol{a}\geq 0\qquad\text{almost everywhere,} \tag{2.12}$$

and the following condition on the relative values of the coefficients:

$$\varepsilon>C'|1-2\flat|\,\|\boldsymbol{a}\cdot\boldsymbol{n}\|_{\infty,\Gamma_N} \tag{2.13}$$

which implies that $\|\boldsymbol{a}\cdot\boldsymbol{n}\|_{\infty,\Gamma_N}$ should not be too high with respect to the diffusion ε. We will distinguish three cases, $\flat=0,1/2,1$. The unique solution is therefore conditioned by the following:

1. The data of the problem are such that $s \in L_\infty(\Omega), \boldsymbol{a} \in L_\infty(\Omega)^{n_d}$;

2. The source term and advection satisfy $s + \frac{1}{2}\nabla \cdot \boldsymbol{a} \geq 0$ almost everywhere.

3. Finally, we require that

   - 0-weak formulation: Γ_N is an outflow or the advection is not too high in the sense of Equation (2.13).
   - 1/2-weak formulation: no additional condition.
   - 1-weak formulation: Γ_N is an inflow or the advection is not too high in the sense of Equation (2.13).

We can easily see the great advantage that the weak 1/2-weak formulation has over the other two: apart from Inequality (2.12), no condition on the magnitude and direction of $\boldsymbol{a}$ is required on Γ_N.

2.1.1 Multiscale Approach

We would like to review the shortcomings of the Galerkin finite element formulation. For the sake of clarity, we will consider here the stationary and homogeneous Dirichlet problem with $\nabla \cdot \boldsymbol{a} = 0$ and $\flat = 0$. We can derive the following error estimate, Quarteroni & Valli (1994)

$$\|u - u_h\|_1 \leq c \frac{M}{N} h^m \|u\|_{m+1}, \tag{2.14}$$

with

$$\begin{aligned} M &= \varepsilon + \|\boldsymbol{a}\|_{\infty,\Omega} + \|s\|_{\infty,\Omega}, \\ N &= \tfrac{\varepsilon}{1+C_\Omega}, \end{aligned}$$

where C_Ω is a geometrical constant, h is the maximum diameter of the polyhedron of the triangulation, and c is a constant depending on the geometry and triangulation of Ω, but not on h. Error estimate (2.14) is optimal in the H^1 norm, so we conclude that the Galerkin method can lack stability when $M \gg N$, that is, when the diffusion ε is small compared to $\|\boldsymbol{a}\|_{\infty,\Omega}$ and $\|s\|_{\infty,\Omega}$ and if h is not sufficiently small. In fact, taking $u = v = u_h$ in Equation (2.10), we have

$$a(u_h, u_h) = \varepsilon \|\nabla u_h\|_0^2 + \|s^{1/2} u_h\|_0^2,$$

as the convective term disappears when it is integrated by parts. We observe that we have no control on the advective term of the equation. In addition, when $s^{1/2}$ is high, we gain control of the L_2 norm of the unknown at the expense of losing control of its gradient. This is why a stabilization method is necessary.

Expressions for τ. We now review some expressions for τ that have been proposed in the litterature. The first expressions were designed in the context of SUPG, GLS or DW method. Typically, two ways have been undertaken; on the one hand, with the help of convergence analysis and error estimate; on the other hand, to obtain a nodally exact solution in some simple 1D problems and using the straighforward extension to multidmensional cases. The Fourier analysis and the use of Green's function was specially designed in the context of ASGS models. In these models, the key point for designing the stabilization parameter is the algebraic approximation of the differential operator $\tau \approx L^{-1}$.

Let us define a as an appropriate norm of the advection $\boldsymbol{a}$ (which is not detailed here for the sake of clarity).

Discrete maximum principle. Codina (1998) obtains the following expression for τ requiring that the associated algebraic system leads to a solution that statisfies the discrete maximum principle:

$$\tau = \left[\frac{4\varepsilon}{h^2} + \frac{2a}{h} + s\right]^{-1}. \tag{2.15}$$

A similar expression is proposed by Shakib *et al.* Shakib, Hughes, & Johan (1991):

$$\tau = \left[\left(\frac{4\varepsilon}{h^2}\right)^2 + \left(\frac{2a}{h}\right)^2 + s^2\right]^{-1/2}. \tag{2.16}$$

Fourier analysis. Another strategy is that used by Codina (Codina (2000)) in the VMS context. The subgrid scale equation is expressed in the Fourier space within each element, and approximated by taking into account the subscales which contain only high wave numbers. This last assumption enables one to get rid of a boundary term. Defining

$$\widehat{g}(\boldsymbol{k}) := \int_K e^{-i\frac{\boldsymbol{k}\cdot\boldsymbol{x}}{h}} g(\boldsymbol{x}) d\Omega_x,$$

one can obtain

$$\widehat{\frac{\partial g}{\partial x_j}}(\boldsymbol{k}) \approx i\frac{k_j}{h}\widehat{g}(\boldsymbol{k}), \quad \widehat{\frac{\partial^2 g}{\partial x_i \partial x_j}}(\boldsymbol{k}) \approx -\frac{k_i k_j}{h^2}\widehat{g}(\boldsymbol{k}),$$

where $\boldsymbol{k} = (k_1, \ldots, k_d)$ is the dimensionless wave number and h the diameter of Ω_K. The result of this is:

$$\begin{aligned} \widehat{\tilde{u}}(\boldsymbol{k}) &\approx \widehat{\tau}(\boldsymbol{k})\widehat{R}(\boldsymbol{k}), \\ \widehat{\tau}(\boldsymbol{k}) &:= \left[\varepsilon\tfrac{|\boldsymbol{k}|^2}{h^2} + i\tfrac{\boldsymbol{a}\cdot\boldsymbol{k}}{h} + s\right]^{-1}. \end{aligned}$$

Then Plancherel's formula and the mean value theorem are applied to obtain the mean energy of the subgrid scale obtained in each element, from which the

stabilization parameter τ is obtained:

$$\tau = \left[\left(c_1 \frac{\varepsilon}{h^2} + s \right)^2 + \left(c_2 \frac{a}{h} \right)^2 \right]^{-1/2},$$

where c_1 and c_2 are constant independent of the equation coefficients and h.

Bubble condensation. The bubble method can be viewed as a particular case of the VMS method. Traditionally, they have been treated in separate contexts due to their different associated numerical approaches. In the case of the VMS models, the subgrid scale is generally expressed as a proportional to the residual $\tau R(u_h)$ while in the bubble method, the subgrid scales are solved numerically, and are generally eliminated using static condensation. In this last context, one can nevertheless relate both methods and obtain the τ expression in terms of the bubbles basis function. An expression of the parameter was designed from convergence and stability theory by Franca and Valentin (Franca & Farhat (1995)):

$$\tau = \min \left(\frac{h}{2a}, \left[\frac{\varepsilon}{h^2} + s \right]^{-1} \right).$$

Green's function. The Green function $g(x, y)$ is introduced to set an equivalent problem to the subgrid scale Equation 2.6 (Hughes et al. (1998)). This enables one to "propagate" the subgrid scale effects into the grid scale equation. As in common ASGS methods, the Green function is localized and is obtained in each element independently as:

$$\begin{array}{rcll} L^*(g(x,y)) & = & \delta(x-y) & \text{in } \Omega_K, \\ g(x,y) & = & 0 & \text{on } \Gamma_K. \end{array}$$

From the solution of this problem the stabilization parameter is computed as:

$$\tau = \frac{1}{h} \int_{\Omega_K} \int_{\Omega_K} g(x,y) d\Omega_x \Omega_y. \tag{2.17}$$

Hauke & García-Olivares (2001) obtains with this method the following expression:

$$\tau = \frac{1}{h} \frac{sh(-1 + e^{h(\lambda_2 - \lambda_1)}) - (1 + e^{h(\lambda_2 - \lambda_1)} - e^{-h\lambda_1} - e^{h\lambda_2})(\lambda_2 - \lambda_1)\varepsilon}{s^2(-1 + e^{h(\lambda_2 - \lambda_1)})}, \tag{2.18}$$

where

$$\begin{array}{rcl} \lambda_1 & = & \frac{a - \sqrt{a^2 + 4\varepsilon s}}{2\varepsilon}, \\ \lambda_2 & = & \frac{a + \sqrt{a^2 + 4\varepsilon s}}{2\varepsilon}. \end{array}$$

Let us outline two important points from what has been done in the litterature. All the methods employed for designing the stabilization parameters

in the context of ASGS involve an averaging over the element: this is why τ depends on the diameter h of the element. The other point is that the subgrid scale is assumed to be "small" or zero on the element boundaries: this assumption is explicit in the case of the Green function and implicit in the case of the Fourier analysis (a commendable characteristic of this method). Note that in the wonderful one-dimensional world, when one requires the grid scale to be nodally exact, one explicitly requires the subgrid scale to be nodally zero.

The method we propose in this work is similar to the previous ones in the sense that a spatial averaging is performed and the subgrid scale is nodally zero. The next generation of stabilization methods, in the framework of ASGS formulations, will certainly overcome these deficiencies. This is necessary for treating anisotropic meshes and for better taking into account the local variation of the subgrid scale within an element.

A variational two-scale method. Let us decompose the exact solution u into a grid scale (resolved) u_h and a subgrid scale $\tilde{u}$ so that

$$u = u_h + \tilde{u}, \tag{2.19}$$

where $\tilde{u}$ belongs to a space $\tilde{U}$ that completes U_h in U. That is $U = U_h \oplus \tilde{U}$. The same sum is performed for the test function space $V = V_h \oplus \tilde{V}$.

The weak form written as a system of equations reads: find $(u_h, \tilde{u}) \in U_h \times \tilde{U}$ such that

$$a(u_h + \tilde{u}, v_h + \tilde{v}) = l(v_h + \tilde{v}) \quad \forall\, (v_h, \tilde{v}) \in V_h \times \tilde{V}.$$

By taking successively $v_h = 0$ and $\tilde{v} = 0$ we obtain the following system:

$$\begin{array}{lcll} a(u_h, v_h) & + & a(\tilde{u}, v_h) & = & l(v_h) & \forall\, v_h \in V_h, \\ a(u_h, \tilde{v}) & + & a(\tilde{u}, \tilde{v}) & = & l(\tilde{v}), & \forall\, \tilde{v} \in \tilde{V}. \end{array}$$

Let us define

$$\int_{\Omega'} (\cdot)\, d\Omega := \sum_K \int_{\Omega_K} (\cdot)\, d\Omega, \tag{2.20}$$

$$\int_{\partial\Omega'} (\cdot)\, d\Gamma := \sum_K \int_{\partial\Omega_K} (\cdot)\, d\Gamma, \tag{2.21}$$

$$\int_{\partial\Omega''} (\cdot)\, d\Gamma := \sum_K \int_{\partial\Omega_K} (\cdot)\, d\Gamma - \int_{\partial\Gamma_N} (\cdot)\, d\Gamma. \tag{2.22}$$

Now the fourth term of the grid scale equation is substituted by the sum of the integral over the elements Ω_K and it is integrating back by parts. The

same is done for the subgrid scale equations. We then obtain:

$$\begin{aligned} &a(u_h, v_h) + \textstyle\int_{\Omega'} L^*(v_h)\tilde{u}\, d\Omega \\ &\qquad + \textstyle\int_{\partial\Omega'} \tilde{u}(\varepsilon\nabla v_h \cdot \boldsymbol{n} + (1-\flat)(\boldsymbol{a}\cdot\boldsymbol{n})v_h)d\Gamma = l(v_h) \qquad \forall\, v_h \in V_h, \\ &\textstyle\int_{\Omega'} L(u_h)\tilde{v}d\Omega + \int_{\Omega'} L(\tilde{u})\tilde{v}d\Omega \\ &\qquad + \textstyle\int_{\partial\Omega''} \tilde{v}(\varepsilon\nabla u \cdot \boldsymbol{n} - \flat(\boldsymbol{a}\cdot\boldsymbol{n})u)d\Gamma = \int_{\Omega'} f\tilde{v}d\Omega \qquad \forall\, \tilde{v} \in \tilde{V}. \end{aligned} \tag{2.23}$$

where the adjoint L^* of operator L is

$$L^*(v_h) = -\varepsilon\Delta v_h - \boldsymbol{a}\cdot\nabla v_h + s v_h.$$

Note that the adjoint does not contain the compressiblity term $(\nabla\cdot\boldsymbol{a})$.

Up to now, no approximation has been introduced. The idea is the following: first, to "simplify" the subgrid scale equation; then, to obtain a solution to this approximate equation; finally, to substitute this solution in the grid scale equation.

A methodology for the solution of the subgrid scale equation. Remembering the definition of the residual given by Equation (2.5), we can rewrite the subgrid scale equation as

$$\int_{\Omega'} L(\tilde{u})\tilde{v}d\Omega = \int_{\Omega'} R(u_h)\tilde{v}d\Omega - \int_{\partial\Omega''} \tilde{v}(\varepsilon\nabla u\cdot\boldsymbol{n} - \flat(\boldsymbol{a}\cdot\boldsymbol{n})u)d\Gamma \quad \forall\, \tilde{v} \in \tilde{V}. \tag{2.24}$$

The procedure to compute the subgrid scale consists in considering a simplified problem for which we can obtain a solution to the subgrid scale and then generalize it. The procedure is described in the following and sketched in Figure 2.2.

1. Consider a one-dimensional problem with constant coefficients.
2. Assume that the subgrid scale is zero on the element nodes, that is we sought for a nodally exact grid scale solution u_h. Therefore, the boundary term in Equation (2.24) disappears. Note that we explicitly choose $\tilde{V} = \tilde{V}_0$ to be the bubble space in each element; the resulting equation is:

$$\int_{\Omega'} L(\tilde{u})\tilde{v}d\Omega = \int_{\Omega'} R(u_h)\tilde{v}d\Omega \quad \forall\, \tilde{v} \in \tilde{V}_0. \tag{2.25}$$

For an element $K = [0, h]$, the equivalent strong form of this problem is:

$$L(\tilde{u}) = R(u_h), \quad \tilde{u} = 0 \text{ at } x = 0, h.$$

3. Assume that the grid scale residual $R(u_h)$ is constant.

4. Set $\tilde{u} = \tau(x)R(u_h)$. Substitute this equality into last equation and solve for $\tau(x)$:
$$L(\tau(x)) = 1, \quad \tau(x) = 0 \text{ at } x = 0, h. \tag{2.26}$$

5. Compute the average $\overline{\tau}$ of $\tau(x)$ over the element as:
$$\overline{\tau} = \frac{1}{h}\int_0^h \tau(x)dx.$$

6. Approximate the subgrid scale as:
$$\tilde{u}(x) = \overline{\tau}R(u_h). \tag{2.27}$$
Thus, the stabilization parameter $\overline{\tau}$ is an approximation to the inverse of the differential operator
$$\overline{\tau} \approx L^{-1}.$$
The proposed method is therefore an Algebraic Subgrid Scale model (ASGS).

7. This expression is generalized to the multidimensional and variable constant case so that the subgrid scale effects on the grid scale equation becomes:
$$\int_{\Omega'} L^*(v_h)\tilde{u}d\Omega = \int_{\Omega'} L^*(v_h)\overline{\tau}(\varepsilon, \boldsymbol{a}, s)R(u_h)d\Omega. \tag{2.28}$$

We now study why the expression for τ obtained by Hauke (2.18) (see Section 2.1.1) using the Green method is the same as the one presented in this work. Let us consider the strong form of problem (2.6) and express the solution by introducing the Green function $g(x, y)$:
$$\tilde{u}(y) = \int_{\Omega_x}\int_{\Omega_y} g(x, y)R(u_h(x))d\Omega_x d\Omega_y. \tag{2.29}$$

We have to find $g(x, y)$ such that $\tilde{u}$ is a solution of Equation (2.6). If we introduce Equation (2.6) in the last expression and integrate by parts we obtain:
$$\begin{array}{rcl} \tilde{u}(y) & = & \int_{\Omega_x}\int_{\Omega_y} g(x, y)L(\tilde{u}(x))d\Omega_x d\Omega_y, \\ & = & -\int_{\Omega_x}\int_{\Omega_y} L^*g(x, y)\tilde{u}(x)d\Omega_x + \int_{\partial\Omega_x}\int_{\partial\Omega_y} g(x, y)B\tilde{u}d\Gamma_x, \end{array}$$

where B is a certain boundary operator. Thus, if the Green function $g(x, y)$ satisfies
$$\begin{array}{rcll} L^*g(x, y) & = & \delta(x - y) & \text{in } \Omega_K, \\ g(x, y) & = & 0 & \text{on } \Gamma_K, \end{array}$$

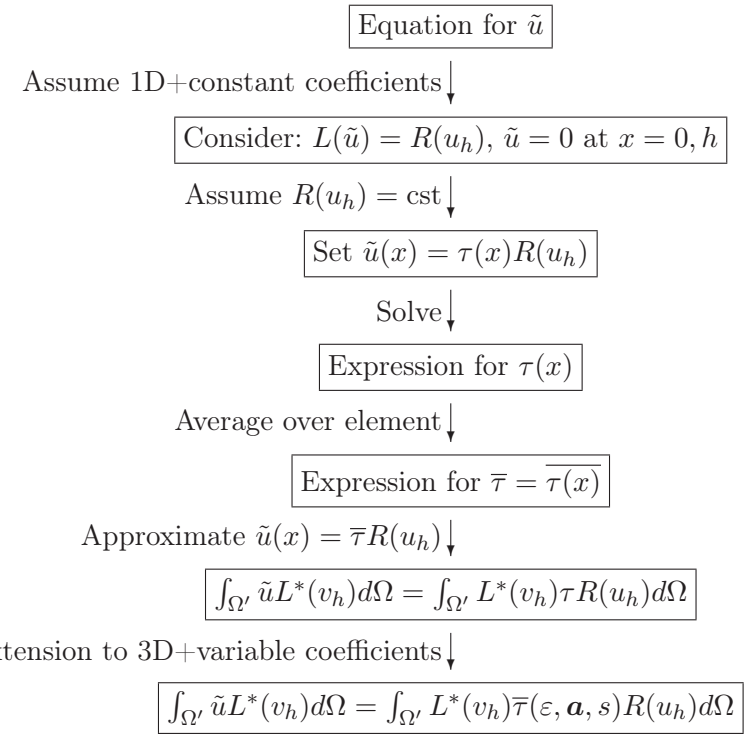


Figure 2.2: Procedure to compute the subgrid scale.

then $\tilde{u}$ can be expressed as given by Equation (2.29).
Now, let us consider the effect of the subgrid scale as given by Equation (2.29) in the grid scale equation. We have:

$$\begin{aligned} \int_{\Omega'} L^*(v_h)\tilde{u}d\Omega &= \int_{\Omega'}\int_{\Omega'} L^*(v_h(y))g(x,y)R(u_h(x))d\Omega_x d\Omega_y, \\ &= \sum_K \int_{\Omega_K}\int_{\Omega_K} L^*(v_h(y))g(x,y)R(u_h(x))d\Omega_x d\Omega_y. \end{aligned}$$

By assuming that the product $L^*(v_h(y))R(u_h(x))$ is constant, the subgrid scale effects reduces to

$$\int_{\Omega'} L^*(v_h)\tilde{u}d\Omega = \sum_K \int_{\Omega_K}\int_{\Omega_K} g(x,y)d\Omega_x d\Omega_y \left(\tfrac{1}{h}\int_{\Omega_K} L^*(v_h)R(u_h)d\Omega\right).$$

By identifying the average of the Green function with the stabilization parameter $\overline{\tau}$,

$$\int_{\Omega'} L^*(v_h)\tilde{u}d\Omega = \int_{\Omega_K} L^*(v_h)\overline{\tau}R(u_h)d\Omega,$$

so that we identify $\tilde{u}$ with $\overline{\tau}R(u_h)$.

Approximate solution of the subgrid scale. Following the algorithm sketched in Figure 2.2, we first have to solve for the subgrid scale on a generic element, and then compute the average over this element, while considering the right-hand side (grid scale residual) constant. We are going to treat the second order as well as the first-order following equations:

$$\begin{aligned}
\text{ADR}: \quad L(\cdot) &= -\varepsilon \tfrac{d^2(\cdot)}{dx^2} + a\tfrac{d(\cdot)}{dx} + s(\cdot), \\
\text{AD}: \quad L(\cdot) &= -\varepsilon \tfrac{d^2(\cdot)}{dx^2} + a\tfrac{d(\cdot)}{dx}, \\
\text{AR}: \quad L(\cdot) &= a\tfrac{d(\cdot)}{dx} + s(\cdot), \\
\text{DR}: \quad L(\cdot) &= -\varepsilon \tfrac{d^2(\cdot)}{dx^2} + s(\cdot).
\end{aligned}$$

with $\varepsilon > 0$ and $s > 0$, as well as $a > 0$ without loss of generality. We present in this subsection the results for $\overline{\tau}$, the average $\tau(x)$ over an element, where $\tau(x)$ is solution of Equation (2.26). That is:

$$\overline{\tau} = \frac{1}{h}\int_0^h \tau(x)dx,$$

with

$$L(\tau(x)) = 1 \text{ in } [0,h], \text{ with } \begin{cases} \tau(x)\mid_{0,h} = 0 & \text{for ADR, AD, DR,} \\ \tau(x)\mid_{0} = 0 & \text{for AR.} \end{cases} \tag{2.30}$$

Note that the case of the AR equation, $\tau(x)$ is only prescribed to zero at the inflow.

Let us introduce the following element dimensionless numbers:

$$\begin{aligned}
\text{Element Péclet:} \qquad \text{Pe}_h &= \frac{ah}{2\varepsilon}, \\
\text{Element Damköhler:} \qquad \text{Da}_h &= \frac{sh}{a}, \\
\text{Element Kinetic number:} \qquad \text{Ki}_h &= \frac{sh^2}{2\varepsilon} = \text{Pe}_h\text{Da}_h,
\end{aligned}$$

and

$$A_h = \sqrt{\text{Pe}_h^2 + 2\text{Pe}_h\text{Da}_h} = \sqrt{\text{Pe}_h^2 + 2\text{Ki}_h}$$

The solutions for the different cases are:

$$\begin{aligned}
\text{ADR:} \quad \tau(x) &= \frac{h}{a}\frac{1}{\mathrm{Da}_h}\left(-\frac{e^{\mathrm{Pe}_h\, x/h}}{\sinh(A_h)}[e^{-\mathrm{Pe}_h}\ \sinh(A_h\,\tfrac{x}{h}) + \sinh(A_h\,(1-\tfrac{x}{h}))] + 1\right), \\
\text{AD:} \quad \tau(x) &= \frac{h}{a}\left(\frac{x}{h} - \frac{e^{2\mathrm{Pe}_h\, x/h}-1}{e^{2\mathrm{Pe}_h}-1}\right), \\
\text{AR:} \quad \tau(x) &= \frac{h}{a}\frac{1}{\mathrm{Da}_h}(1-e^{-\mathrm{Da}_h x/h}), \\
\text{DR:} \quad \tau(x) &= \frac{1}{s}\left(\frac{\sinh(\sqrt{2\mathrm{Ki}_h}\,x/h)(\cosh(\sqrt{2\mathrm{Ki}_h})-1)}{\sinh(\sqrt{2\mathrm{Ki}_h})} - \cosh(\sqrt{2\mathrm{Ki}_h}\,x/h) + 1\right).
\end{aligned}$$

The stabilization parameter τ has dimension of time. It is therefore convenient to introduce the dimensionless parameter α such that τ can be written as follows:

$$\begin{aligned}
\text{ADR, AD, AR:} \quad \tau &= \frac{h}{2a}\alpha(\mathrm{Pe}_h, \mathrm{Da}_h), \\
\text{DR:} \quad \tau &= \frac{1}{s}\alpha(\mathrm{Ki}_h).
\end{aligned} \tag{2.31}$$

Table 2.2 gives the results of the expression for α obtained with the strategy proposed together with the expression proposed by Codina and Shakib. Figure 2.3 shows the dependence of α upon Pe_h and Da_h for the ADR equation. The asymptotic behaviors of the three methods are similar although they slightly differ in some special cases. These special cases will be examined in the example presented in the next section. Note that with the expression for the ADR and AR equations, the same results as Hauke are obtained, see Hauke & García-Olivares (2001) (although the expression for ADR presented here has been rearranged).

Remark 1: Advection-diffusion equation for linear elements. Let us take a look at the advection-diffusion equation ($s = 0$) in one dimension. If linear elements are used, then both $R(u_h)$ and $L^*(v_h)$ are constant and do not include the diffusion term (in this case, the ASGS method is equivalent to the SUPG method). Then the expression $\tilde{u}(x) = \tau(x)R(u_h)$ is exact, with $\tau(x)$ solution of Equation (2.26). The subgrid scale term of the grid scale equation

	Codina	Shakib	Present work
ADR	$\frac{\mathrm{Pe}_h}{\mathrm{Pe}_h+\mathrm{Pe}_h\mathrm{Da}_h/2+1}$	$\frac{\mathrm{Pe}_h}{\sqrt{\mathrm{Pe}_h^2+(\mathrm{Pe}_h\mathrm{Da}_h/2)^2+9}}$	$\frac{2}{\mathrm{Da}_h}\left[\frac{A_h}{\mathrm{Pe}_h\mathrm{Da}_h}\left(\frac{\cosh(\mathrm{Pe}_h)-\cosh(A_h)}{\sinh(A_h)}\right)+1\right]$
AD	$\frac{\mathrm{Pe}_h}{\mathrm{Pe}_h+1}$	$\frac{\mathrm{Pe}_h}{\sqrt{\mathrm{Pe}_h^2+9}}$	$\coth(\mathrm{Pe}_h)-\frac{1}{\mathrm{Pe}_h}$
AR	$\frac{1}{\mathrm{Da}_h/2+1}$	$\frac{1}{\sqrt{(\mathrm{Da}_h/2)^2+1}}$	$\frac{2}{\mathrm{Da}_h^2}\left[\mathrm{Da}_h-1+e^{-\mathrm{Da}_h}\right]$
DR	$\frac{1}{2/\mathrm{Ki}_h+1}$	$\frac{1}{\sqrt{(6/\mathrm{Ki}_h)^2+1}}$	$2\frac{1-\cosh(\sqrt{2\mathrm{Ki}_h})}{\sqrt{2\mathrm{Ki}_h}\sinh(\sqrt{2\mathrm{Ki}_h})}+1$

Table 2.2: Different expressions for α.

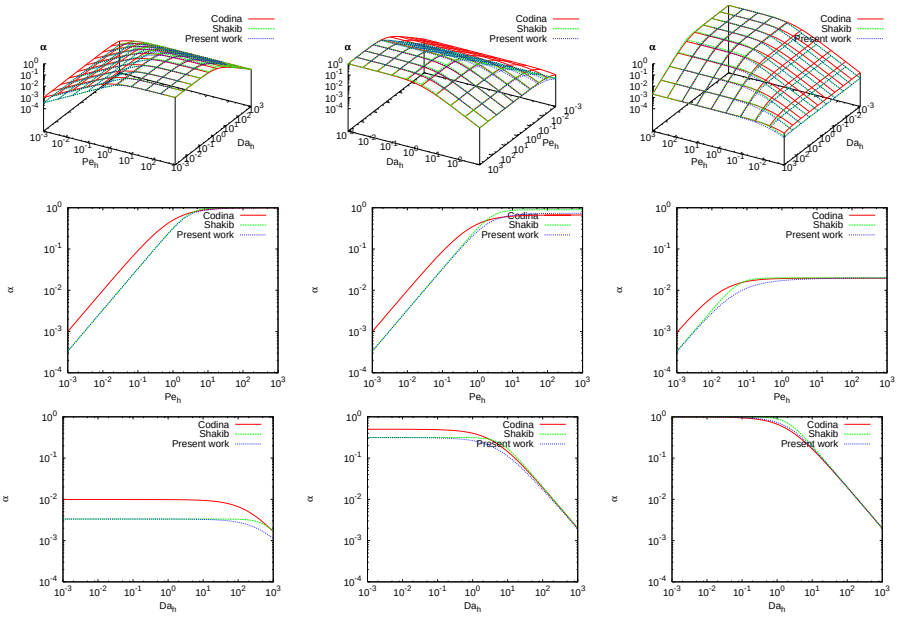


Figure 2.3: α for the ADR equation. (Top) Different Views. (Mid.) As a function of Pe_h for $\mathrm{Da}_h = 10^{-2}, 10^0, 10^2$ from left to right. (Bot.) As a function of Da_h for $\mathrm{Pe}_h = 10^{-2}, 10^0, 10^2$ from left to right.

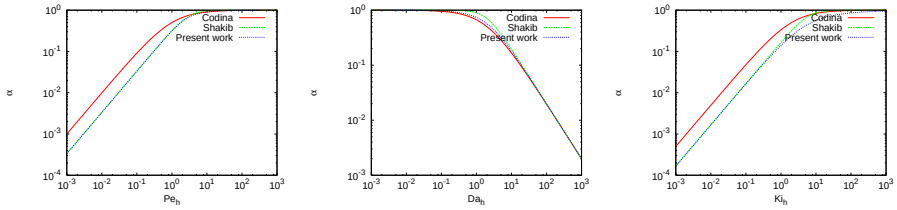


Figure 2.4: α for different equations. (Left) AD. (Mid.) AR. (Right) DR.

becomes for an element $\Omega_K = [0, h]$:

$$\begin{aligned}
\int_{\Omega_K} \tilde{u} L^*(v_h) d\Omega &= \int_0^h L^*(v_h)\tau(x)R(u_h)dx, \\
&= L^*(v_h)R(u_h)\int_0^h \tau(x)dx, \\
&= L^*(v_h)R(u_h)h\overline{\tau}, \\
&= \int_0^h L^*(v_h)\overline{\tau}R(u_h)dx.
\end{aligned}$$

This means that, for this particular case, the solution u_h is nodally exact using $\overline{\tau}$ in the integral instead of $\tau(x)$. In fact we recognize that the expression for $\overline{\tau}$ is the one usually used in the litterature. In addition, we note that $u_h + \tilde{u} = u_h + \tau(x)R(u_h)$ is exact in every element Ω_K.

Remark 2: What is h for quadratic elements? Let us consider the case of quadratic elements. As in the case of linear elements, we ask for the solution to be nodally exact. Then we take as a stabilization parameter $\overline{\tau}_2$ so that

$$\overline{\tau}_2 = \frac{1}{h/2}\int_0^{h/2} \tau_2(x)dx,$$

where $\tau_2(x)$ is solution of

$$L(\tau_2(x)) = 1 \text{ in } [0, h/2], \text{ with } \begin{cases} \tau_2(x) \mid_{0,h/2} = 0 \text{ for ADR, AD, DR,} \\ \tau_2(x) \mid_0 = 0 \text{ for AR,} \end{cases}$$

and, equivalently,

$$\overline{\tau}_2 = \frac{1}{h/2}\int_{h/2}^{h} \tau_2(x)dx,$$

where

$$L(\tau_2(x)) = 1 \text{ in } [h/2, h], \text{ with } \begin{cases} \tau_2(x) \mid_{h/2,h} = 0 \text{ for ADR, AD, DR,} \\ \tau_2(x) \mid_0 = 0 \text{ for AR.} \end{cases}$$

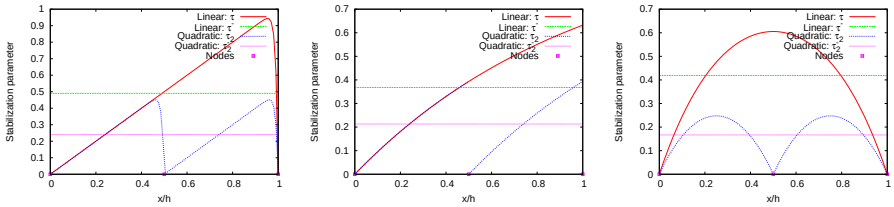


Figure 2.5: Comparisons of the stabilization parameters for a linear and a quadratic element $[0, h]$. (Left) AD. (Mid.) AR. (Right) DR.

For linear elements we have $\overline{\tau} = \overline{\tau}(h)$. We can easily check that $\overline{\tau}_2(h) = \overline{\tau}(h/2)$. Therefore, the subgrid scale term in the grid scale equation for an element Ω_K reads:

$$\begin{aligned}
\int_{\Omega_K} L^*(v_h)\tilde{u}d\Omega &= \int_0^{h/2} L^*(v_h)\overline{\tau}_2(h)R(u_h)dx \; + \int_{h/2}^{h} L^*(v_h)\overline{\tau}_2(h)R(u_h)dx, \\
&= \int_0^{h/2} L^*(v_h)\overline{\tau}(h/2)R(u_h)dx + \int_{h/2}^{h} L^*(v_h)\overline{\tau}(h/2)R(u_h)dx, \\
&= \int_0^{h} L^*(v_h)\overline{\tau}(h/2)R(u_h)dx.
\end{aligned}$$

Therefore, for quadratic elements, the expression of $\overline{\tau}$ given by Equation (2.31) with α given by Table 2.2 holds with h substituted by $h/2$. Figure 2.5 illustrates the case of the AD ($\varepsilon = 0.01$, $a = 1$), AR ($a = 1$, $s = 1$) and DR ($\varepsilon = 0.1$, $s = 1$) equations, all with a unit right-hand side.

2.1.2 Stabilized finite element formulation

The stabilized finite element formulation consists in substituting the equation for the subgrid scale (2.27), together with Equation (2.31), into Equation (2.23). However we need a further approximation concerning the boundary term in the latter equation. We saw in the last section that for the one-dimensional problem, we chose the subgrid scale to vanish on the node: thus the boundary term vanishes exactly. In multi-dimensional problems, Codina, using a Fourier analysis, showed that the subgrid scale tends to zero on the element boundary.

$$a(u_h, v_h) + \int_{\Omega'} L^*(v_h)\overline{\tau}R(u_h)\, d\Omega = l(v_h).$$

	ε	a	s
ADR1	5	1	10^3
ADR2	5×10^{-2}	1	10^2
ADR3	5×10^{-3}	1	10^1
AD	5×10^{-2}	1	0
AR	0	1	10^2
DR	10^{-3}	0	1

Table 2.3: Numerical example. Cases and coefficients.

By rearranging the terms, this equation can be rewritten in terms of the equation residual as:

$$\int_\Omega -R(u_h)[v_h - \overline{\tau} L^*(v_h)]\, d\Omega + \int_\Omega \varepsilon \nabla u_h \cdot \nabla v_h \, d\Omega + \int_\Omega \varepsilon \Delta u_h v_h \, d\Omega$$
$$-\flat \int_\Omega [(\boldsymbol{a} \cdot \nabla u_h) v_h + u_h (\boldsymbol{a} \cdot \nabla v_h) + (\nabla \cdot \boldsymbol{a}) v_h]\, d\Omega = \int_{\Gamma_N} g v_h \, d\Omega.$$

2.1.3 Numerical Example

We solve the one-dimensional problem on the domain $[0, 1]$. Both linear and quadratic elements are compared using 10 elements and 5 elements ($h = 0.2$), respectively. The cases have been chosen using Figures 2.3, 2.4 to select values of Pe_h and Da_h for which the α's give different values. The data are given in Table 2.3. For all of them the source term is $f = 1$, and the boundary conditions are $u(0) = u(1) = 0$, except for the AR case for which only $u(0) = 0$ is prescribed.

Figures 2.6 and 2.7 show the results obtained. Note that for quadratic elements, the solution is drawn as if it was linear within each element, as we are only interested in nodal values. As a general observation, we can state that the stabilization parameter presented here never gives the worst result. For ADR1, we obtain results similar to those of Shakib for both linear and quadratic elements. The results of ADR2 are better than those of Codina and Shakib. The latter give very bad results in the quadratic case. For the ADR3 case, Codina obtains a slightly better solution. The advection-diffusion equation gives nodally exact results for the presented strategy, as explained in Section 2.1.1. They are very similar to those of Shakib. In the case of the AR equation, an overshoot of 50% is obtained with all the method. It is somewhat lesser for quadratic elements. Finally, the present stabilization parameter based on exact solution gives the better results for the DR method.

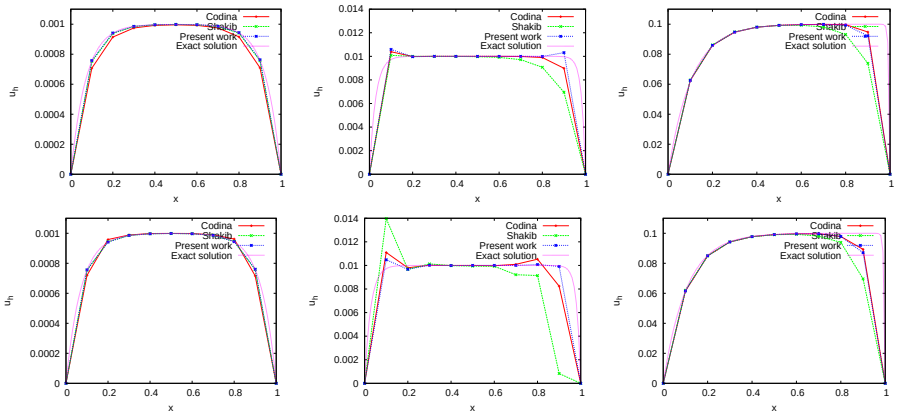


Figure 2.6: 1D problem: numerical solutions compared to the exact solution for linear and quadratic elements. (Top) Linear element. (Bot.) Quadratic element. (Left) ADR1 (Mid.) ADR2. (Right) ADR3.

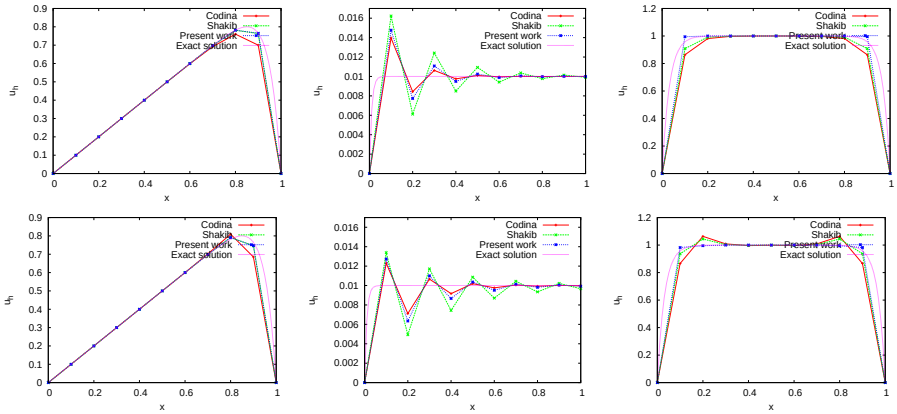


Figure 2.7: 1D problem: numerical solutions compared to the exact solution for linear and quadratic elements. (Top) Linear element. (Bot.) Quadratic element. (Left) AD. (Mid.) AR. (Right) DR.

2.2 Incompressible Navier-Stokes Equations

The equations for an incompressible flow will be described. Next, the variational and stabilized form will be presented as well as the time discretization strategy. Finally we will explain the Navier-Stokes solver used in this work and the strategy to transform the solution process into a *fractional step*.

2.2.1 Governing Equations

On the basis of the second law of motion, the Navier-Stokes equations are:

$$\begin{aligned} \rho\partial_t \boldsymbol{u} + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - 2\nabla\cdot(\mu\boldsymbol{\varepsilon}(\boldsymbol{u})) + \nabla p &= \rho\boldsymbol{f}, && (2.32)\\ \nabla\cdot\boldsymbol{u} &= 0, && (2.33) \end{aligned}$$

where $\boldsymbol{u}$ is the velocity of the fluid and p its pressure; μ is the dynamic viscosity and ρ the density considered to be constant over the computational domain and insensitive to pressure variations. $\boldsymbol{\varepsilon}(\boldsymbol{u})$ is the rate of deformation tensor given by

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^t)$$

Finally $\boldsymbol{f}$ is the vector of body forces (for example, gravity). A detailed derivation of these continuum equations of the fluid motion can be found in Batchelor (1970).
The Navier-Stokes equations are solved in a domain Ω of dimension n_d together with appropiate boundary conditions on the contour $\Gamma : \delta\Omega$. For example,

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_g \quad \text{on } \Gamma_D \times (0,T),\\ \sigma\cdot\boldsymbol{n} &= \boldsymbol{t}_n \quad \text{on } \Gamma_N \times (0,T),\\ \boldsymbol{u} &= \boldsymbol{u}_0 \quad \text{on } \Omega \times 0, \end{aligned}$$

where $\Gamma = \Gamma_D \cup \Gamma_N$, $\mathbf{n}$ is the outward unit normal and σ is the stress tensor

$$\sigma = -p\mathbf{I} + 2\mu\varepsilon(\boldsymbol{u})$$

being $\mathbf{I}$ the n_d-dimensional identity.
Let $\mathbf{U} := [\boldsymbol{u}, p]^t$ and define the differential operator $L(\mathbf{U})$ and force term $\mathbf{F}$ as

$$L(\mathbf{U}) := \begin{bmatrix} \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - 2\nabla\cdot(\mu\boldsymbol{\varepsilon}(\boldsymbol{u})) + \nabla p \\ \nabla\cdot\boldsymbol{u} \end{bmatrix} \quad (2.34)$$

$$\mathbf{F} := \begin{bmatrix} \rho\boldsymbol{f} \\ 0 \end{bmatrix} \quad (2.35)$$

We have to introduce the matrix $\mathbf{M}$ so that

$$\mathbf{M} = diag(\rho\mathbf{I}, 0)$$

So using this notation, the Navier-Stokes equation presented before can be written in a compact form as:

$$\mathbf{M}\partial_t\mathbf{U} + L(\mathbf{U}) = \mathbf{F} \quad (2.36)$$

2.2.2 Stabilized finite element formulation

We now derive the variational formulation of our problem. Let us introduce the following functional spaces:

$$\begin{aligned} V &= \boldsymbol{v}\epsilon H^1(\Omega)^{n_d} | \boldsymbol{v}_{|\Gamma_D} = 0\}, \\ Q &= L^2(\Omega), \\ U &= \boldsymbol{v}\epsilon H^1(\Omega)^{n_d} | \boldsymbol{v}_{|\Gamma_D} = \boldsymbol{u}_g\}, \\ P &= \left\{ p\epsilon L^2(\Omega) \middle| \int_\Omega p d\Omega = 0 \text{ if } \Gamma_N = \emptyset \right\} \end{aligned}$$

The first step to solve the Navier-Stokes equations is to linearise them. Let us denote by m the iteration number of the iterative scheme. For the sake of clarity, we only consider here the Picard linearization, that is,

$$[(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}]^{m+1} \approx (\boldsymbol{u}^m\cdot\nabla)\boldsymbol{u}]^{m+1}$$

The variational formulation of the problem reads as follows according to the decomposition of the advection term done in Equation 2.8. Given $\boldsymbol{u}^0\epsilon U$, for $m = 0, 1...$ until convergence, find $(\boldsymbol{u}^{m+1}, p^{m+1})\epsilon U \times P$ so that

$$a^m(\boldsymbol{u}^{m+1}, \boldsymbol{v}) - b(p^{m+1}, \boldsymbol{v}) + b(q, \boldsymbol{u}^{m+1}) = l(\boldsymbol{v}) \tag{2.37}$$

for all $(\boldsymbol{v}\times q)\epsilon V \times Q$, where

$$\begin{aligned} a^m(\boldsymbol{u}, \boldsymbol{v}) &= 2\int_\Omega \mu\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) d\Omega + (1-\flat)\int_\Omega [\rho(\boldsymbol{u}^m\cdot\nabla)\boldsymbol{u}]\cdot\boldsymbol{v} d\Omega \\ &\quad -\flat\int_\Omega \rho[(\boldsymbol{u}^m\cdot\nabla)\boldsymbol{v}]\cdot\boldsymbol{u} d\Omega + \flat\int_{\Gamma_N}\rho(\boldsymbol{u}^m\cdot\boldsymbol{n})\boldsymbol{u}\cdot\boldsymbol{v} d\Gamma, \\ b(p, \boldsymbol{v}) &= \int_\Omega p\nabla\cdot\boldsymbol{v} d\Omega \\ l(\boldsymbol{v}) &= \int_\Omega \rho\boldsymbol{f}\cdot\boldsymbol{v} d\Omega + \int_{\Gamma_N}\rho\boldsymbol{t}_n\cdot\boldsymbol{v} d\Gamma \end{aligned}$$

It is well-known that the latter formulation can lack stability for different reasons. One is related to the compatibility of the finite element spaces for the velocity and the pressure which have to satisfy the so-called Ladyzhenskaya-Brezzi-Babuska condition, see Fortin & Brezzi (1991). This condition is necessary in order to control pressure field. Another reason for the lack of stability is related to the relative importance of the viscous and advective effects. It can be directly related to the instabilities caused by high advection in the case of the ADR equation, as studied in section 2.1. The stabilized technique applied to this Navier-Stokes equation is based on the VMS method presented before, the ASGS.

Let $\boldsymbol{a}$ be the convection velocity known from the previous iteration (coming from the Picard linearization method). And let us decompose the Navier-Stokes differential operator L defined in 2.34 into two components L_1 and L_2 so that $L = L_1 + L_2$ with

$$L_1(\mathbf{U}) := \begin{bmatrix} \rho(\boldsymbol{a}\cdot\nabla)\boldsymbol{u} \\ \nabla\cdot\boldsymbol{u} \end{bmatrix} \tag{2.38}$$

$$L_2(\mathbf{U}) := \begin{bmatrix} -2\nabla\cdot(\mu\boldsymbol{\varepsilon}(\boldsymbol{u})) + \nabla p \\ 0 \end{bmatrix} \tag{2.39}$$

where L_2 represents the part of the operator that is integrated by parts. The stabilized formulation reads:

$$(\mathbf{M}\partial_t\mathbf{U}_h + L_1(\mathbf{U}_h), \mathbf{V}_h) + (2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})_h, \boldsymbol{\varepsilon}(\boldsymbol{v})_h) - (p_h, \nabla\cdot\boldsymbol{v}_h) \tag{2.40}$$

$$+(\mathbf{M}\partial_t\tilde{\mathbf{U}}, \mathbf{V}_h) + (\tilde{\mathbf{U}}, L^*(\mathbf{V}_h)) = L(\mathbf{V}_h) \tag{2.41}$$

In this equation $\mathbf{V}_h := [\boldsymbol{v}_h, q_h]^t$, $\boldsymbol{v}_h$ and q_h being the velocity and pressure test functions, respectively. For the right-hand side we have to taken in an appropiate discrete space; $L(\mathbf{V}) := (\rho\boldsymbol{f}, \boldsymbol{v})$. The subgrid scale $\tilde{\mathbf{U}}$ is computed element-wise by solving the following equation:

$$\mathbf{M}\partial_t\tilde{\mathbf{U}} + \boldsymbol{\tau} = \mathbf{R}(\mathbf{U}_h),$$
$$\mathbf{R}(\mathbf{U}_h) := \mathbf{F} - \mathbf{M}\partial_t\mathbf{U}_h - L(\mathbf{U}_h),$$

where τ is a square matrix such that

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1\mathbf{I} & 0 \\ 0 & \tau_2 \end{bmatrix} \tag{2.42}$$

and τ_1 and τ_2 are given byCodina (2001)

2.2.3 Time discretization

Let us introduce a uniform partition of the time interval $[0, T]$ and define

$$\begin{aligned} \boldsymbol{u}^{n+\theta} &:= \theta\boldsymbol{u}^{n+1} + (1-\theta)\boldsymbol{u}^n, \\ \delta t &:= t^n - t^{n-1}, \\ \delta_t\boldsymbol{u}^{n+\theta} &:= \frac{\boldsymbol{u}^{n+\theta} - \boldsymbol{u}^n}{\theta\delta t} \end{aligned}$$

According to this integration rule, the time-discretized Navier-Stokes equations are solved as follows. Given an initial condition $\boldsymbol{u}^0$, find $\boldsymbol{u}^{n+1}$ and p^{n+1} for each $n \geq 0$ so that

$$\begin{aligned} \rho\delta_t\boldsymbol{u}^{n+\theta} + \rho(\boldsymbol{u}^{n+\theta}\cdot\nabla)\boldsymbol{u}^{n+\theta} - 2\mu\nabla\cdot\boldsymbol{\varepsilon}(\boldsymbol{u}^{n+\theta}) + \nabla p^{n+\theta} &= \rho\boldsymbol{f}^{n+\theta} \quad \text{in} \quad \Omega \\ \nabla\cdot\boldsymbol{u}^{n+\theta} &= 0 \quad \text{in} \quad \Omega \end{aligned}$$

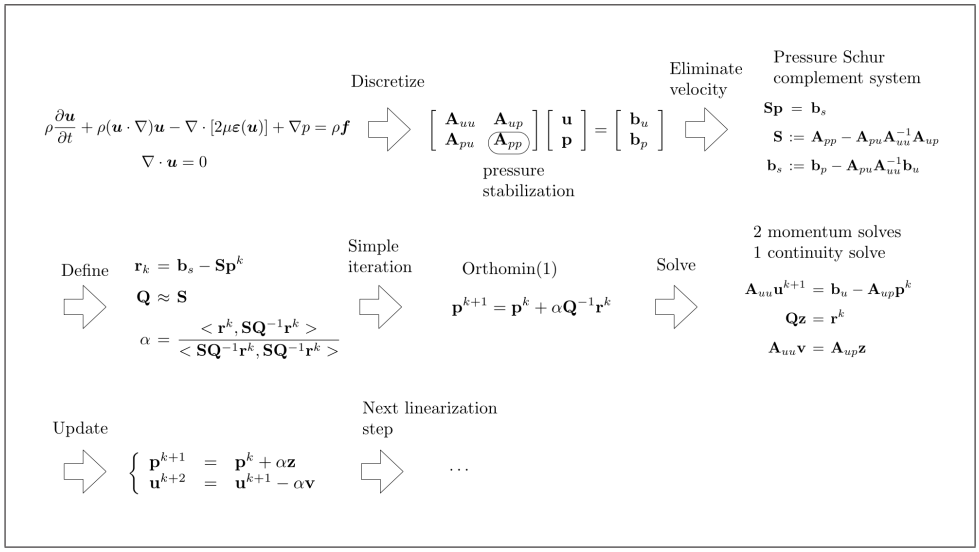


Figure 2.8: Algebraic split strategy. Orthomin(1) method applied to the pressure Schur complement system.

2.2.4 Navier-Stokes solver and algebraic split strategy

At each time step, the linearized system

$$\begin{bmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{bmatrix} \tag{2.43}$$

must be solved. Four sub-matrices arise. Matrix $\mathbf{A}_{uu}$ includes the Galerkin as well as the stabilization terms, like the SUPG-like term and the continuity enforcing term. Matrix $\mathbf{A}_{up}$ includes the stabilization terms and the Galerkin pressure gradient term. Matrix $\mathbf{A}_{pu}$ includes the velocity divergence operator as well as the part of the pressure stabilization involving the velocity in the momentum residual. Finally, matrix $\mathbf{A}_{pp}$ includes only the pressure stabilization. Note that this sub-matrix is null if div-stab elements are used. When this system is solved in one blow using either a direct solver or an iterative solver with preconditioning, the resulting scheme is referred to as a *monolithic scheme*. The next section briefly explains the split strategy for transforming the solution process into a *fractional scheme*.

In this section, we will obtain an algorithm to solve Equation 2.43. The steps are summed up in Figure 2.8. In this subsection, we shall limit ourselves to briefly describing the algebraic split strategy used to solve the linearized system. Although the complete development of the algorithm can be found inHouzeaux, Aubry, & Vázquez (2011), we shall nevertheless give some details.

Pressure Schur complement system. Let us manipulate the matrix system 2.43 to compute the Schur complement system, Golub & Loan (1996), for the pressure. The Schur complement system is the pressure equation one obtains after eliminating the velocity from the momentum equations:

$$\begin{aligned} \mathbf{Sp} &= \mathbf{b}_s, \quad \text{with,} && (2.44)\\ \mathbf{S} &= \mathbf{A}_{pp} - \mathbf{A}_{pu}\mathbf{A}_{uu}^{-1}\mathbf{A}_{up},, && (2.45)\\ \mathbf{b}_s &= \mathbf{b}_p - \mathbf{A}_{pu}\mathbf{A}_{uu}^{-1}\mathbf{b}_u. && (2.46) \end{aligned}$$

Preconditioned Orthomin(1) iteration. The idea is now to apply a relaxed preconditioned Richardson iteration (also referred to as a *simple iteration*) method, Golub & Loan (1996); Greenbaum (1997), to solve Equation 2.44. Let $\boldsymbol{r}^k$ be the residual of the Schur complement system at iteration k: $\mathbf{r}_k = \mathbf{b}_s - \mathbf{S}\mathbf{p}^k$.

We introduce a relaxation parameter α so that the simple preconditioned iteration reads:

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \alpha\mathbf{Q}^{-1}\mathbf{r}^k. \tag{2.47}$$

The preconditioner $\mathbf{Q}$ should approximate $\mathbf{S}$ defined in Equation 2.45. When α is constant and $\alpha < 1$, it is referred to as under-relaxation parameter. When α is constant and $\alpha > 1$, it is referred to as the over-relaxation parameter. The Orthomin scheme consists in choosing α in a dynamic way. To do so, let us multiply Equation 2.47 by $\mathbf{S}$ and add $-\mathbf{b}_s$ on both sides of the resulting equation. We obtain

$$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha\mathbf{S}\mathbf{Q}^{-1}\mathbf{r}^k. \tag{2.48}$$

Minimizing $\|\mathbf{r}^{k+1}\|^2$ we obtain the following equation for α:

$$\alpha = \frac{< \mathbf{r}^k, \mathbf{S}\mathbf{Q}^{-1}\mathbf{r}^k >}{< \mathbf{S}\mathbf{Q}^{-1}\mathbf{r}^k, \mathbf{S}\mathbf{Q}^{-1}\mathbf{r}^k >}. \tag{2.49}$$

The resulting scheme is shown in Algorithm 1. At each iteration k the residual is measured as:

$$\text{Continuity equation residual} = \frac{\|\mathbf{b}_p - \mathbf{A}_{pu}^k\mathbf{u}^k - \mathbf{A}_{pp}^k\mathbf{p}^k\|}{\|\mathbf{b}_p - \mathbf{A}_{pu}^k\mathbf{u}^k\|} \tag{2.50}$$

We can check that this algorithm is momentum-preserving. It involves two momentum solves, that is one more than classical fractional step techniques. In addition, we can derive from it a continuity-preserving version. Figure 2.9 compares the convergence of the Orthomin(1) with that of the continuity-preserving Richardson method (which has close similarities with classical fraction step methods). The example is a transient LES simulation for which we show the first five time steps. An evident gain in convergence is obtained by

Algorithm 1 Momentum preserving Orthomin(1) iteration

1. Solve momentum eqn $\mathbf{A}_{uu}\mathbf{u}^{k+1} = \mathbf{b}_u - \mathbf{A}_{up}\mathbf{p}^k$.
2. Compute Schur complement residual $\mathbf{r}^k = [\mathbf{b}_p - \mathbf{A}_{pu}\mathbf{u}^{k+1}] - \mathbf{A}_{pp}\mathbf{p}^k$.
3. Solve continuity eqn $\mathbf{Q}\mathbf{z} = \mathbf{r}^k$.
4. Solve momentum eqn $\mathbf{A}_{uu}\mathbf{v} = \mathbf{A}_{up}\mathbf{z}$.
5. Compute $\mathbf{x} = \mathbf{A}_{pp}\mathbf{z} - \mathbf{A}_{pu}\mathbf{v}$.
6. Compute $\alpha = < \mathbf{r}^k, \mathbf{x} > / < \mathbf{x}, \mathbf{x} >$.
7. Update velocity and pressure

$$\begin{cases} \mathbf{p}^{k+1} &= \mathbf{p}^k + \alpha\mathbf{z}, \\ \mathbf{u}^{k+2} &= \mathbf{u}^{k+1} - \alpha\mathbf{v}. \end{cases}$$

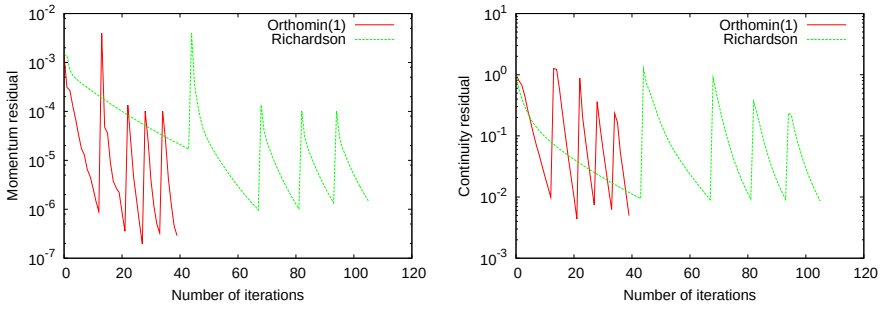


Figure 2.9: Comparison of convergences of Orthomin(1) and Richardson. (Left) Momentum. (Right) Continuity.

the Orthomin(1) method, even though it requires one additional momentum solve.

The Schur complement preconditioner Q. In the litterature, the preconditioner is usually based on the following splitting

$$\mathbf{Q} = \mathbf{A}_{pp} + \mathbf{P},$$

where $\mathbf{P}$ approximates the second term of the Schur complement matrix given by Equation 2.45 such that

$$\mathbf{P} \approx -\mathbf{A}_{pu}\mathbf{A}_{uu}^{-1}\mathbf{A}_{up}.$$

Let us introduce the momentum operator $\mathcal{M}$ (after first order time discretization) acting on the velocity as $\mathcal{M} = \rho/\delta t + \boldsymbol{u}\cdot\nabla - \nabla\cdot[2\mu\boldsymbol{\varepsilon}]$. If we identify $\mathbf{A}_{pu}$ with the divergence operator, $\mathbf{A}_{uu}$ with $\mathcal{M}$ and $\mathbf{A}_{up}$ with the gradient operator (that is, without considering the stabilization contribution

to the matrices), $\mathbf{P}$ can be computed from the weak form of Uzawa's operator $-\nabla \cdot \mathcal{M}^{-1}\nabla$ as described in Houzeaux et al. (2011). Using the so-called stabilization parameter τ, presented in the section 2.1.1 as an algebraic approximation of the inverse momentum operator, $\mathbf{P}$ is approximated by the following weak form

$$\mathbf{P} \Leftarrow \int_{\Omega} \tau \nabla p \cdot \nabla q d\Omega. \tag{2.51}$$

The important fact to note here is that the resulting preconditioner $\mathbf{Q}$ is symmetrical.

2.3 Turbulence Modelling

In 1937, Taylor and von Kárman proposed the following definition of turbuleuce:

> Turbulence is an irregular motion which in general makes its appearance in fluids, gaseous or liquid, when they flow past solid surfaces or even when neighboring streams of the same fluid flow past or over one another.

It is characterized by the presence of a wide range of excited length and time scales. The irregular nature of turbulence stands in contrast to laminar motion, so called historically, because the fluid was imagined to flow in smooth laminae, or layers. Virtually all flows of practical engineering interest are turbulent. Turbulent flows always occur when the inertial forces are relatively much more important than the viscous forces, that is for large Reynolds number, an adimensional parameter devoted to quantify the relative importance of these two types of forces for given flow conditions. Careful analysis of solutions to the Navier-Stokes equations, or more typically to its boundary-layer form, show that turbulence develops as an instability of laminar flow.

For a viscous fluid, the instabilities result from interaction between the Navier-Stokes equation's non-linear inertial terms and viscous terms. The interaction is very complex because it is strong rotational, fully three-dimensional (even if the original laminar flow or the initial disturbance is two-dimensional) and time-dependent (nevertheless, it is classified as stationary if the mean flow is time-independent).

It is observed that turbulent flows are always dissipative. Turbulence consists of a continuous spectrum of scales ranging from largest to smallest. In order to visualize a turbulent flow with a spectrum of scales, one often refers to turbulent eddies. A turbulent eddy can be thought of as a local swirling motion whose characteristic dimension is the local turbulence scale. Eddies are

generated in regions of high shear in the main flow field, i.e. near the solid wall or in the vicinity of interface between two streams flowing at different velocities. Eddies are three-dimensional. At any given time the flow contains eddies of various sizes. The size of the largest eddies is governed by the size of the flow and its geometry. The largest eddies break into smaller and smaller eddies until they are finally dissipated through viscosity (dissipation = friction loses). The largest eddies interact with the main flow and extract energy from it by a process called vortex stretching. Kinetic energy is transferred from large eddies to progressively smaller and smaller eddies in what is called the energy cascade. The smallest scale of motion which can occur in a turbulent flow is dictated by the viscosity. The energy associated with the smallest eddies is dissipated and converted into thermal energy. The transfer of energy from the main flow is expressed by the non-linear terms in the Navier-Stokes equations. These terms play a very important role in the generation and maintenance of turbulence.
The structure of the largest eddies is highly anisotropic and flow-dependent due to their strong interaction with the main flow. The diffusive action of viscosity tends to smear out directionality at small scales and therefore the smallest eddies are isotropic. A characteristic of turbulent flow, which results from the movement of eddies, is the greater ability for mixing or diffusion. Transportable quantities related to the flow - such as momentum, heat, sediment or pollutants - spread much more rapidly in turbulent than in laminar flow. The mixing motion resulting from fluctuations causes the viscosity to seem much greater than it actually is.

Additional stresses (known as Reynolds stresses) are developed in turbulent flows. Because of the large magnitude of the Reynolds stresses, there is much greater energy loss in turbulent than in laminar flows.

Turbulence modeling is one of three key elements in CFD. Very precise mathematical theories have evolved for the other two key elements, grid generation and algorithm development. Despite the many approaches and turbulence models that have been attempted since the eighties, no universal model has been devised. An ideal model should introduce the minimum amount of complexity while capturing the essence of the relevant physics but this is very difficult because an enormous amount of information is required to completely describe a turbulent flow.

Numerical approaches for turbulent flows. In contrast to laminar flow problems, numerical simulation of turbulent flows cannot be carried out by simply discretising the governing equations and solving them on a certain mesh. This is caused by the fact that turbulence is essentially three-dimensional and simultaneously contains many length scales. With increasing Reynolds number, the length scales of the smallest eddies in the flow become smaller and smaller.

Consequently, the amount of computational resources necessary to describe all the length scales that occur increases with the Reynolds number. Even the largest supercomputers do not have the required speed and memory capacity to handle this amount of data, except for turbulent flow with relatively low Reynolds.

In order to compute all significant motions of a turbulent flow, the domain on which the computation is performed must be at least as large as the largest eddy, and the mesh must be as fine as the smallest eddy.
The most important computational approaches developed up to now to simulate turbulent flows are:

- Direct Numerical Simulation (DNS);
- Large Eddy Simulation (LES);
- Detached Eddy Simulation (DES);
- Reynolds Averaged Navier-Stokes Models (RANS).

Although one can numerically solve the complete set of Navier-Stokes equations in three-dimensional space and time, as the first method presented above, the DNS method, because of large demand on computer resources, is limited to relatively low Reynolds and Rayleigh numbers and simple geometries.

For practical problems of industrial and environmental relevance, one of the most viable approaches is to consider RANS equations, the solution of which requires low-to-moderate computer resources that are affordable. However, RANS methods require some approximations, based on certain physical principles, known as "turbulence modelling".

A middle of the road between DNS and RANS is represented by the LES method, which resolves in space and time only the large-scale eddy motion, but employs models for subscale motion, usually defined in terms of numerical mesh size (hence called subgrid-scale models). LES can deal with higher Reynolds numbers and more complex geometries. However, LES is still quite demanding in terms of computer resources, closer to DNS than to RANS. This is because both DNS and LES always require solving the instantaneous Navier-Stokes equations in time and three-dimensional space, even if the time- or ensemble-averaged properties are steady and two- or one-dimensional, e.g. a steady fully developed flow in a pipe or in a jet.

Unlike DNS and LES, RANS can be used in steady form, and the problem can be solved only in two- and even one dimension if the average flow conditions satisfy these constraints.

Also, there have been many activities to combine LES and RANS in a hybrid manner. Because the resolution problem in LES is especially acute close to a solid surface where there are no large eddies and where proper numerical resolving of the small-scale eddy-structure requires very fine computational mesh, the hybrid approach uses RANS modelling in the near-wall region and LES solutions in the usually much larger flow region away from the wall.

As the method used in the simulations of CFD where the HERMESH method has been tested consists in RANS equations, it will be described in more detail below.

Reynolds Averaged Navier-Stokes Models: RANS. The quickest and computationally cheapest approach among those listed above is the Reynolds Averaged Navier-Stokes, based on ideas proposed by Reynolds over one century ago.

In this approach, each unknown variable is decomposed into a mean part $\bar{\phi}$ and a fluctuating part ϕ' , $\phi = \bar{\phi} + \phi'$. This decomposition is substituted in the Navier-Stokes equations and the equations are ensemble averaging over time. The nonlinearity of the equations gives rise to new terms (second moments) that make the set of equations be not closed (more unknowns than equations). This problem is known as the Turbulence Closure Problem. For this we need additional algebraic or differential relations. A set of mathematical equations which provide unknown variables, is called the Turbulence Closure Model.
The second moments are always vectors of higher order than the basic variables that complicate the closure problem. The type (algebraic or differential) and the number of auxiliary equations define the *closure level*. Two basic levels of modelling are currently used in computational fluid dynamics and transport processes: Eddy Viscosity/Diffusivity Models (EVM) (known also as the first-order models) and Second-Moment Closure Models (SMC) (known also as Reynolds stress/flux models or second-order models). Each category has a number of variants. The first-order models assume that the turbulent fluxes of momentum, heat and species are directly related to the mean flow field, i.e. mean velocity, mean temperature and mean concentrations, respectively. In the second-order models, the turbulent flux is obtained by solving separate differential transport equations for each flux component $\overline{\phi' u'_j}$.
As compared with DNS and LES, the RANS approach offers decisive computational advantages: in addition to dispensing with the need to solve the N-S equations in time and three-dimensional space for every problem, the RANS approach tolerates a much coarser computational grid and larger time steps if steady flow is sought than DNS and LES.
However, it should be borne in mind that the RANS approach is a drastic simplification within the description of turbulence. Statistical averaging brings

about a "loss of information" the consequence of which is the appearance of "superfluous" variables (higher moments). Providing these variables by Turbulence Models is always approximative (more or less, depending on the modelling level) and, consequently associated with some errors.
If we apply to the so-called Reynolds decomposition we obtain: $\mathbf{u} = \boldsymbol{u} + \boldsymbol{u}'$ and substituing it in Equation 2.32 we obtain the RANS equation:

$$\rho\partial_t \boldsymbol{u} + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - 2\nabla\cdot(\mu\varepsilon(\boldsymbol{u})) + \overline{\rho\boldsymbol{u}'\otimes\boldsymbol{u}'} + \nabla p = \rho\boldsymbol{f}; \tag{2.52}$$

where the fourth term represents the Reynolds stress term, $\boldsymbol{\tau}$ and it has to be modelled.
Two RANS models are described below since they are used in the resolved problems presented in this work, k-ε turbulence model and Spalart-Almaras turbulence model.

k-ε turbulence model. k-ε turbulence model is a two-equation model, which means that it includes two extra transport equations to represent the turbulent properties of the flow. This allows a two-equation model to account for history effects like convection and diffusion of turbulent energy.

$$\begin{aligned} \rho\partial_t k + \rho(\boldsymbol{u}\cdot\nabla k) - \nabla\cdot\big((\mu + \tfrac{\mu_t}{\sigma_\varepsilon})\nabla k\big) + \rho\varepsilon - P_k &= 0 \\ \rho\partial_t \varepsilon + \rho(\boldsymbol{u}\cdot\nabla\varepsilon) - \nabla\cdot\big((\mu + \tfrac{\mu_t}{\sigma_\varepsilon})\nabla\varepsilon\big) + \rho C_{\varepsilon 2}\varepsilon^2/k - C_{\varepsilon 1}\varepsilon/k P_k &= 0, \end{aligned} \tag{2.53}$$

with the following relations:

$$\begin{aligned} \mu_t &= C_\mu k^2/\varepsilon \\ P_k &= 2\mu_t \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u}), \end{aligned} \tag{2.54}$$

where k is the turbulent kinetic energy and ε the dissipation rate. μ_t is the eddy viscosity that needs to be modelled and P_k is the production term of kinetic energy, both expressed in 2.54. $C_\mu, C_{\varepsilon 1}, C_{\varepsilon 2}, \sigma_k$ and σ_ε are the $k-\varepsilon$ model constants.

Spalart-Almaras turbulence model. The Spalart–Allmaras model, Spalart & Allmaras (1992), is a one-equation model for turbulent viscosity. This model was devised "using empiricism and arguments of dimensional analysis, Galilean invariance, and selective dependence on molecular viscosity". It involves an eddy-viscosity variable $\tilde{\nu}$, related to the eddy-viscosity ν_t by:

$$\nu_t = \tilde{\nu}. \tag{2.55}$$

The high Reynold transport number equation for $\tilde{\nu}$ is:

$$\rho\partial_t\tilde{\nu} + \rho\boldsymbol{u}\cdot\nabla\tilde{\nu} = c_{b_1}\rho S\tilde{\nu} + \frac{1}{\sigma}\nabla\cdot[\rho(\nu+\tilde{\nu})\nabla\tilde{\nu}] + \frac{c_{b_2}}{\sigma}\rho(\nabla\tilde{\nu})^2 - c_{w_1}f_w\rho\frac{\tilde{\nu}^2}{d^2},$$

where d is the shortest distance to the wall and κ is the Von-Karman constant. The constants of the model are given later on. Equation (2.3) is not the original SA model. For the sake of clarity, some terms have been voluntarily omitted. The laminar region and transition cannot be simulated using the version presented previously; see the original publication of the authors for more information Spalart & Allmaras (1992).

The function f_w is given by

$$f_w = g\left[\frac{1+c_{w_3}^6}{g^6+c_{w_3}^6}\right]^{\frac{1}{6}}, \qquad \text{with} \tag{2.56}$$

$$g = r + c_{w_2}(r^6 - r), \qquad r := \frac{\tilde{\nu}}{S\kappa^2 d^2}. \tag{2.57}$$

The production term, the first term of the right-hand side of (2.3) involves the vorticity S given by

$$S = \sqrt{2\boldsymbol{\Omega}(\boldsymbol{u}) : \boldsymbol{\Omega}(\boldsymbol{u})}, \qquad \text{and} \tag{2.58}$$

$$\boldsymbol{\Omega}(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} - \nabla \boldsymbol{u}^t). \tag{2.59}$$

The values of the constants of the model are

$$\begin{aligned} &c_{b_1} = 0.1355, c_{b_2} = 0.622, \sigma = 0.667, \kappa = 0.41, \\ &c_{w_1} = c_{b_1}/\kappa^2 + (1 + c_{b_2})/\sigma, c_{w_2} = 0.3, c_{w_3} = 2.0. \end{aligned} \tag{2.60}$$

2.4 Level Set Equation

The Level Set method leads to a transport partial differential equation, presented in section 2.1, the solution of which determines the position of the free surface as an iso-value of the unknown of this equation. The basic idea of the level set method is to define a smooth scalar function, say $\psi(\boldsymbol{x}, t)$, over the computational domain Ω that determines the extent of subdomains Ω_1 and Ω_2. For instance, we may assign positive values to the points belonging to Ω_1 and negative values to the points belonging to Ω_2. The position of the fluid front will be defined by the iso-value contour $\psi(\boldsymbol{x}, t) = 0$. The evolution of the front $\psi = 0$ in any control volume $V_t \subset \Omega$ which is moving with a divergence free velocity field $\boldsymbol{u}$ leads to:

$$\partial_t \psi + (\boldsymbol{u} \cdot \nabla)\psi = 0. \tag{2.61}$$

Function ψ is the solution of the hyperbolic equation (2.61) with the boundary conditions:

$$\psi = \overline{\psi} \qquad \text{on } \Gamma_{\text{inf}} \times (t_0, t_f), \tag{2.62}$$

$$\psi(\boldsymbol{x}, 0) = \psi_0(\boldsymbol{x}), \tag{2.63}$$

where Γ_{inf} represents the inflow boundary and t_0, t_f are the initial and final time respectively. The initial condition ψ_0 is chosen in order to define the initial position of the fluid front to be analyzed. The boundary condition $\overline{\psi}$ determines which fluid enters through a certain point of the inflow boundary.

Due to the pure convective type of the equation for ψ, we use the SUPG technique for the spatial discretization. The temporal evolution is treated via the standard trapezoidal rule.

The space discrete problem using SUPG stabilization, mentioned in 2.1, for the Level Set problem consists in finding ψ_h in the appropriate space X_h such that

$$\int_\Omega \varphi_h \left(\partial_t \psi_h + (\boldsymbol{u}_h \cdot \nabla)\psi_h\right) \mathrm{d}\Omega + \sum_{k=1}^{n_{\text{el}}} \int_{\Omega^k} \tau \left(\boldsymbol{u}_h \cdot \nabla \varphi_h\right) \left(\partial_{\text{t}} \psi_{\text{h}} + (\mathrm{u}_{\text{h}} \cdot \nabla)\psi_{\text{h}}\right) \mathrm{d}\Omega = 0,$$

for all $\varphi_h \in V_h$, where V_h is an appropriate test function space. n_{el} is the number of elements in the mesh and Ω^e is the element domain. The stabilization parameter for the Level Set equation is calculated element-wise as

$$\tau = \frac{h}{2\,|\boldsymbol{u}_h|}.$$

where h is the characteristic length of the element. As in the Navier Stokes case, the minimum element length has been used for anisotropic meshes.

For the numerical solution of the level set equation, it is preferable to have a function without large gradients. Since the only requirement such a function must meet is $\psi = 0$ at the interface, a signed distance function ($|\nabla\psi| = 1$) is used. Under the evolution of the level set equation, ψ will not remain a signed distance function and thus needs to be reinitialized. This can be achieved by redefining ψ for each node of the finite element mesh according to the following expression:

$$\psi = \text{sgn}(\psi^0)d,$$

where ψ^0 stands for the calculated value of ψ, d is the distance from the node under consideration to the front, and $\text{sgn}(\cdot)$ is the sign of the value enclosed in the parenthesis.

In order to calculate the distance d we are currently using a geometrical method based on a skd-tree, see Khamayseh & Kuprat (2008). Computing the distance from a point to a surface mesh is a crucial issue in the implementation of the level set reinitialization. For each point where one wants to know the distance, it involves searching among all the triangular faces into which the surface is divided to find the one that gives the minimum distance. This search

can affect the performance of the whole system in a negative way: the time of simulation can grow considerably. To reduce the number of computations of this expensive task, we use geometric search structures to define the surface mesh. Our implementation uses a bounding volume hierarchy known as skd-tree. In these binary trees, each node has a set of faces of the surface mesh and their corresponding associated bounding volume. The implementation details for building these structures are described in Khamayseh & Kuprat (2008). This reference also describes a way to use the skd-trees to determine the distance between a point and the surface mesh. The idea is to minimize an upper bound on the distance between the point and the surface mesh while traversing the binary tree from the root node. This upper bound is computed as the distance to the farthest point in the bounding volume corresponding to the current node.

2.5 Solid Mechanics Equations

Solid mechanical problems consist in the simulation of a deformable body, which is subjected to some external forces or imposed displacements. Although a extensive reference in solid mechanics is Belytschko, Moran, & Liu (1999), we are going to describe the physical problem, two models of materials and space and time discretization implemented in this work.
Let $\boldsymbol{\varphi} : \mathbb{R}^3 \to \mathbb{R}^3$ be a function that maps a material point $\boldsymbol{X} \in B_0$ in the reference configuration to its corresponding point $\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{X}) \in B$ in the deformed configuration. The deformation gradient tensor $\boldsymbol{F}$ is defined as

$$\boldsymbol{F} := \operatorname{Grad} \boldsymbol{x} = \boldsymbol{\nabla}_0 \boldsymbol{x} \,, \tag{2.64}$$

where Grad or $\boldsymbol{\nabla}_0$ is gradient operator with respect to the reference configuration. In Cartesian basis, the above expression can be written in index notation as

$$F_{ij} = \frac{\partial x_i}{\partial X_j} \,. \tag{2.65}$$

Since $\boldsymbol{x}(\boldsymbol{X}) = \boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X})$, with $\boldsymbol{u}$ the displacement vector, thus the deformation gradient can be given by $\boldsymbol{F} = \boldsymbol{I} + \boldsymbol{\nabla}_0 \boldsymbol{u}$, where $\boldsymbol{\nabla}_0 \boldsymbol{u}$ is the displacement gradient and $\boldsymbol{I}$ is the second-order identity tensor. In components, $F_{ij} = \delta_{ij} + \partial u_i / \partial X_j$, where δ_{ij} is a Kronecker-delta function, *i.e.*, $\delta_{ij} = 1$ if $i = j$ or $\delta_{ij} = 0$ otherwise.

The equation of the balance of momentum with respect to the reference (or undeformed) configuration can be written as

$$\operatorname{Div} \boldsymbol{P} + \boldsymbol{b}_0 = \rho_0 \ddot{\boldsymbol{u}} \,, \quad \forall \boldsymbol{X} \in B_0 \,, \tag{2.66}$$

where ρ_0 is the mass density (per unit reference volume) and Div is the divergence operator with respect to the reference configuration, $\operatorname{Div} \boldsymbol{P} = \boldsymbol{\nabla}_0 \cdot \boldsymbol{P}$.

Using index notation, the above balance of momentum equation can be written as

$$\frac{\partial P_{ij}}{\partial X_j} + b_{0i} = \rho_0 \frac{\partial^2 u_i}{\partial t^2} \,, \quad \forall X_j \in B_0 \,, \tag{2.67}$$

In the above expression, tensor $\boldsymbol{P}$ and vector $\boldsymbol{b}_0$ stand for, respectively, the first Piola–Kirchhoff stress (or nominal/engineering stress) and the distributed body force on the undeformed body. The following boundary conditions are applied:

$$\boldsymbol{u}(\boldsymbol{X}) = \bar{\boldsymbol{u}} \,, \quad \forall \boldsymbol{X} \in \partial_u B_0 \,, \tag{2.68}$$

$$\boldsymbol{t}_0(\boldsymbol{X}) = \boldsymbol{P}\boldsymbol{n}_0(\boldsymbol{X}) = \bar{\boldsymbol{t}}_0 \,, \quad \forall \boldsymbol{X} \in \partial_t B_0 \,. \tag{2.69}$$

2.5.1 Linear isotropic elasticity

Linear isotropic elasticity, referred to as ISOLIN, is a finite deformation interpretation of the infinitesimal linear elasticity model, which can be writen as

$$\boldsymbol{S} = 2\mu \boldsymbol{E} + \mathrm{tr}(\boldsymbol{E})\lambda \, \boldsymbol{I} \,, \tag{2.70}$$

where $\boldsymbol{S}$ is the second Piola–Kirchhoff tensor (in the undeformed configuration), $\boldsymbol{E}$ is the Green–Lagrange strain tensor, and λ and μ are material parameters representing elastic Lame's constants. The second Piola–Kirchhoff tensor $\boldsymbol{S}$ is related to the first Piola–Kirchhoff stress tensor $\boldsymbol{P}$ as $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{F}^T$ and the Green–Lagrange strain tensor is obtained as $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I})$. Using index notation, the linear isotropic elasticity model ISOLIN can be written as

$$S_{ij} = 2\mu E_{ij} + \mathrm{tr}(\boldsymbol{E})\lambda \, \delta_{ij} \,, \tag{2.71}$$

where $P_{ij} = S_{ik}F_{jk}$, $E_{ij} = \frac{1}{2}(F_{ki}F_{kj} - \delta_{ij})$, $\mathrm{tr}(\boldsymbol{E}) = \sum_i E_{ii}$.

The fourth-order stiffness tangent corresponding to the above linear isotropic elasticity model ISOLIN can be expressed in index notation as

$$C_{ijkl} := \frac{\partial S_{ij}}{\partial E_{kl}} = \mu \left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) + \lambda \delta_{ij}\delta_{kl} \,. \tag{2.72}$$

Sometimes, material isotropic elasticity parameters are described using Young's modulus E and Poisson ratio ν. In this case, the Lame's constants λ and μ can be computed as

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2(1+\nu)} \,. \tag{2.73}$$

It is important here not to confuse the Young's modulus E with the components of the Green–Lagrange strain tensor E_{ij}.

2.5.2 Neo-Hookean hyperelasticiy model

In the Neo-Hookean hyperelasticity model, referred to as NEOHOOKE, the constitutive stress-strain relation is derived from an elastic potential, which is defined as

$$\psi = \frac{1}{2}\lambda \left(\ln(J)\right)^2 - \mu \left(\ln(J) - \mathrm{tr}(\boldsymbol{E})\right) , \tag{2.74}$$

with $J = \det(\boldsymbol{F})$. The second Piola–Kirchhoff stress is obtained from the derivative of the above elastic potential as follows:

$$\boldsymbol{S} := \frac{\partial \psi}{\partial \boldsymbol{E}} = \left(\lambda \ln(J) - \mu\right) \boldsymbol{C}^{-1} + \mu \boldsymbol{I} , \tag{2.75}$$

where $\boldsymbol{C}$ is the left Cauchy–Green tensor (in the undeformed configuration), computed as $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F} = 2\boldsymbol{E} - \boldsymbol{I}$. In index notation, the (isotropic) Neo-Hookean hyperelasticity model can be written as

$$S_{ij} = \left(\lambda \ln(J) - \mu\right) C^{-1}_{ij} + \mu\, \delta_{ij} , \tag{2.76}$$

where $C_{ij} = F_{ki}F_{kj} = 2E_{ij} + \delta_{ij}$.

In terms of index notation, the fourth-order stiffness tangent for the above Neo-Hookean hyperelasticity model can be written as

$$C_{ijkl} := \frac{\partial S_{ij}}{\partial E_{kl}} = \mu \left(C^{-1}_{ik}C^{-1}_{jl} + C^{-1}_{il}C^{-1}_{jk}\right) + \lambda C^{-1}_{ij}C^{-1}_{kl} . \tag{2.77}$$

2.5.3 Finite element formulation

The weak form of balance of the momentum equation (2.66) can be formulated for any arbitrary admissible virtual displacement $\boldsymbol{w}(\boldsymbol{X})$, such that,

$$\int_{B_0} \mathrm{Div}\, \boldsymbol{P} \cdot \boldsymbol{w}\, \mathrm{d}V + \int_{B_0} \boldsymbol{b}_0 \cdot \boldsymbol{w}\, \mathrm{d}V = \int_{B_0} \rho_0 \ddot{\boldsymbol{u}} \cdot \boldsymbol{w}\, \mathrm{d}V . \tag{2.78}$$

For an finite element approximation $\Omega_0 = \bigcup_e \Omega_0^e$ of the undeformed continuum body B_0, let $\boldsymbol{u}_h$ be a polynomial approximation of degree k to the actual displacement $\boldsymbol{u}$, such that

$$\boldsymbol{u}_h(\boldsymbol{X}) = \sum_a N^a(\boldsymbol{X})\, \boldsymbol{u}_a, \tag{2.79}$$

where $\boldsymbol{u}_a$ is the a-component of the nodal vector of $\boldsymbol{u}$ displacement and $\boldsymbol{u}_h(\boldsymbol{X})$ represents the interpolated displacement value.

The finite element formulation for the balance of momentum equation (2.78) thus reads as follows: find $\boldsymbol{u} \in \mathbb{R}^3$ so that for all elements $\Omega_0^e \in \Omega_0$,

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{f}_{int} = bm\mathbf{f}_{ext}, \tag{2.80}$$

where $\mathbf{M}$, $\mathbf{f}_{int}(\mathbf{u})$, and $\mathbf{f}_{ext}$ are, respectively, the mass matrix, the vector of internal and external forces. These quantities are constructed and assembled from the corresponding element quantities $\mathbf{M}^e$, $\mathbf{f}^e_{int}$ and $\mathbf{f}^e_{ext}$, which are the corresponding mass matrix, internal force vector and external force vectors. These elemental tensors are calculated using Gauss quadrature approximation in the isoparametric referential and are given (in indicial form) by:

$$\mathbf{M}^e_{iakb} = \sum_q w_q \rho_0 \delta_{ik} \mathbf{N}^e_a(\boldsymbol{\xi}_q)\mathbf{N}^e_b(\boldsymbol{\xi}_q)\mathbf{J}_0(\boldsymbol{\xi}_q), \tag{2.81}$$

$$\mathbf{f}^e_{int,ia} = \sum_q w_q \mathbf{P}_{i\mathbf{J}}(\boldsymbol{\xi}_q)\mathbf{N}^e_{a,\mathbf{J}}(\boldsymbol{\xi}_q)\mathbf{J}_0(\boldsymbol{\xi}_q), \tag{2.82}$$

$$\mathbf{f}^e_{ext,ia} = \sum_p w_p t_{0i}(\boldsymbol{\xi}_p)\mathbf{N}^e_a(\boldsymbol{\xi}_p)\mathbf{J}_0(\boldsymbol{\xi}_p) + \sum_q w_q b_{0i}(\boldsymbol{\xi}_q)\mathbf{N}^e_a(\boldsymbol{\xi}_q)\mathbf{J}_0(\boldsymbol{\xi}_q), \tag{2.83}$$

where δ_{ik} is the Kronecker symbol, $\boldsymbol{\xi}_q$ corresponds to the coordinates of Gauss point q in the isoparametric referential, w_q is its corresponding weight, and $\mathbf{J}_0$ is the Jacobian between the isoparametric referential and the reference configuration. The same notations with subscripts p (first term of Equation (2.83)) are for the surface elements belonging to Γ_{d0} mapped onto the corresponding faces of the volume elements.

As the internal force vector $\mathbf{f}^e_{int}$ is not necessarily linear in $\mathbf{u}$, a Newton-Raphson procedure can be used in conjunction with the time discretization scheme (see next Subsection 2.5.4). In this case, the elemental stiffness matrix $\mathbf{K}^e$ is needed:

$$\mathbf{K}^e_{iakb} = \frac{\partial \mathbf{f}^e_{int,ia}}{\partial \mathbf{u}_{kb}} = \sum_q w_q \frac{\partial \mathbf{P}_{i\mathbf{J}}}{\partial \mathbf{F}_{kL}}(\boldsymbol{\xi}_q)\mathbf{N}^e_{a,J}(\boldsymbol{\xi}_q)\mathbf{N}^e_{b,L}(\boldsymbol{\xi}_q)\mathbf{J}_0(\boldsymbol{\xi}_q), \tag{2.84}$$

where the tangent moduli $\frac{\partial \mathbf{P}_{i\mathbf{J}}}{\partial \mathbf{F}_{kL}}$ is provided by the constitutive law at each time step.

2.5.4 Time discretization

The generalized Newmark formulation used here for the balance of momentum equation (2.80) can be written as

$$\mathbf{M}\ddot{\mathbf{u}}_{n+1} + \mathbf{f}^{n+1}_{int} = \mathbf{f}^{n+1}_{ext}\,, \tag{2.85}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t \dot{\mathbf{u}}_n + (\Delta t)^2 \left\{ \left(\frac{1}{2} - \beta \right) \ddot{\mathbf{u}}_n + \beta \ddot{\mathbf{u}}_{n+1} \right\}, \tag{2.86}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + \Delta t \left\{ (1-\gamma)\, \ddot{\mathbf{u}}_n + \gamma \ddot{\mathbf{u}}_{n+1} \right\}, \tag{2.87}$$

where subscripts "n" and "$n+1$" indicate that the variables are evaluated at time t_n and t_{n+1}, respectively (the subscript "h" has been dropped for simplicity). Parameters β and γ set the characterictics of the Newmark scheme. Parameter $\beta = 0$ leads to an explicit Newmark scheme, whereas $0 < \beta \leq 0.5$ leads to an implicit scheme. In the latter case, the set of equations (2.85)–(2.87) are solved for unknown displacement $\mathbf{u}_{n+1}$, velocity $\dot{\mathbf{u}}_{n+1}$, and acceleration $\ddot{\mathbf{u}}_{n+1}$ using the iterative Newton-Raphson algorithm.

2.6 Manufactured Solution

The two main objectives of manufactured solution are: on the one hand, to test that the equation is well coded; on the other hand, to perform a mesh convergence test using a target solution $\boldsymbol{u}^{(e)}$, with a given degree of regularity. In Roache (1998) we find the definition of code verification, although this definition can differ between different authors. One of the earliest articles published related to this issue is Shih (1985).

We have applied the manufactured solution technique to prove the order of accuracy of the HERMESH method.
A manufactured solution is an exact solution to some PDE that has been constructed by solving the following problem: $\mathcal{L}(\boldsymbol{u}) = f$. The idea is to *manufacture* a solution $\boldsymbol{u_e}$ and then apply the differential operator $\mathcal{L}$ to this solution $\boldsymbol{u_e}$ to find the right hand side, f, so that $\mathcal{L}(\boldsymbol{u}) = \mathcal{L}(\boldsymbol{u_e})$. In addition, $\boldsymbol{u}_e$ is prescribed as a Dirichlet condition on the Dirichlet part of the boundary and the exact traction on the Neumann part of the boundary. We have proved that HERMESH method is exact if the solution belongs to the finite element space. That is, an up to linear solution is nodally exact.
The norms of the error considered in these tests are computed as follows:

$$\epsilon(\boldsymbol{u}) = \frac{\sqrt{\int_\Omega (\boldsymbol{u}_h - \boldsymbol{u}^{(e)})^2 d\Omega}}{\sqrt{\int_\Omega \boldsymbol{u}^{(e)^2} d\Omega}}, \tag{2.88}$$

$$\epsilon(\boldsymbol{\nabla} u) = \frac{\sqrt{\int_\Omega \sum_{ij} (\boldsymbol{\nabla}_j u_{i_h} - \boldsymbol{\nabla}_j {u_i}^{(e)})^2 d\Omega}}{\sqrt{\int_\Omega \sum_{ij} \boldsymbol{\nabla}_j {u_i}^{(e)^2} d\Omega}}, \tag{2.89}$$

where $\epsilon(\boldsymbol{u})$ is the L^2 norm of the relative error of the solution and $\epsilon(\boldsymbol{\nabla} u)$ of the gradient.

Manufactured solution for the ADR equation

In the case of advection-diffusion-reaction equation we seek for the solution of Equation 2.1 with the following f for the right hand side:

$$f = \mathcal{L}(u_e) := -\varepsilon \Delta u_e + \nabla \cdot (\boldsymbol{a} u_e) + s u_e. \tag{2.90}$$

If u_e belongs to the finite element space (up to bilinear), then the algorithm should give $u = u_e$.

Manufactured solution for the Navier-Stokes equations

To apply the manufactured solution to the Navier-Stokes system implies imposing both exact solution for velocity, $\boldsymbol{u}_e$ and pressure p_e with a given degree

of smoothness. Then we seek for this solution by adding two force terms $\boldsymbol{f}_m$ and f_c to the momentum and continuity equations, respectively:

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \nabla\cdot[2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})] + \nabla p = \boldsymbol{f}_m, \tag{2.91}$$

$$\nabla\cdot\boldsymbol{u} = f_c, \tag{2.92}$$

with

$$\boldsymbol{f}_m = \rho\frac{\partial \boldsymbol{u}_e}{\partial t} + \rho(\boldsymbol{u}_e\cdot\nabla)\boldsymbol{u}_e - \nabla\cdot[2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}_e)] + \nabla p_e, \tag{2.93}$$

$$f_c = \nabla\cdot\boldsymbol{u}_e, \tag{2.94}$$

In addition, $\boldsymbol{u}_e$ is prescribed as a Dirichlet condition on the Dirichlet part of the boundary and the exact traction on the Neumann part of the boundary. In addition, if the flow is confined, the pressure is defined up to a constant. In this case, the pressure is prescribed to p_e on one node of the domain. Note that if the manufactured solution belongs to the finite element space (linear in this case), the finite element solution is exactly the manufactured solution.

Manufactured solution for solid mechanics equation

In index notation, the equation we are solving is:

$$\rho\frac{\partial^2 u_i}{\partial t^2} - \frac{\partial P_{ij}}{\partial x_j} - b_{0,i} = 0. \tag{2.95}$$

In order to test the mesh convergence, we propose a manufactured solution $\boldsymbol{u}^{(e)}$ and solve the following equation:

$$\rho\frac{\partial^2 u_i}{\partial t^2} - \frac{\partial P_{ij}}{\partial x_j} = -\frac{\partial P^{(e)}_{ij}}{\partial x_j}. \tag{2.96}$$

If $\boldsymbol{u}^{(e)}$ belongs to the finite element space (up to bilinear), then the algorithm should give $\boldsymbol{u} = \boldsymbol{u}^{(e)}$.

We have:

$$\begin{aligned}
\frac{\partial P^{(e)}_{ij}}{\partial x_j} &= \left(\frac{\partial u^{(e)}_i}{\partial x_k} + \delta_{ik}\right)\frac{C_{kjml}}{2}\left(\frac{\partial^2 u^{(e)}_n}{\partial x_j \partial x_m}F^{(e)}_{nl} + \frac{\partial^2 u^{(e)}_n}{\partial x_j \partial x_l}F^{(e)}_{nm}\right) \\
&\quad + \frac{C_{kjml}}{2}\frac{\partial^2 u^{(e)}_i}{\partial x_k \partial x_j}\left(F^{(e)}_{nm}F^{(e)}_{nl} - \delta_{ml}\right), \\
F^{(e)}_{ml} &= \frac{\partial u^{(e)}_m}{\partial x_l} + \delta_{ml}.
\end{aligned}$$

2.7 Alya: Parallel Computational Mechanics code

Alya System, which was developed at the Barcelona Supercomputing Center (BSC-CNS), is a Computation Mechanics (CM) code with two main features. Firstly, it is specially designed to run with the highest efficiency standards on large-scale supercomputing facilities. Secondly, it is capable of solving different physics problems, each one with its own modelling characteristics, in a coupled way. These two main features are intimately related, which means that any complex coupled problems solved by Alya will still be solved efficiently. Alya's architecture is modular, grouping the different tasks into kernel, modules and services. The kernel, the core of Alya, contains all the facilities required to solve any set of discretized PDEs (e.g., the solver, the I/O, the coupling, the elements database, the geometrical information, etc.). Each module describes the physical description of a given problem (e.g, the discretized terms of the PDE, the meaning of the boundary and initial conditions, etc.). Finally, the services contain the tool boxes providing several independent procedures to be called by modules and kernel.
The parallelization of Alya is a service inside the code is based on a mesh partitioning technique performed by METIS (see the reference: METIS (2014) for more details) and uses a Master-Slave strategy. The main input data of METIS are the element graph and the weight of the vertices of the graph. The number of Gauss integration points is used to control the load balance of hybrid meshes related to the former aspect. The element graph is a crucial point in the application of the HERMESH method and will be discussed later on. To construct this element graph needed by METIS, two strategies are possible. On the one hand, an adjacent element to an element e can be considered as all the elements sharing a node with e. However, this graph can be quite memory- and time-consuming. On the other hand, the other strategy consists in taking as adjacent elements to e only the elements sharing a face with e. The latter option requires much less memory. As a reference for valuing the difference in memory consuming, for a regular hexahedra mesh the face connectivity criterion gives 6 neighbours while the node connectivity criterion gives 26 neighbours. The former strategy will be referred to as *by-faces* and the latter as *by-nodes*.
Based on the Master-Slave strategy, the Master is in charge of reading the mesh, of performing the partitioning and the output. The slaves are in charge of the construction of the local right-hand side (b_i) and the local matrices (A_i) and of the solution of the resulting system in parallel. In other words, each process will be in charge of each subdomain, which are the slaves. In the assembling tasks, no communication is needed between the slaves, and the scalability only depends on the load balancing. In the iterative solvers, the scalability depends on the size of the interfaces, which METIS minimizes, and the communication scheduling. All the details on the code parallelization can

be found in Houzeaux, Vázquez, Aubry, & Cela (2009).

Figure 2.10 is a schematic flowchart for the execution of a simulation using Alya. The tasks that the master process is responsible for are shown on the left side of the same figure with a grey background. As we have explained, it performs the first steps of the execution, namely reading the file and partitioning the mesh. Afterwards, the master sends the corresponding subdomain information to each worker process or slaves; then the master and the slaves enter the time and linearization loops, represented as one single loop. Along

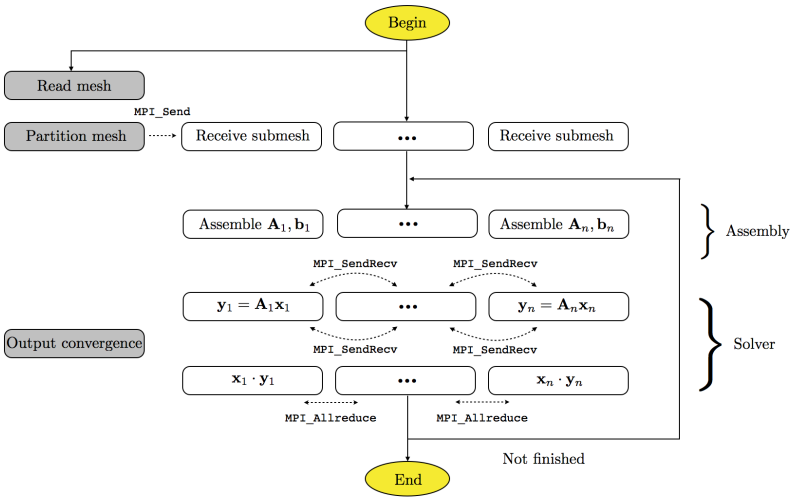


Figure 2.10: Parallel flowchart. Master (grey) and n slaves (white).

with the execution of the iterative solvers carried out by the slaves, two types of communications are required to exchange interface information with their neighbours using the MPI functions:

- Point-to-point communications via `MPI_SendRecv`, which are used when sparse matrix-vector products are carried out.
- Global communications via `MPI_AllReduce`, which are used to compute residual norms and scalar products.

All solvers need both these types of communication, but, when using complex solvers like the DCG, additional operations may be required, such as the `MPI_AllGatherv` functions explained in öhner, Mut, Cebral, Aubry, & Houzeaux (2011). In the current implementation of Alya, the solution obtained in parallel is, up to round-off errors, the same as the sequential one all the way through the computation. This is because the mesh partition is only used for

distributing work without, in any way, altering the actual sequential algorithm. This would not be the case if one took into account more complex solvers, like the primal/dual Schur complement solvers, or more complex preconditioners, like linelet Soto, Löhner, & Camelli (2003) or block LU.

The code has proven to scale well on the main European supercomputers, as shown in Figure 2.11. As implemented in this work, the HERMESH method

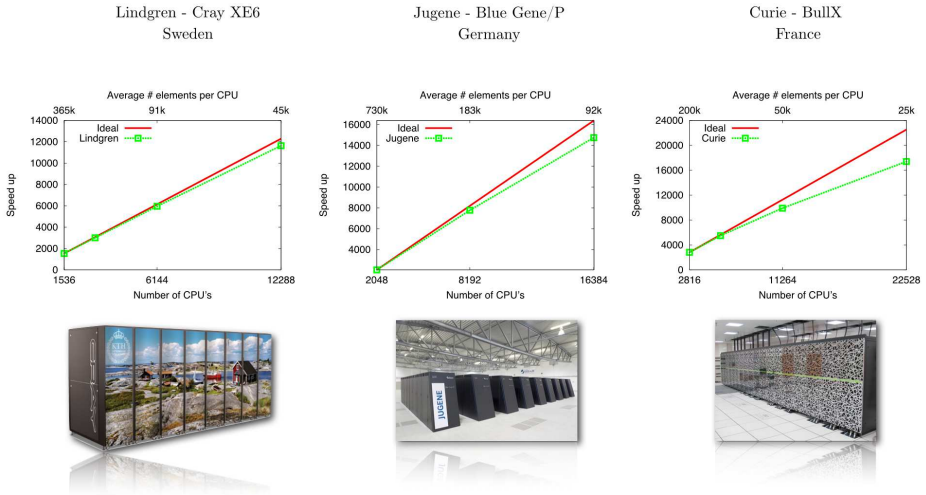


Figure 2.11: Parallel performance of Alya.

can be viewed as part of the preprocess and the mesh partitioning is carried out after the mesh gluing. Therefore, the parallel performance of the code is automatically inherited by the current implementation.

Chapter 3

Proposed Method : *HERMESH*

3.1 Motivation

As has been presented in the first chapter we propose to devise a mesh composing method with certain characteristics. In particular, the three requirements of the method were: implicit, versatile and parallel. These are essential conditions in order to be able to solve real problems in different contexts as they are solved in the department where HERMESH has been developed. So the method has to be able to address problems with non-matching grids whether they be disjoint with (as can be observed in Figure 3.1-a) or without a gap (as can be observed in Figure 3.1-b). Also, the method has to be able to solve the mesh coupling with a certain overlapping between independent meshes (as we can see in Figure 3.1-c) or even Chimera-type problems (illustrated in 3.1-d).

The scenarios represented in Figure 3.1 a,b and c belongs to the application referred to in the first chapter as *mesh gluing*. The other scenario, Figure 3.1-d, corresponds to the *Chimera-type problem*. The particular details of the proposed method for both cases will be explained in the following chapters, 4 and 5. In this chapter, the principles and general properties of the HERMESH method will be described, as will certain details related to the implementation.

In order to fulfil the three main requirements described before (implicit, parallel and versatile), we have come a long way, and it is basically divided in these three main points:

- Making implicit the explicit conditions transmissions which was described in Houzeaux (2003).

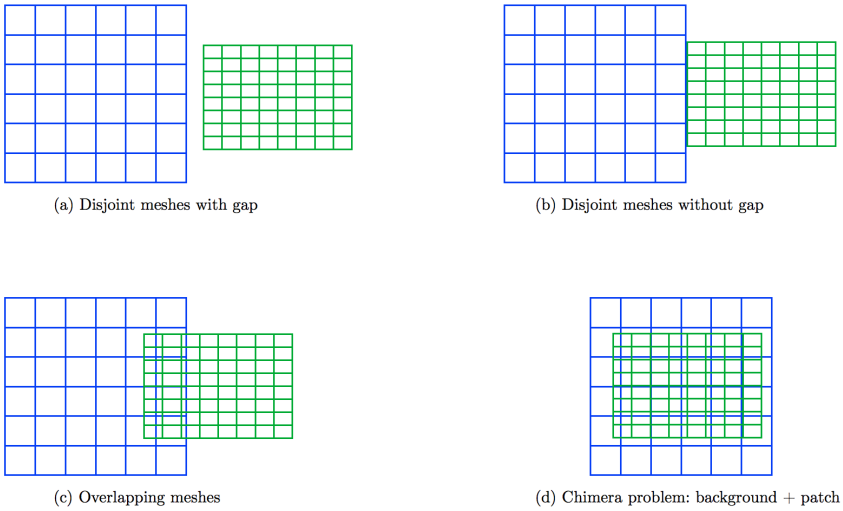


Figure 3.1: Possible HERMESH scenarios

- Stabilizing the transmission conditions in the variational multiscale framework.
- The HERMESH method.

These points will be described in this chapter in order to gain a better understanding of proposed method.

3.2 First effort: implicit transmission conditions

In order to achieve the main objectives of the new mesh coupling technique, the first effort entailed implementing in an implicit way the classical transmission conditions like Dirichlet, Neumann or Robin, typical of the *iteration-by-subdomain* methods explained in the first chapter. This first idea comes from the natural evolution of the work developed in Houzeaux (2003). They developed a new Chimera method, *overlapping mixed iteration-by-subdomain domain decomposition method* as an extension of certain existing DD methods to the case of overlapping subdomains. The transmission conditions on the interfaces are mixed, i.e. they are of different type on each side of the interfaces and the solutions on the subdomains are coupled iteratively until convergence is achieved. In that work, a study of a one-dimensional scalar advection-diffusion-reaction equation enabled them to identify the possible benefits of using overlapping subdomains together with a mixed DD method.

The overlap is good for two main reasons, convergence and order of accuracy in Neumann conditions. In particular, they analyzed the Dirichlet/Dirichlet method, or Schwarz method, the Dirichlet/Neumann method and the Dirichlet/Robin method for overlapping and disjoint subdomains. They discussed the importance of the relaxation parameter to gain control on the stability of the DD algorithm, and apart from the general ADR equation, they studied three limiting behaviors of the equation, i.e. the Poisson equation, the advection-diffusion equation, and the hyperbolic limit. The most important result is that even in the hyperbolic limit, Dirichlet and Neumann (or Robin) conditions can be placed indifferently with respect to the direction of the advection in order to achieve convergence. They concluded that the overlap renders mixed methods more robust. Then they studied an overlapping Dirichlet/Robin method within a variational framework for a two-subdomain partition and showed the convergence of the relaxed sequential algorithm. Using the finite element approximation, they showed that the overlapping methods lead to an algebraic preconditioned Richardson procedure. Although reflecting transmission conditions are undesirable in the advection dominated range, as they destabilize the iterative algorithm, they showed that mixed methods diffuse the error much more rapidly and that considerable gain in convergence can be obtained even with a small geometric overlap.
Before showing the way of implicitly imposing the transmission conditions, we are going to briefly review the explicit strategy. In particular, we are going to concentrate on the D/N case when we solve the problem $L(u) = f$ for a simple two-subdomain problem, as the one illustrated in Figure 3.2. Considering that L represent de Laplacian operator, in a strong form for the continuous case, the problem is formulated as follows:

$$\begin{cases} L(u_1^{k+1}) = f_1 & \text{in } \Omega_1, \\ u_1^{k+1} = u_2^k & \text{on } \Gamma_{12}, \\ L(u_2^{k+1}) = f_2 & \text{in } \Omega_2, \\ \nabla u_2^{k+1} \cdot \boldsymbol{n}_2 = \nabla u_1^{k+1} \cdot \boldsymbol{n}_1 & \text{on } \Gamma_{21}, \end{cases} \tag{3.1}$$

where k and $k+1$ is the iteration index. Let us consider the operator $L(u)$ is the one defined in 2.1 by the bilinear form given by 2.3 and the appropriate spaces U_i, V_i with $i = 1, 2$, meaning that the weak form of the same problem has the next expression:

$$\begin{cases} a(u_1^{k+1}, v_1) = (f_1, v_1) & \forall v_1 \in V_1, \\ u_1^{k+1} = u_2^k & \text{on } \Gamma_{12}, \\ a(u_2^{k+1}, v_2) = (f_2, v_2) - \int_{\Gamma_{21}} \nabla u_1^{k+1} \cdot \boldsymbol{n}_2 v_2 d\Gamma & \forall v_2 \in V_2, \end{cases} \tag{3.2}$$

where we have used the fact that $\boldsymbol{n}_1 = -\boldsymbol{n}_2$. Now, we want to evaluate the possibility of implementing this transmission condition implicitly to avoid the iteration between subdomains with the associated convergence problems.

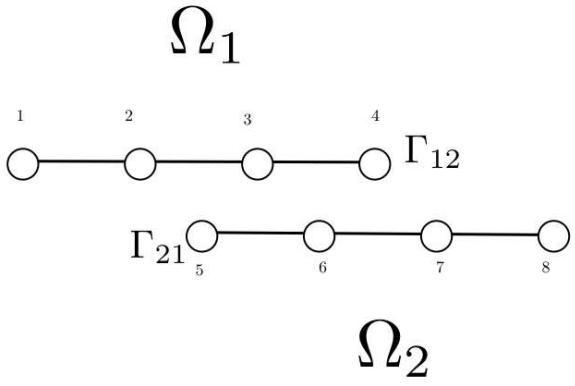


Figure 3.2: Two overlapping subdomain problems in one dimension.

How we proceeded. From the previous problem presented in Equations 3.2, we are going to present the formulation in the discrete level of the implicit boundary condition, Dirichlet and Neumann.

- The Dirichlet condition is linearly imposed by interpolating u_2 on Γ_{12} from the other subdomain Ω_2.
- The Neumann condition is computed by adding the contribution of Ω_1 on Γ_{21}. This can be implemented using different orders of accuracy, as follows:
  - *First order*: We have to search the element of the other subdomain where the Gauss point of the interface Γ_{21} is located. From this element we interpolate the gradient of the velocity, ∇u_1, with the gradient of the shape function evaluated in the interface Gauss point. See the left part of Figure 3.3.
  - *Second order*: We have to search the element of the other subdomain where the Gauss point of the interface Γ_{21} is located. From this element we calculate the smoothed gradient on the nodes and interpolate it with the value of the shape function in the interface's Gauss point. This strategy is illustrated on the right part of Figure 3.3.

Following this ideas, the implicit formulation of the problem is:

$$\begin{cases} a(u_{h,1}, v_{h,1}) = (f, v_{h,1}) & \forall v_{h,1} \in V_{h,1}, \\ u_{h,1} - u_{h,2} = 0 & \text{on } \Gamma_{12}, \\ a(u_{h,2}, v_{h,2}) + \int_{\Gamma_{21}} \nabla u_{h,1} \cdot \boldsymbol{n}_2 v_{h,2} d\Gamma = (f, v_{h,2}) & \forall v_{h,2} \in V_{h,2}, \end{cases} \tag{3.3}$$

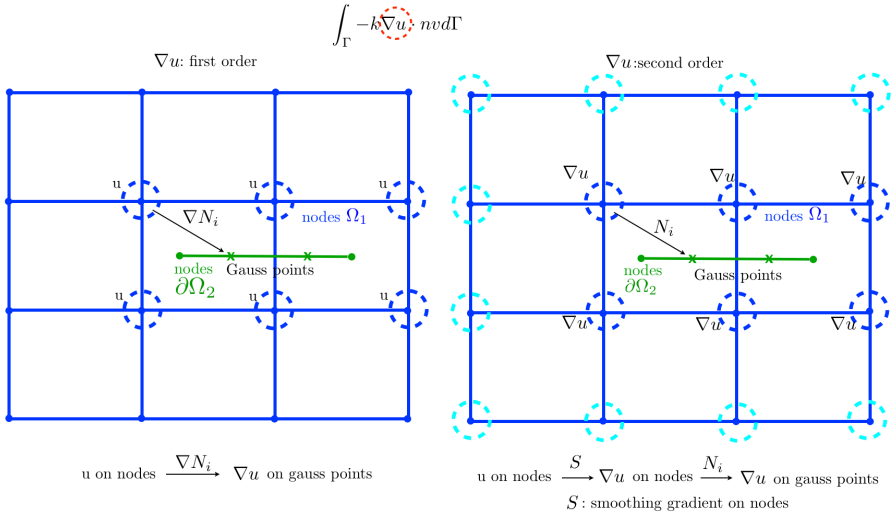


Figure 3.3: First and second order of the gradient in Neumann conditions.

In this implicit formulation, the superscripts are no more necessary, as we avoid the iterative loop between subdomains. At this discrete level, we have to assemble in the matrix these transmission conditions, i.e., $u_{h,1} - u_{h,2} = 0$ in the equations of $u_{h,1}$ for the Dirichlet condition and the integral boundary $\int_{\Gamma_{21}} \nabla u_{h,1} \cdot \boldsymbol{n}_2 v_{h,2} d\Gamma$ which represent the Neumann condition. These new contributions imply new connectivities of the interface nodes with the nodes of the other subdomain. What determines the nodal connectivities in a finite element code is the element connectivity. Let us define `lnods(1:nnode,1:nelem)` the element connectivity array - that is, the nodes belonging to a given element, and where `nnode` is the number of nodes per element. From this array we create the resulting matrix graph in CSR format as a linked list with the arrays `ia` and `ja`, such that `ja`(`ia`(i):`ia`($i+1$) $-$ 1) are the neighboring nodes of node i . Bearing in mind that our objective is to introduce the implicit transmission conditions in a way as unintrusive as possible, we originally suggested to implement the strategy introducing a new elements called*virtual elements*. These elements are responsible for adding the new terms associated to the row of the fringe nodes, as we will see next. In the Figure 3.4, on the right and top, we can compare the difference between an explicit and an implicit strategy.

For the particular two-subdomain problem formulated in the equations 3.3, these terms are represented in Figure 3.5 where we can see that in node 4, where we impose the Dirichlet condition, we eliminate the row associated to it and substitute that with the coefficients that impose continuity of the solution with the other subdomain. We have pictured in green the new connectivities mentioned before, coming from the introduction of the *virtual element* associated

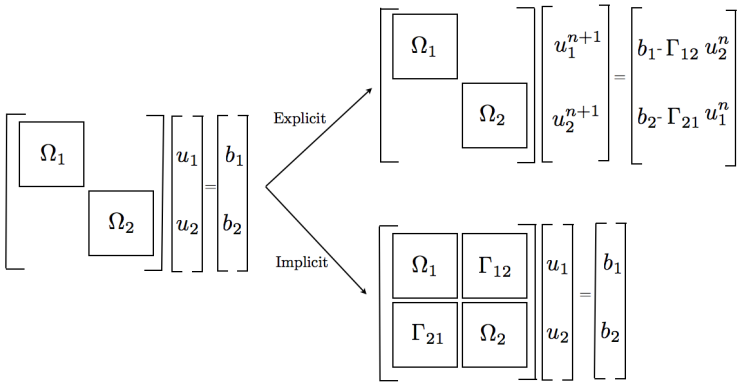


Figure 3.4: Schematic system matrix. (Top) Explicit scheme. (Bottom) Implicit scheme. (Left) Without coupling. (Right) Coupled system.

to the Dirichlet condition. In the same way, in order to impose the Neumann

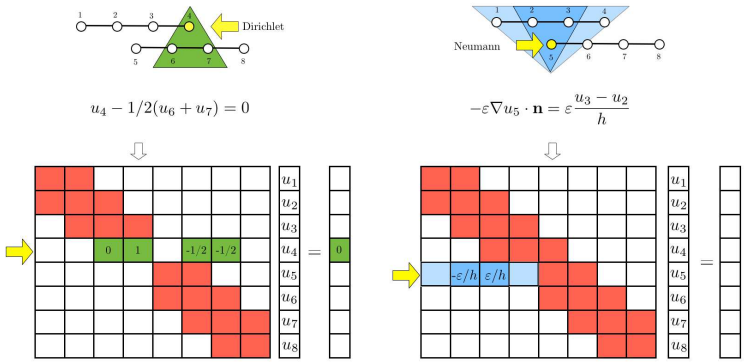


Figure 3.5: Assembling of the Dirichlet and Neumann conditions

condition on node 5, we have also pictured in blue the new connectivities, created by the *virtual element* associated with this node. In this case, we do not eliminate the original coefficients but we add new coefficients in the matrix. If the gradient of the unknown in the Neumann expression is calculated at first order, only nodes 2 and 3 are taken into account, but if it is calculated at second order using the gradient's projection, values at the position 1 and 4 also appear. The difference of the order convergence in both cases is shown in Figure 3.6. We can observe that the smoothed gradient obtains a second order mesh converge while the other strategy has a first order of convergence.

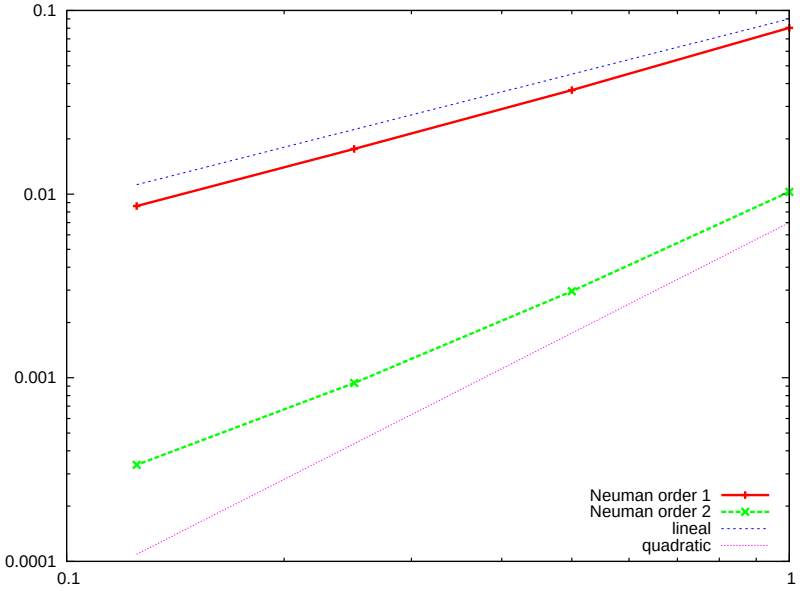
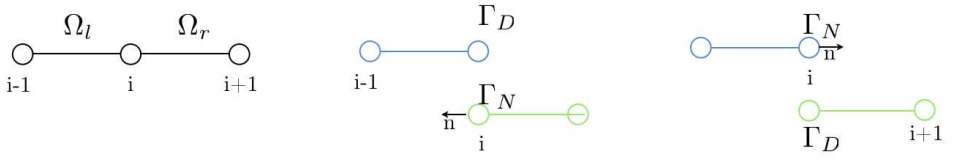


Figure 3.6: Order of convergence of D/N problem with order 1 or order 2 of Neumann condition.

3.3 Stabilized and implicit Neumann transmission conditions

Now that the implicit transmissions condition have been presented, the idea is to study the contribution of the subscale in Neumann transmissions. As is well known, the equation associated to one node i comes from the contribution of the elements connected to it, when we solve with finite element methods.



(a) One-domain problem (b) Two-domain problem: D/N (c) Two-domain problem: N/D

Figure 3.7: (Left) One-domain problem.(Middle) D/N two-domain problem.(Right) N/D two-domain problem.

In particular, for the advection-difussion-reaction problem, i.e. Equation

2.1, the terms associated to both elements connected with the node i, Ω_l (left element) and Ω_r (right element) in case (a) of Figure 3.7 in the one-dimensional case, using the variational multiscale approach described in 2.1, are:

$$\begin{aligned}\Omega_l : & \frac{\varepsilon}{h}(u_i - u_{i-1}) + (1-\flat)\frac{a}{2}(u_i - u_{i-1}) \\ & +(s + (1-\flat)\frac{da}{dx})\frac{h}{3}(u_i + \frac{u_{i-1}}{2}) - \frac{\flat a}{2}(u_i + u_{i-1}) \\ & +\frac{\tau a^2}{h}(u_i - u_{i-1}) - \frac{\tau s a}{2}(u_i - u_{i-1}) - \frac{fh}{2} + \frac{\tau s f h}{2} - \tau a f \end{aligned} \tag{3.4}$$

and for the other element:

$$\begin{aligned}\Omega_r : & \frac{\varepsilon}{h}(u_i - u_{i+1}) - (1-\flat)\frac{a}{2}(u_i - u_{i+1}) \\ & +(s + (1-\flat)\frac{da}{dx})\frac{h}{3}(u_i + \frac{u_{i+1}}{2}) + \frac{\flat a}{2}(u_i + u_{i+1}) \\ & +\frac{\tau a^2}{h}(u_i - u_{i+1}) + \frac{\tau s a}{2}(u_i - u_{i+1}) - \frac{fh}{2} + \frac{\tau s f h}{2} + \tau a f \end{aligned} \tag{3.5}$$

If we compare the terms of the left volume element, Equation 3.4 with the corresponding first order Neumann transmission condition given by the term of the right part of Figure 3.5, we observe that the first term is the only term that we recover in the interface boundary conditions (if $\flat = 0$). So there are many other terms in the volume element equation, about which we do not have information in the Neumann transmission condition. For this reason, we thought that perhaps some of them appeared in the stabilized form of the Neumann boundary condition, which is given by the following expression, stemming from the Equation 2.23 :

$$-\int_{\Gamma_N} (\varepsilon \nabla u_h \cdot \boldsymbol{n} - \flat(\boldsymbol{a}\cdot\boldsymbol{n})u_h)v_h d\Gamma + \int_{\Gamma_N} \tilde{u}(\varepsilon\nabla v_h \cdot \boldsymbol{n} + (\boldsymbol{a}\cdot\boldsymbol{n})v_h)d\Gamma, \tag{3.6}$$

where the term $\tilde{u}$ is given by the expression: $\tilde{u} = \tau(x)R(u_h)$ where $R(u_h)$ represents the residual of the Equations, i.e., $R(u_h) = f - L(u_h)$.

We have compared the terms stemming from the case (a) of Figure 3.7, with the stabilized Neumann conditions in the interface of the two-domain problem as is illustrated in case (b) or (c). For simplicity, the two-subdomain problems will be disjoint and with matching nodes. In particular, for advection-diffusion problem, without reaction term, $\boldsymbol{a} > 0$ for both cases, D/N and N/D we are going to compare the contribution in the interface stemming from Equation 3.6 and the volume term corresponding to the element of the neighbor subdomain, i.e., Equation 3.4 in D/N case or Equation 3.5 in N/D case. The expressions becomes:

- D/N: Considering $n = -1$, $\nabla v_h = -1/h$ and $v_h = 1$ in Γ_N;

– interface condition:

$$-\int_{\Gamma_N}(\varepsilon\nabla u_h\cdot\boldsymbol{n}-\flat(\boldsymbol{a}\cdot\boldsymbol{n})u_h)v_h d\Gamma+\int_{\Gamma_N}\tilde{u}(\varepsilon\nabla v_h\cdot\boldsymbol{n}+(1-\flat)(\boldsymbol{a}\cdot\boldsymbol{n})v_h)d\Gamma$$
$$=\frac{\varepsilon}{h}(u_i-u_{i-1})-\flat a(u_i+u_{i-1})+(1-\flat)\frac{\tau a^2}{h}(u_i-u_{i-1})$$
$$-\frac{\tau a\varepsilon}{h^2}(u_i-u_{i-1})+\tau f\left(\frac{\varepsilon}{h}-a\right); \quad (3.7)$$

– left volume element contribution of the neighbor subdomain, Ω_l:

$$\frac{\varepsilon}{h}(u_i-u_{i-1})+(1-\flat)\frac{a}{2}(u_i-u_{i-1})+\frac{\flat a}{2}(u_i+u_{i-1})$$
$$+\frac{\tau a^2}{h}(u_i-u_{i-1})-\frac{fh}{2}-\tau af; \quad (3.8)$$

- N/D: Considering $n=1$, $\nabla v_h=1/h$ and $v_h=1$ in Γ_N;

– interface condition:

$$-\int_{\Gamma_N}(\varepsilon\nabla u_h\cdot\boldsymbol{n}-\flat(\boldsymbol{a}\cdot\boldsymbol{n})u_h)v_h d\Gamma+\int_{\Gamma_N}\tilde{u}(\varepsilon\nabla v_h\cdot\boldsymbol{n}+(1-\flat)(\boldsymbol{a}\cdot\boldsymbol{n})v_h)d\Gamma$$
$$=-\frac{\varepsilon}{h}(u_{i+1}-u_i)+\flat a(u_i+u_{i-1})-(1-\flat)\frac{\tau a^2}{h}(u_{i+1}-u_i)$$
$$-\frac{\tau a\varepsilon}{h^2}(u_{i+1}-u_i)+\tau f\left(\frac{\varepsilon}{h}+a\right); \quad (3.9)$$

– right volume element contribution of the neighbor subdomain, Ω_r:

$$-\frac{\varepsilon}{h}(u_{i+1}-u_i)+(1-\flat)\frac{a}{2}(u_{i+1}-u_i)-\frac{\flat a}{2}(u_{i+1}+u_i)$$
$$-\frac{\tau a^2}{h}(u_{i+1}-u_i)-\frac{fh}{2}+\tau af; \quad (3.10)$$

So, apparently, with the stabilization term in the interface more terms included in the volume element expression appear. To quantify the effect of this term in the Neumann transmission condition, we conducted the mesh convergence test for an advection-diffusion problem with two subdomains imposing both D/N and N/D with different regimes for the Peclet number, i.e., $Pe=\frac{aL}{2\varepsilon}$, $Pe=0.5$, $Pe=5.0$ and $Pe=5.0\cdot10^{-4}$, and with $\flat=0$. Two different stabilization parameters τ have been tested, given by Equation 2.31 and Table 2.2. In particular, Codina and Present work.

We can observe with these graphics shown in Figures 3.8, 3.9 and 3.10 that the stabilization term added in the Neumann transmission condition is not relevant. We just only see the utility of the term in a pure convection problem where we obtain error zero with the exact stabilization parameter while if we do not add the stabilization term in the Neumann transmission condition the problem is unresolved, so we conclude that the problem in non well-posed.

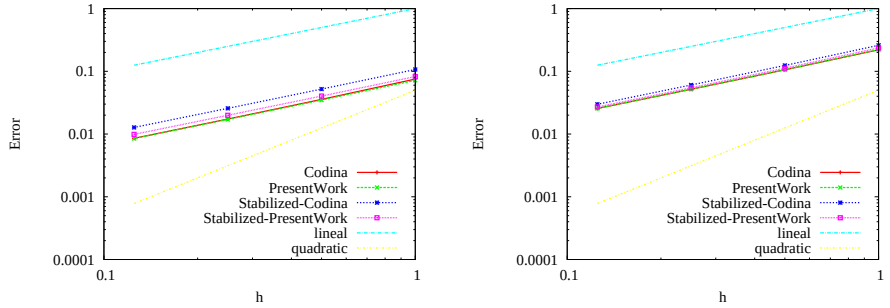


Figure 3.8: Advection-diffusion equation with $Pe = 0.5$. (Left) D/N transmission conditions. (Right) N/D transmission conditions.

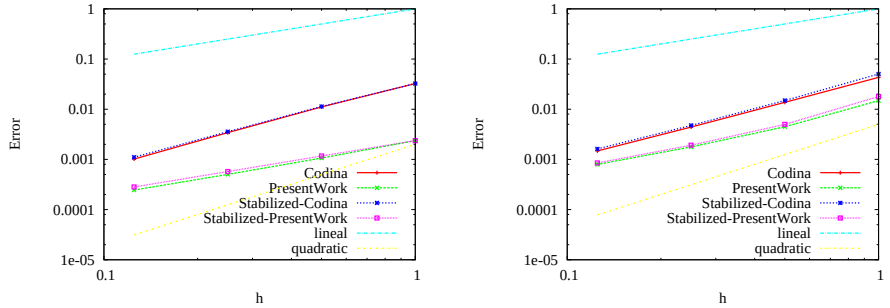


Figure 3.9: Advection-diffusion equation with $Pe = 5.0$. (Left) D/N transmission conditions. (Right) N/D transmission conditions.

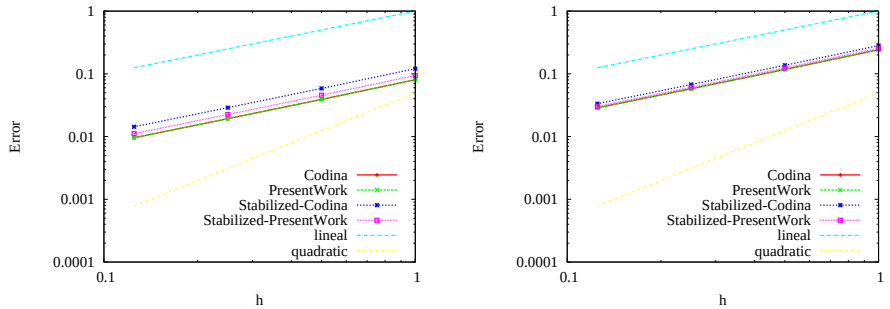


Figure 3.10: Advection-diffusion equation with $Pe = 5.010^{-4}$. (Left) D/N transmission conditions. (Right) N/D transmission conditions.

3.4 HERMESH. General description

After ruling out our previous hypothesis about the stabilized implicit Neuman transmission condition we thought in another strategy in order to achieve the implicit coupling. The rest of this chapter presents the principles of the HERMESH method. The framework in which the method works is that of the finite element methods. For an extensive theory of this numerical technique, see Zienkiewicz & Taylor (1977); Hughes (2012) among many others. HERMESH is based on a geometrical coupling that allows us to assemble independent meshes, with or without overlapping. The continuity of the solution is achieved by newly constructed elements which have slightly different behavior from that of the original elements of the meshes. These elements are referred to by us as *extension elements* since, in fact, what they do is extend one subdomain to the other and connect the existing nodes in both meshes. This is an important point because the method does not introduce more degrees of freedom. The idea is depicted in a simple one-dimensional case in Figure 3.11, where we have illustrated the extension elements that are constructed when coupling two disjoint meshes and reproducing the one-domain problem.

In the top part of the figure, we have shown a one-domain problem in one dimension and the shape function associated to each node of the mesh. Next, in the middle part, we have depicted the problem in two disconnected subdomains. Finally, in the bottom part of Figure, we have drawn the two extension elements (with discontinuous line in the figure) associated to each interface node, called fringe nodes. We can observe that in order to assemble the two independent meshes, we have to create two elements while, in the one-domain case, the two fringe nodes are connected with one element. Each extension element connects the nodes of the adjacent subdomains in order to create a shape function with compact support. So, as the extension elements only contain the shape function associated to the fringe node, the contribution in the global matrix is different from any other original element, as we will see in detail below.
To better understand what the contribution is, in an algebraic sense, of this non-standard extension elements, let us consider the following simple Laplace problem:

$$d^2u/dx^2 = 0 \quad \text{in } [0,4]\,, \tag{3.11}$$

$$u(0) = 1, \quad u(4) = 3, \tag{3.12}$$

In Figure 3.12 we can compare the resolution of this equation on two overlapping subdomains with the Dirichlet/Dirichlet (D/D) method (left part of the figure) and two disjoint subdomains with the HERMESH method (right part of Figure). The top part of the Figure shows the meshes and node numberings before the couplings. The top matrices are the matrices assembled on these meshes, blue block of matrix (corresponding to blue subdomain) and

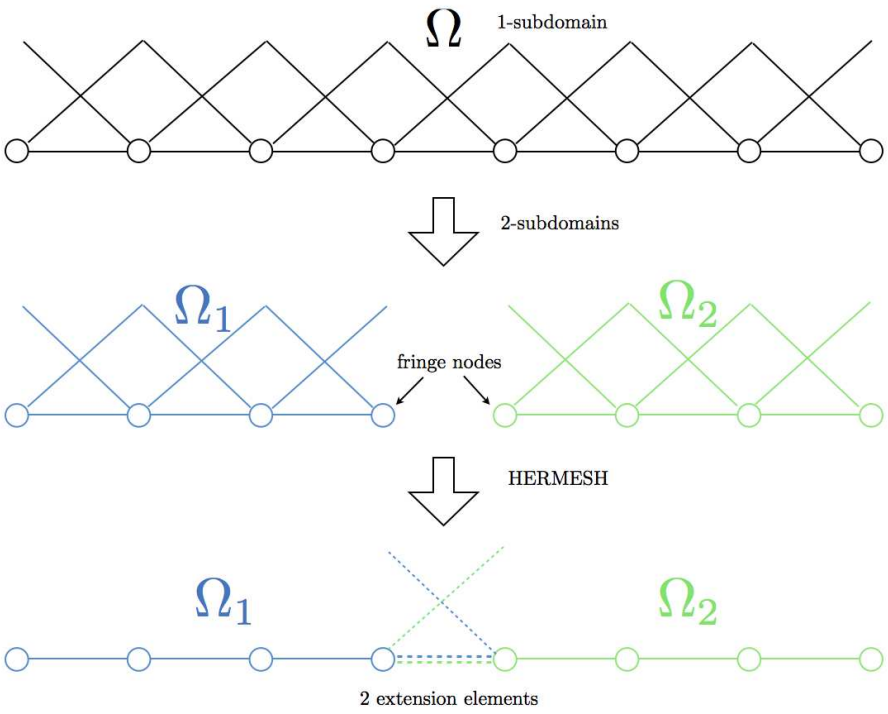


Figure 3.11: Joining two-subdomain problem with HERMESH in one-dimensional case.

green block (green subdomain), and therefore before the couplings. Then, the middle meshes illustrate the couplings. Finally, the bottom matrices are the resulting matrices obtained after the D/D and HERMESH couplings. As far as the D/D method is concerned, the coupling is achieved by substituting the rows of nodes 4 and 5 in order to impose $u_4 = u_7$ and $u_5 = u_2$. As far as the HERMESH method is concerned, the steps of the method are:

- *Step 1: Identifying interfaces.* Obtaining a list of the fringe nodes of each subdomain. In this case, the fringe nodes are nodes 3 and 6.

- *Step 2: Extending shape functions.* Extending the shape functions of the fringe nodes towards the neighboring subdomains.

- *Step 3: Imposing Dirichlet condition.* Implicitly imposing the Dirichlet conditions by choosing the neighbor's nodes to close the support of the shape functions. The shape function of node 3 extends to node 7, while the shape function of node 6 extends to node 2.

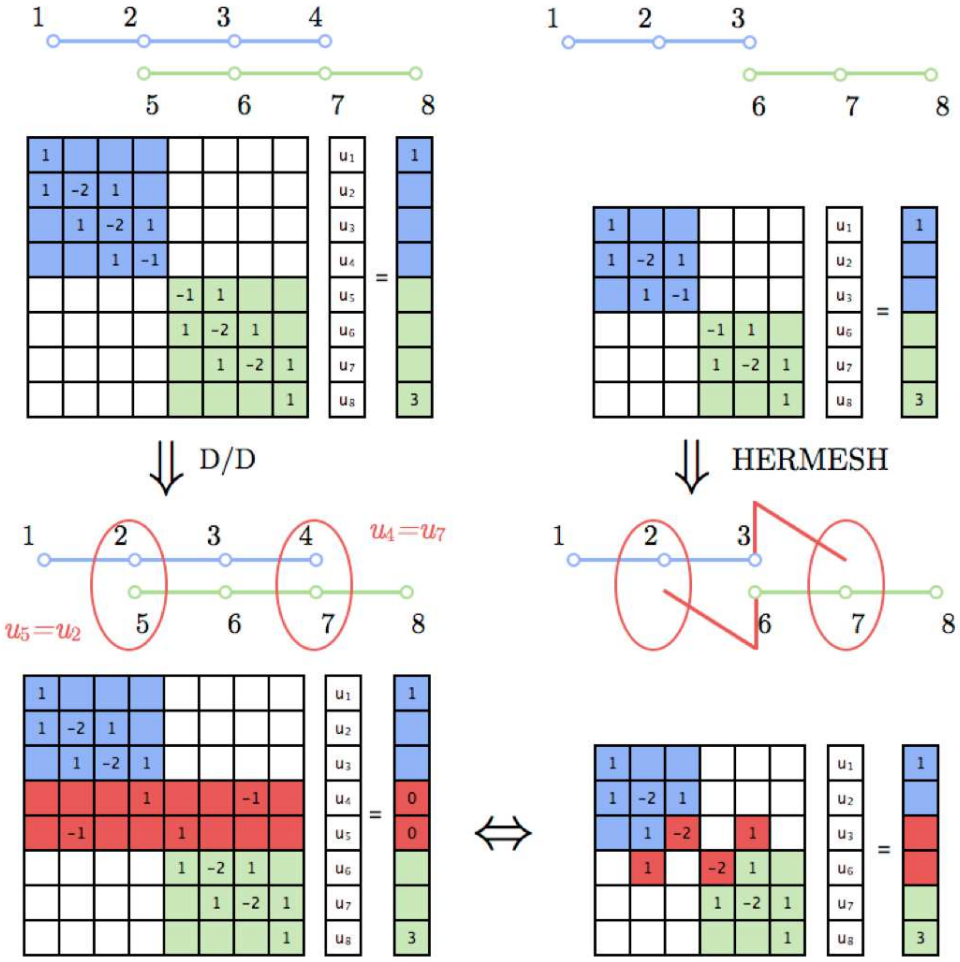


Figure 3.12: Principles of the HERMESH method and comparison with a Dirichlet/Dirichlet method.

In practice, Step 2 and Step 3 are carried out at the same time. That is, the extensions of the shape functions of the fringe nodes (3 and 6) are carried out by adding *extension elements* to the mesh, shown in Figure 3.12 (Mid.). In one dimension, the extension element of a fringe node is the element to be created from the boundary fringe node (here node 3 and 6) to a free node of the neighboring subdomain (nodes 2 and 7). The process of the extension elements construction in two- and three- dimensions is more elaborate and will be described in the next section.
To finalize the presentation of the HERMESH method, we would like to em-

phasize the main properties of the method:

- Error is zero if the solution is in the finite element space.
- Method is order 2 for linear elements.
- If the problem composed by independent meshes plus the extension elements coincide with the mesh in one domain problem, both solutions are exactly the same.
- Gradients and mass matrix are calculated naturally.
- Coupling is implicit.
- Parallelization is direct.

3.5 Construction of the extension elements

The description of the construction process of the extension elements in two and three dimensions will be divided for clarity. The extension elements extend from the boundary connected to the fringe nodes. In two dimensions these boundaries are bar elements, while in three dimensions the boundaries can be quadrilaterals or triangles. In two dimensions, we use triangles to extend from the bar elements. In three dimensions, two extension elements are necessary: pyramids for quadrilaterals boundaries and tetrahedra for triangles boundaries.

3.5.1 Two dimensions

As illustrated in Figure 3.13, the process for constructing the extension elements from Ω_1 to Ω_2 consists of the following steps:

- Identifying fringe nodes f of Ω_1.
- Given a fringe node, we identify its host element in Ω_2, that is, the element where the node lies. This could be efficiently done with either a bin or a quad/oct tree strategy, see Houzeaux & Codina (2003a) for more details.
- Making a list of candidate nodes (used to create the extension triangle elements). This list is first filled with the neighboring nodes of the host element nodes. In this list, eliminating the nodes of Ω_2 located inside Ω_1. To do that, we use an a *skd-tree* strategy, as explained in Khamayseh & Kuprat (2008). Skd-trees are used to efficiently find the signed shortest distance between a point and a surface, as is mentioned in the presentation of the level set equations in chapter 2.

- Computing the exterior normals of the two edges connected to the fringe node. Applying a *windscreen criterion*: taking the candidate nodes out of the list if the angle formed between these normals and the vector between the fringe and candidate node is above a certain criteria. For example, in top part of Figure 3.13, the bottom right candidate node would be taken out.

- From this reduced candidate list, choosing the nodes that form the best triangles in terms of quality Q, to be treated later. They are the red elements in Figure 3.13 and we have illustrated two different options that could be valid. On the left, the first option is composed of three extension elements. This is because the candidate node chosen by the left boundary, the extension node in blue, is different from the extension node of the right boundary. So we have to close the extension with an additional triangle, the middle one formed by the fringe node and two selected candidate nodes. This way of closing the extension in variational terms means that the shape function associated to the fringe node has a compact support. On the right part of Figure 3.13, the extension node of both boundaries is the same, so we no longer need a triangle to close the extension.

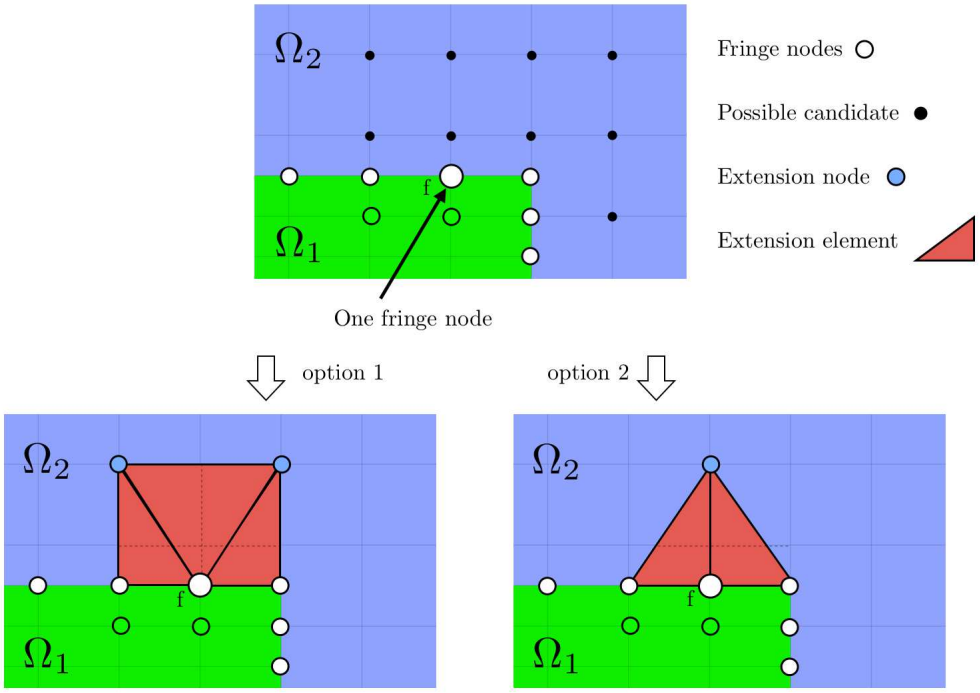


Figure 3.13: Two-dimensional extensions process.

Conformity issue. In order to have a greater flexibility when selecting the best extension elements, we have allowed the possibility for the crown made by extension elements to be non-conforming. The crown is defined here as the union of extension elements of one subdomain to another. Figure 3.14 depicts

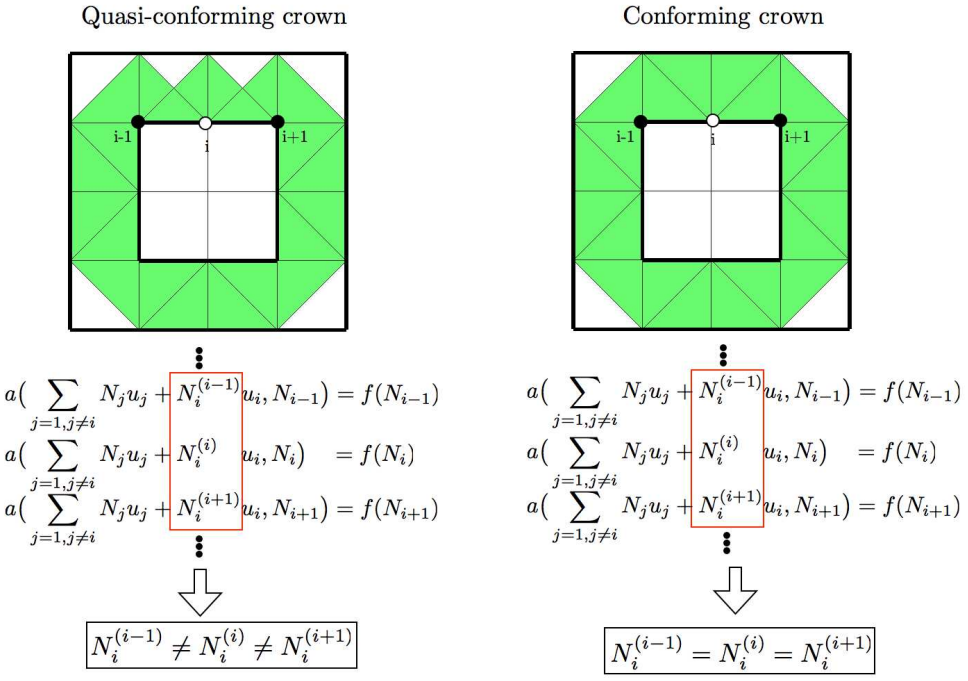


Figure 3.14: Extension elements in two dimensions. (Left) Non-conforming crown. (Right) Conforming crown.

the concept of a crown for both, quasi-conforming (left part of the figure) and conforming (right part of the figure) cases, making clear the difference at the discrete level formulation. We observe that in quasi-conforming crown, the shape function associated to node i is different depending on the equation to be consider. In the equation of node $i-1$ the contribution of the node i, that is the shape function $N^(i-1)_i$ is not the same as the contribution of node i in its own equation $(N^(i)_i)$ neither the contribution of node i in the equation of the neighbor node $i+1$ $(N^(i+1)_i)$. On the other hand, the conformity crown, all the shape function are the same.

In Figure 3.15 we have pictured the crown associated to one fringe node, in order to appreciate the difference between conforming and non-conforming crown. It shows the extension elements of three fringe nodes, leading to a non-conforming crown (left part) and to a conforming crown (right part). On the left part, the elements extending from the boundaries connected to the middle fringe node (yellow one) are not coinciding. The resulting test function

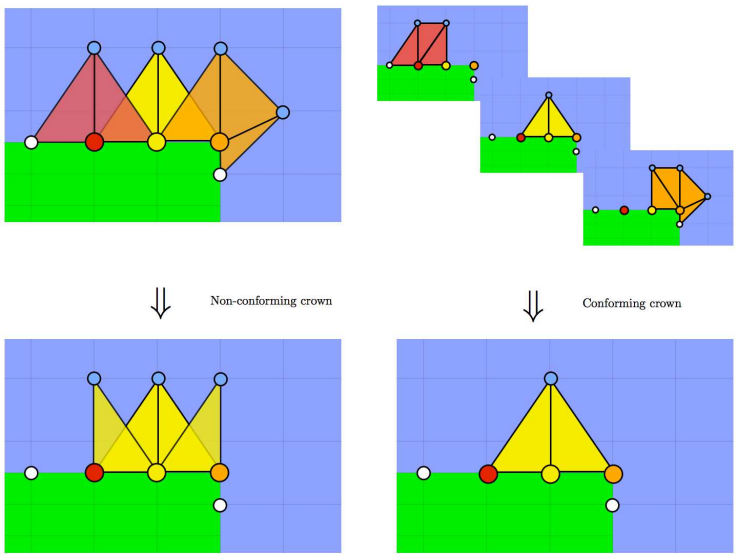


Figure 3.15: Extension elements in two dimensions. (Left) Non-conforming crown. (Right) Conforming crown.

used to assemble the equation of this node is uniquely defined and its support corresponds to the yellow triangle. However, when assembling the equation of its neighboring fringe nodes, different approximations are used (for example, right red triangle does not coincide with left yellow triangle). On the other hand, in the conforming case, the extension elements connected to the fringe nodes always coincide. The crown is thus conforming. This discussion about conformity is valid to a 3-dimensional case but for simplicity it is shown here in a two-dimensional case.

3.5.2 Three dimensions

The extension to 3D is not as straightforward as it looks, and the challenge is higher for boundary layer meshes. The process is divided into two steps. First, the creation of extension elements connected to the boundaries (faces) of the fringe node and then, if necessary, the closing of the extension.

Figure 3.16 shows an example of one of these extension functions associated to one of the fringe nodes. The steps for creating the extension associated to a fringe node f from its connected boundaries are the following.

Step 1. Create extensions from connected boundaries:

- Ordering the boundaries (faces) b_i connected with f in a clockwise or counter-clockwise direction.

- Creating for each b_i a list of candidate extension nodes e_i of the other subdomain, according to the *windscreen criteria*, mentioned in section 3.5.1.
- Ordering this list in accordance with the quality of the tetrahedra which could be formed between b_i and e_i.
- Grouping the selected extension nodes according to the number of appearances in the previous boundary b_i list. The ranking is therefore a combination between quality and popularity.

With respect to Figure 3.16, we have chosen 3 extension nodes e_1, e_2, e_3 and created at this first step 4 extension elements, t_1, t_2, t_3, t_4. Note that e_2 is used by two extension elements that come from b_2 and b_3.

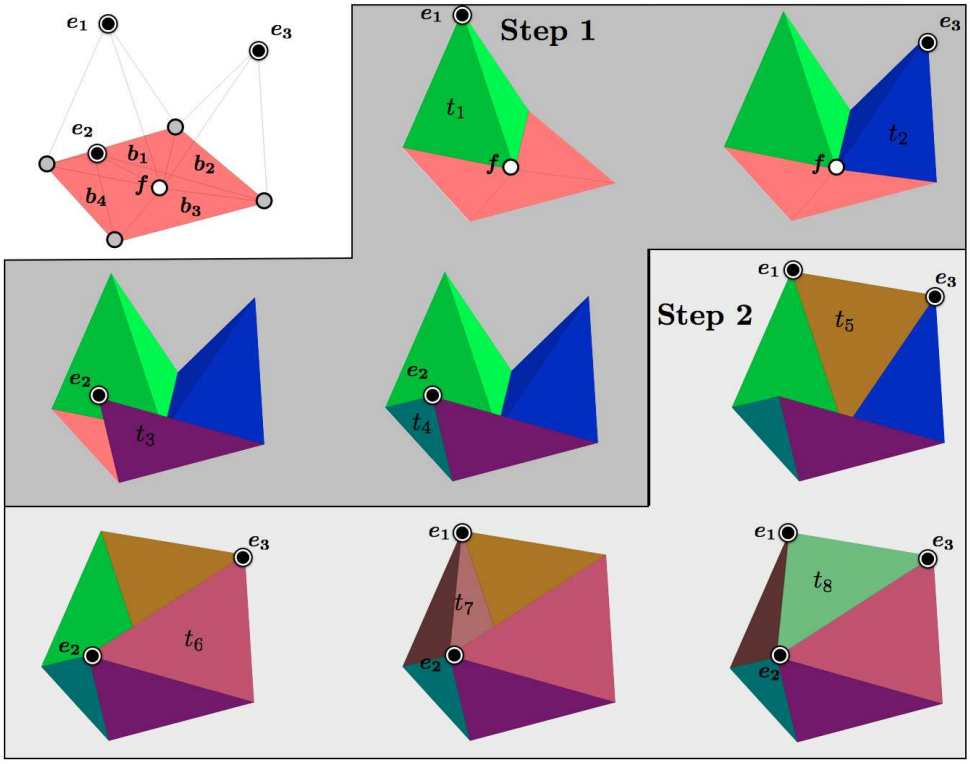


Figure 3.16: 3D extensions process. Step 1 and Step 2.

Then, if necessary, depending on the number of selected candidates, the extension should be closed with more tetrahedra, as in the example of Figure 3.16. The procedure is as follows.

Step 2. Close extensions:

- Join pairs of extension elements having node f + two contiguous nodes e_i's + boundary node belonging to two contiguous boundaries b_i with different nodes e_i's. In the case of Figure 3.16, the first closing element is the brown one, t_5 shown at the first stage of the second step. Using this procedure we then create the next two purple elements involving $e_3 - e_2$, i.e. t_6 and $e_2 - e_1$, i.e. t_7.

- If needed, close the extensions from above. At this stage we are only left with a minimum of three extension nodes e_i and the fringe. In particular, with three different e_i, only one tetrahedron is required to close the extension and is formed by the e_i's + f. This is the case of the last element in Figure 3.16, , tetrahedron t_8. For more than three extension nodes (say n), we have multiple alternatives to create the remaining tetrahedra to close the extension. The number of combinations $\mathcal{C}_n$ is given by the so-called Catalan number, see Shapiro (1976) for more details. $\mathcal{C}_n$ is the number of different ways a convex polygon with $n + 2$ sides can be cut into triangles connecting vertices with straight lines. The Catalan number is exponentially complex. As we have few elements in average, we observe linear complexity in practice. From this $\mathcal{C}_n$ possibilities, we choose the one that gives the best global quality Q.

Figure 3.17 and 3.18 shows two practical examples where Step 1 (Figure 3.17) and Step 1 + Step 2 (3.18) are required. In the first case, Figure 3.17, we observe that the extension is automatically closed at the end of Step 1 since all the boundaries have chosen the same extension node e_i so no more tetrahedra are needed. In the second case, Figure 3.18 we observe that there are two different extension nodes e_1 and e_2 such that the extension becomes closed with after the first stage of step 2.

As we noted before, in three dimensions, we have to use pyramids as extension elements when the interface is formed by quadrilaterals. This is frequently the case in problems with boundary layers. The procedure for the HERMESH method is the same as has been described previously. The quality criteria used for pyramids is the average of the quality criteria of the two tetrahedra in which the pyramid could be decomposed. If the extension elements associated to one fringe node of the quadrilateral boundary need for *step 2* described before, the elements to close the extension will be tetrahedra. In Figure 3.19 we show the two different types of extensions in three-dimensional problems, tetrahedra on the left and pyramids on the right.

It is worth mentioning that if the fringe node belongs to a boundary layer, when we create the extension elements we have to consider the anisotropy and respect the same aspect ratio for the new extension elements, as is illustrated in Figure 3.20. On the left part we can see the pyramid extension element adapted to the boundary layer where it is created; on the right part of the

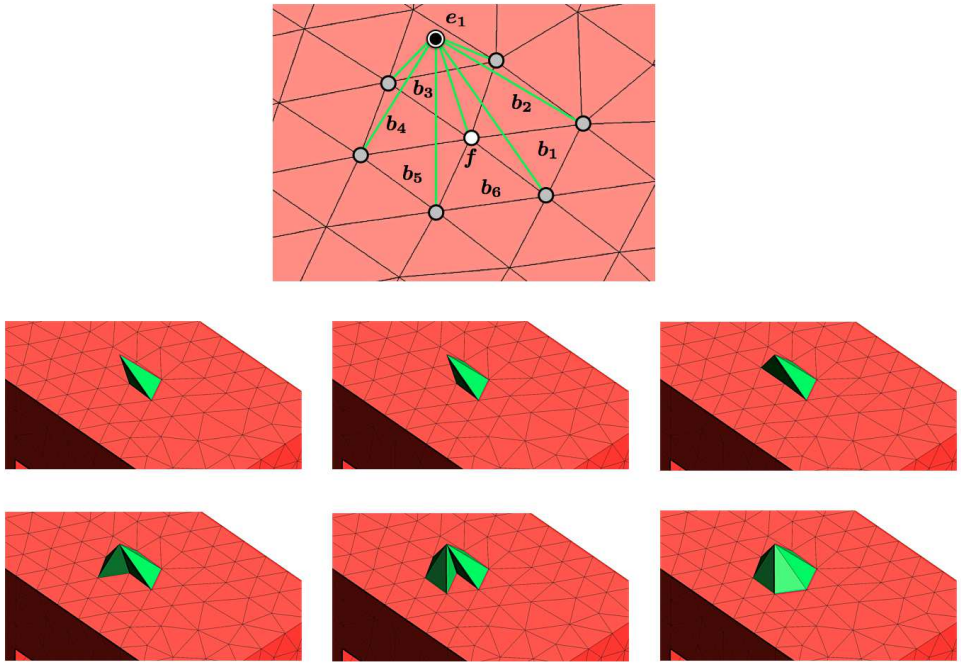


Figure 3.17: 3D extensions process. For one fringe node f with one extension node e_1: Only Step 1. 3D extensions process

same figure, we can see the pyramid extension elements constructed following only the quality criterion without taking into account the aspect ratio of the boundary layer.

3.6 Implementation

In this section we treat some details of the implementation of the method which we consider to be relevant for an understanding of the method. We are going to devote special attention to the parallel issues related to the HERMESH method.

3.6.1 Element assembly

The following is specific to the finite element method with an element-based assembly. Once the extensions have been created, a very simple implementation is possible. It is probably not the most efficient one, but the one that will certainly not degrade the comprehension of the resulting code. The extension elements are attached to a given fringe node and should only be used to assemble the fringe node equation. Therefore, something must be done during

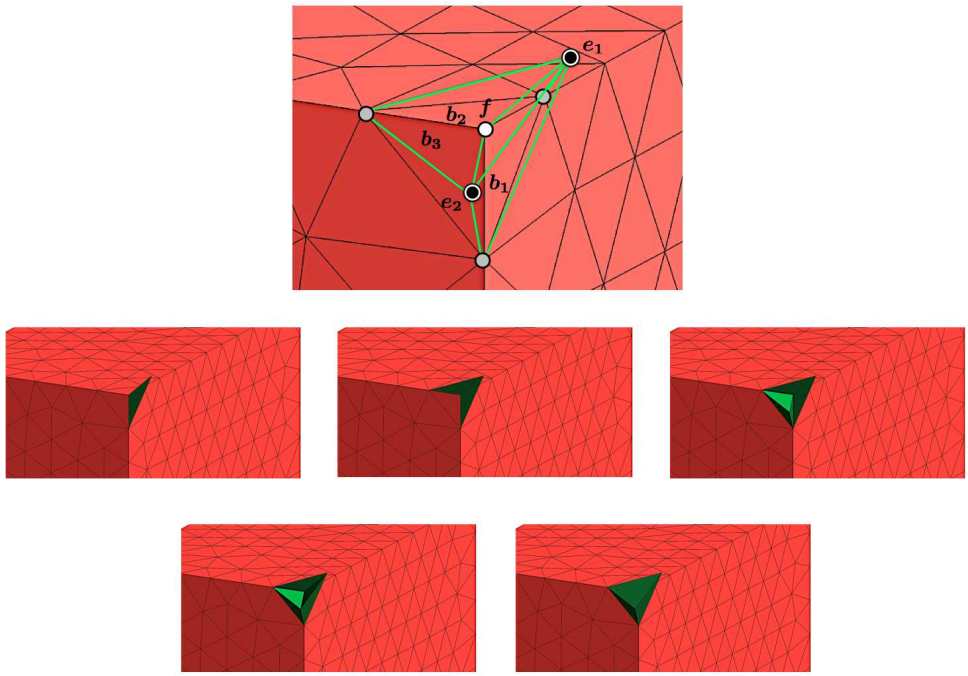


Figure 3.18: 3D extensions process. For one fringe node f with closing and two extension nodes e_1 and e_2: Step 1 and 2.

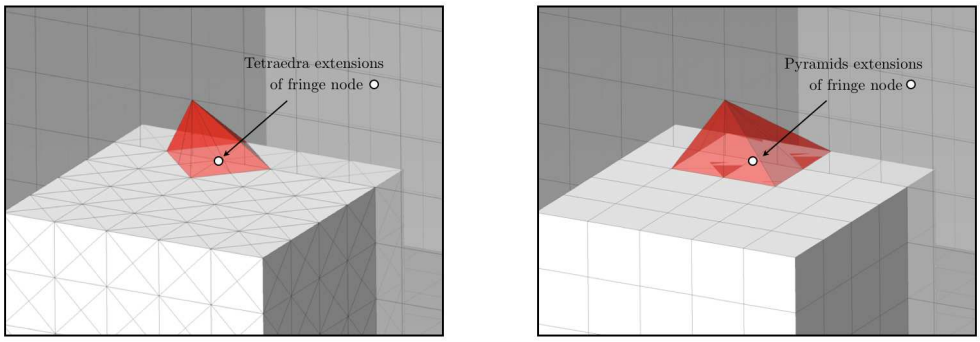


Figure 3.19: Extension elements in 3D. (Left) Tetrahedra to extend from a triangle boundaries. (Right) Pyramids to extend from quadrilateral boundaries.

the assembly process.

Let us create an array `letyp(nelem)` where `nelem` is the total number of

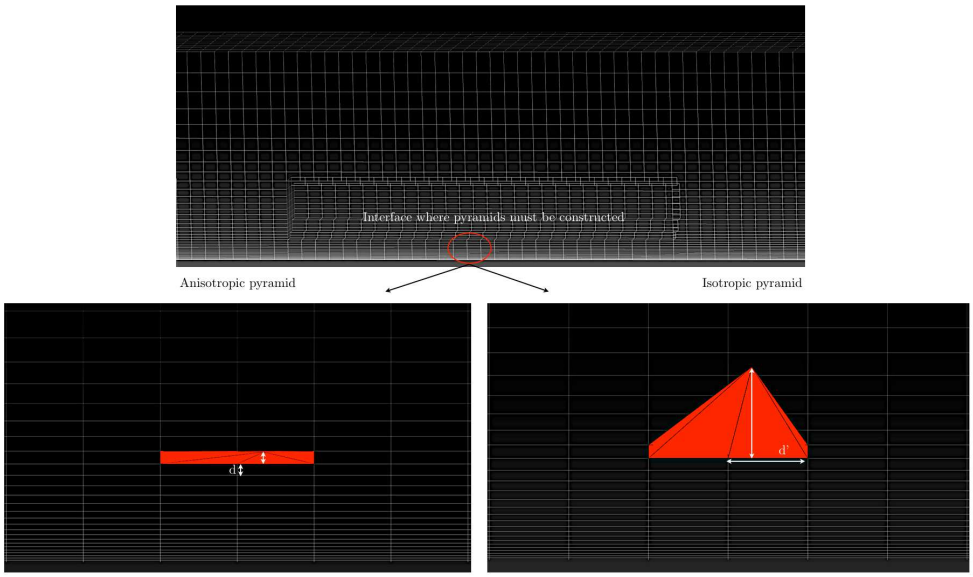


Figure 3.20: Example of pyramid extension in a boundary layer (Left) Pyramid maintaining the aspect ratio, i.e., anisotropy. (Right) Isotropic pyramid without to consider the aspect ratio of the boundary layer.

elements (including extensions). Say this array is:

$$\begin{cases} \text{Normal element:} & \texttt{letyp(ielem)} = \texttt{0}, \\ \text{Extension element:} & \texttt{letyp(ielem)} = \texttt{1}. \end{cases}$$

Let us put the fringe node `ipoin` in the first place in the element connectivity, `lnods` presented before in Section 3.2, when we deal with the implementation of a extension elements. For example, if `ielem` is an extension element of the fringe node `ipoin`, the connectivity would be `lnods(1,ielem)` = `ipoin`. A classical assembly process in a finite element code (not edge-based) consists in looping over the elements, computing the elemental LHS (matrix) and RHS (vector) and then gathering the result into the global matrix and RHS. Now, if we deal with extension elements, what we need is only to set to zero all the rows of the LHS and RHS except for the first one. This task is illustrated in Figure 3.21 for the element assembly of a scalar equation for triangular elements. The code inside the conditioning is the only specificity of the HERMESH method.

3.6.2 Extension on a real boundary

This point concerns fringe nodes on which a Neumann type boundary condition is imposed. The issue is depicted in Figure 3.22. On the top and bottom figures, we have picture two options for the extension elements of the fringe node f. In

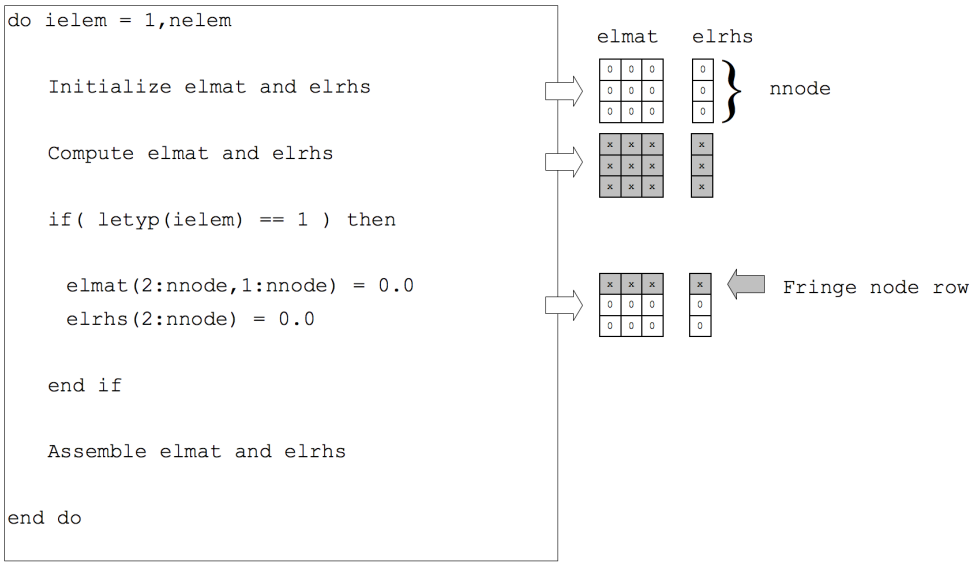


Figure 3.21: Implementation of the HERMESH method for a scalar equation.

order to make sure that the extension elements are closed with the Neumann boundaries of the other subdomain, we construct the extension of a fringe node which belongs to a real boundary like the bottom figure. The problem with the top option is that we would introduce a 45° artificial boundary. When imposing a zero flux Neumann condition, we would therefore perpendicularize the isovalues near the fringe node with respect to the extension outer boundary. Although this problem does not exist with Dirichlet boundary conditions, the candidate nodes associated to a fringe node located in a real boundary have to be located in the real boundary in order to avoid the mentioned problem associated to the Neuman conditions.

3.6.3 HERMESH breaks symmetry

This issue is studied for Navier-Stokes equations solved with Schur complement preconditioner. One very important aspect is that the HERMESH method presented here does not lead to a directed graph of the nodal connectivity (represented by the CSR format used for the assembly of the matrices). A directed graph connects vertices through edges in a symmetrical way, contrary to undirected graphs. More precisely, the graph created by the HERMESH coupling is said to be hybrid, in the sense that it is both directed and undirected. In fact, when a fringe node is connected to an extension node via an extension element, only the equation of the fringe node is assembled in the global system, as we have explained before. Therefore, in the matrix, the row of a fringe node has non-zero coefficients in the columns of its extension nodes. However, these

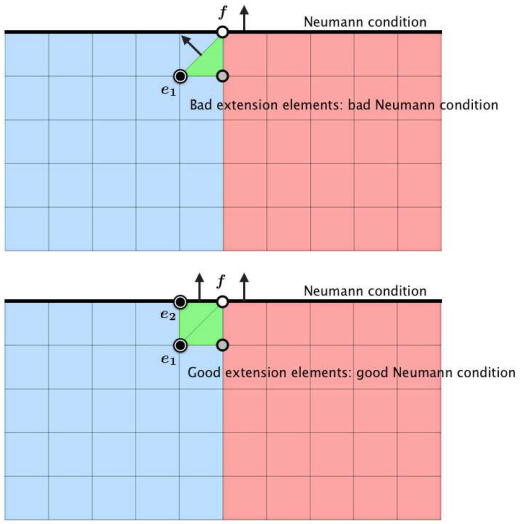


Figure 3.22: Problem when a fringe node is on a Neumann boundary. (Top) Bad extension elements. (Bot.) Good extension elements.

extension nodes are not necessarily themselves connected to the fringe node. The graph is therefore locally directed, as the relation between a fringe node and an extension node is not symmetrical. This is illustrated in Figure 3.23, where the row of the fringe node is coloured according to its existing connectivities. On the other hand, the extension nodes are not connected to any node in subdomain 1.

If the matrix graph does not have a symmetrical structure, then the resulting matrix will obviously not either. Therefore, and unintuitively, the pressure matrix $\mathbf{A}_{pp}$ and the preconditioner $\mathbf{Q}$ are no longer symmetrical. $\mathbf{A}_{pp}$ is not an issue. The problems stems from the iterative solution of the Schur complement preconditioner $\mathbf{Q}$ presented in chapter 2, in section 2.2.4 and involved in Step 3 of Algorithm 1 therein.

If one wants to use an efficient iterative solver for symmetrical systems like the Conjugate Gradient (CG) or the Deflated Conjugate Gradient (DCG), the matrix should be symmetrized. Remember that $\mathbf{Q}$ is only a preconditioner; thus should the solution procedure of the Navier-Stokes equations converge, then it will converge to the same solution regardless of $\mathbf{Q}$. The symmetrization can be simply achieved. We propose four simple options, illustrated in Figure 3.24. On the left part of Figure, the element matrix of an extension element is represented for triangle elements (3 degrees of freedom per element). In order to obtain a global symmetrical system, the extension element matrices should be symmetrical as well. The first option, referred to as "*all*-preconditioner" consists in maintaining the whole element matrix, which is obviously symmetrical

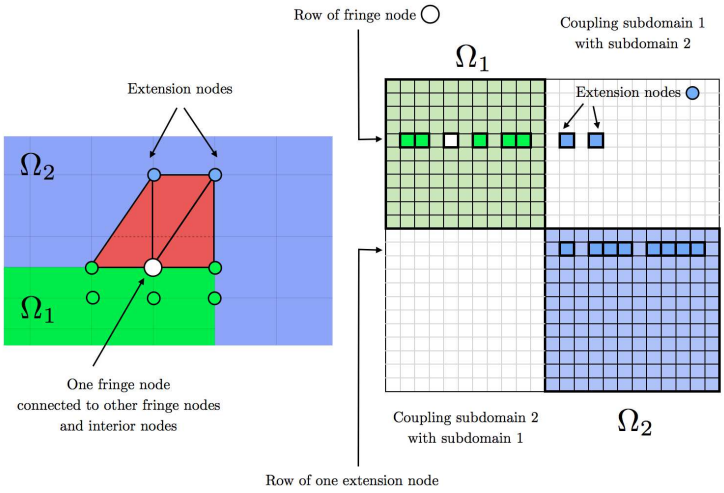


Figure 3.23: Matrix graph is not fully undirected. A fringe node is connected to extension nodes, but these ones are not connected to the fringe node.

as it comes from the assembly of Equation 2.51. The second option, referred to as "*symmetrized*-preconditioner" consists in assembling only the fringe node row and the fringe node column. In the third option, referred to as "*zero*-preconditioner", the extension element matrix is not assembled at all. Finally, the fourth option, "*diag*-preconditioner" consists in retaining only the diagonal. We will show that the best option is the first one, which is the one that retains the whole extension element matrix. In order to use symmetrical itera-

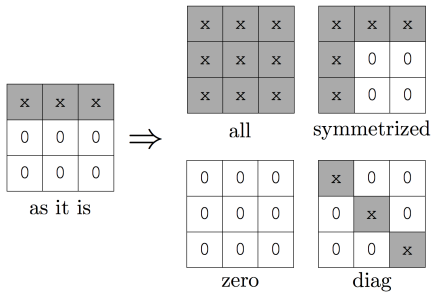


Figure 3.24: Schur complement solver. Four strategies for obtaining a symmetrical pressure Schur complement preconditioner.

tive solvers like the CG or the DCG, the system obtained with the HERMESH method should be symmetrized. In the following example we want to compare the four different strategies proposed here, namely all, symmetrized, zero, diag. The example is the flow over a square cylinder at Reynolds 10 based on the

square length and inflow velocity. A symmetry condition is applied on the top and bottom. The cylinder is meshed with an independent mesh and is coupled with another mesh representing the rest of the domain.

We first compare the four convergence histories of the continuity equation. In this example, the Orthomin(1) (Algorithm 1 presented in section 2.2.4) linearization and time iterations are coupled. That is for each time step, only one Orthomin(1) and linearization iteration step is carried out. At iteration k the residual is measured as the expression given by 2.50.

Figure 3.25 (Left) shows the convergence of the different options. We observe that assembling the whole extension is by far the best option. The same behavior has been observed in many other examples, so we conclude that this is the best way to construct a symmetric preconditioner for the pressure Schur complement.

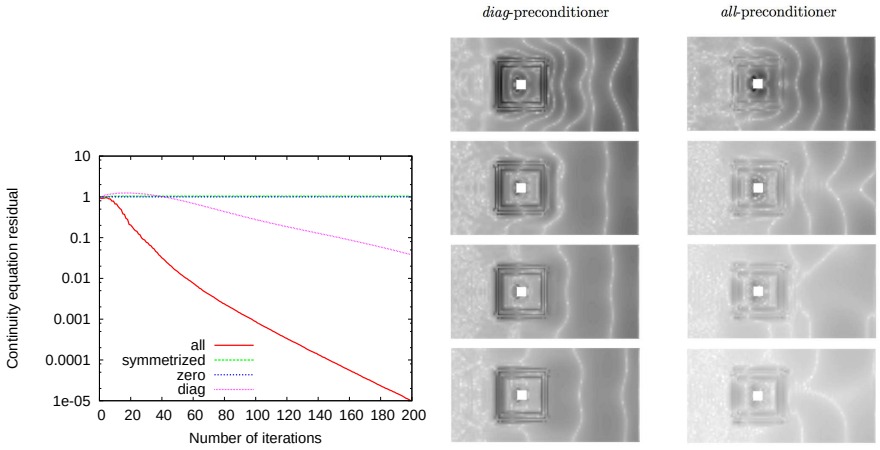


Figure 3.25: Schur complement preconditioner. (Left) Convergence history of the continuity equation. (Right) Evolution of the residual of the continuity equation: *diag*-preconditioner in the left column and *all*-preconditioner in the right column.

Let us observe the damping of the continuity equation residual. Figure 3.25 (Right) shows the evolution of this residual for some consecutive iterations, for both the *diag*-preconditioner (left column) and *all*-preconditioner (right column). We observe that with the first method, the error is much more concentrated near the interfaces and damped out much more slowly than with the *all*-preconditioner.

With respect to the fill-in of the matrix, Figure 3.26 illustrates the new connection obtained with the extension elements, for this example.

We have just mentioned the way to obtain a symmetric Schur complement preconditioner $\mathbf{A}_q$. In the case of the other matrices, i.e. $\mathbf{A}_{uu}$, $\mathbf{A}_{up}$, $\mathbf{A}_{pu}$ and $\mathbf{A}_{pp}$, the same procedure as the one described in Section 3.6.1 is employed, as

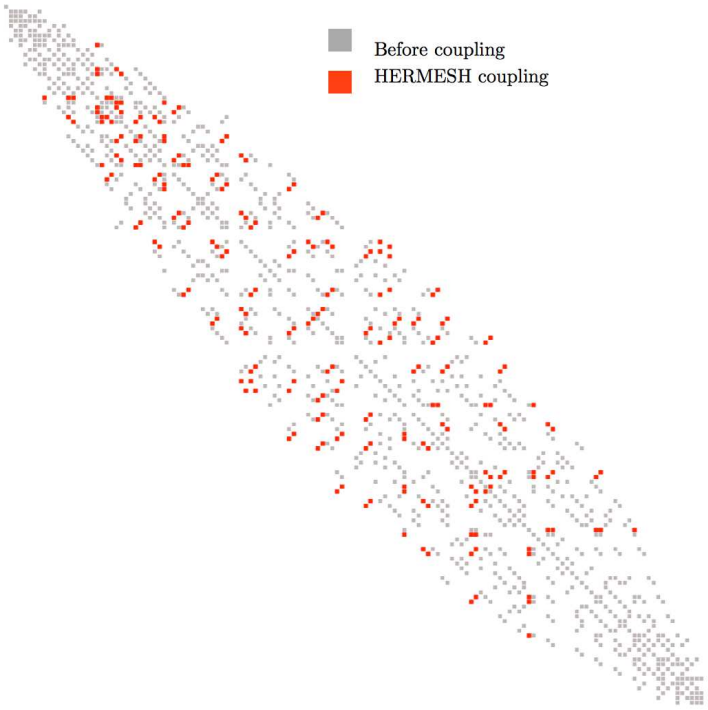


Figure 3.26: Schur complement preconditioner. All elemental contributions of extension elements are assembled.

$\mathbf{A}_{uu}$ is unsymmetrical in any case (assuming we have convection). The procedure is illustrated in Figure 3.27, where `ndime` is the dimension of the problem.

3.6.4 Parallelization

This section deals with the critical points of the parallelization method when HERMESH is applied. Before discussing the relevant aspects related to the parallelization with the HERMESH method, we would like to make clear the nomenclature through a simple picture shown in Figure 3.28. We have adopted the term *interface* to the boundary between one independent mesh to the other, while the term *boundary* is used to refer to the boundary between different processors also called CPU's. The boundary of the whole domain is referred by us as *real boundary*.

Within the modular structure of the Alya code described in Section 2.7, the HERMESH method corresponds to a service in such a way that the method could be valid for any of the modules corresponding to different physics inside the code and could be run in parallel maintaining the same performance but with some particularities. More precisely, the HERMESH service is a prepro-

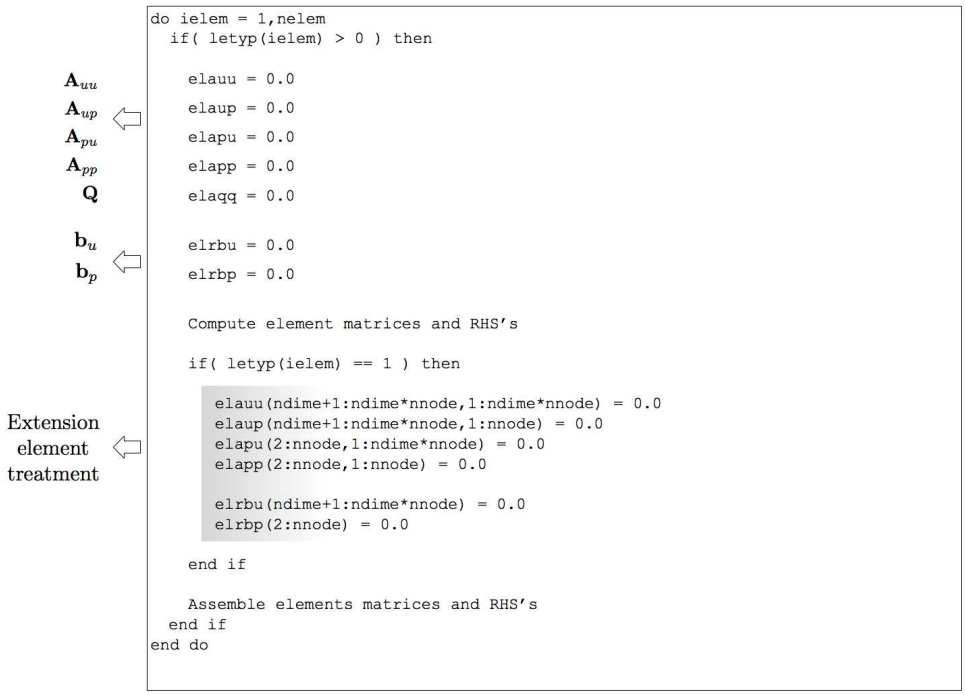


Figure 3.27: Implementation of the HERMESH method for the Navier-Stokes equations. Extension element is completely assembled for **Q** .

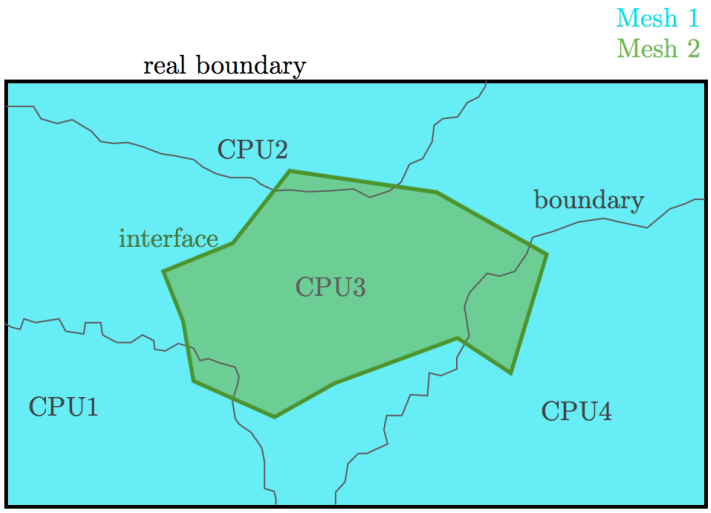


Figure 3.28: Nomenclature

cess step in the code.

As we have explained in that Section 2.7, for mesh partitioning, METIS requires the element graph and to construct this, there are two possible options, referred to as *by-nodes* or *by-faces*. Although the latter strategy requires much less memory, when we use extension elements, we have to apply the first strategy to divide the mesh, as illustrated in Figures 3.29 for a 2D mesh. We want to remember that fringe nodes are connected to the extension nodes via elements and not faces. This particularity leads to a problem when METIS divides the mesh if the graph is given *by-faces*. This is because METIS has no way to know the neughbors of the extension elements on the side where they extend.
We have run a test in order to evaluate the differences between both options in mesh partitioning with the HERMESH method, *by-faces* or *by-nodes*. The test corresponds to a cube domain formed by two meshes, one inside the other with 374832 tetrahedra in total. After the creation of the extension elements to couple both meshes, we have divided the composite mesh into 63 processors and with both classes of graphs. We show three important issues in the division performed by METIS for this problem through Figure 3.30:

1. The top graphic represents the number of elements in each processor. We can observe in it that there is one processor, the number fourteen, with much more elements than the others which will imply a load imbalance.

2. The middle graphic represents the number of boundaries between processors. Again we observe that processor fourteen contains much more boundary nodes, which implies a big size in the communications and in turn this slows down the point-to-point communication steps (e.g. matrix-vector-product).

3. The bottom graphic represents the number of neighboring processors for both cases. We observe that in average there are more neighbors with the graph given *by-faces* (which could imply more time in the communications, depending on the length of the communications if one consider that data transfer time is given by $t = na + \sum_{i=1}^{n} l_i b$, where the first term corresponds to the latency or startup time and the second one depends on the messages size).

To analyze all these questions at more depth, we show a trace visualized with the program *paraver*, PARAVER (2014). Paraver describes in a graphical way the status of the processes involved in the parallel execution of the code. The x-asis is time, and the y-axis shows, according to a specific color, the status of each process. In particular, blue color means idle process. Figure 3.31 compares the parallel performance of the code using *by-nodes* and *by-faces* graphs. The top graphics of the figure represents the assembly process

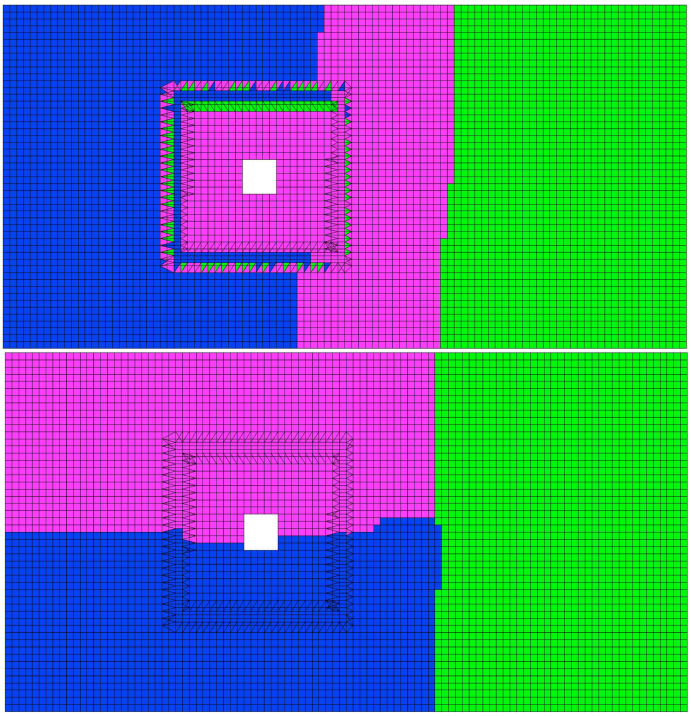


Figure 3.29: 2D Mesh partitioned into 3 subdomains. (Left) Element graph based on nodes. (Right) Element graph based on faces.

(loop over the elements painted in green) while the bottom graphics show some matrix-vector products of the GMRES algebraic solver (painted in red). The top figures confirm the load imbalance already observed in Figure 3.31 (Top). We note that the assembly step in process 14 is longer than that of other processes. On the other hand, we observe that using the *by-nodes* graph, five matrix-vector products can be carried in the same time as the *by-faces* graph can only perform three of these. This is due to the fact that in average, a process has to communicate with much more processes, as already shown by Figure (Bottom). These communications are depicted with yellow arrows in the bottom right part of the figure.

With Figures 3.30 and 3.31, we have shown that the *by-faces* graph has several shortcomings: (1) Load imbalance; (2) More communications. The reason is that the *by-faces* graph gives a wrong information to METIS. Let us try to explain why. The *by-faces* graph connects elements which share faces. We know that in a normal finite element mesh, a face is shared by at most two elements. But the HERMESH method (as it is implemented in this work) can construct lots of extension elements from a single face. It means that these elements can

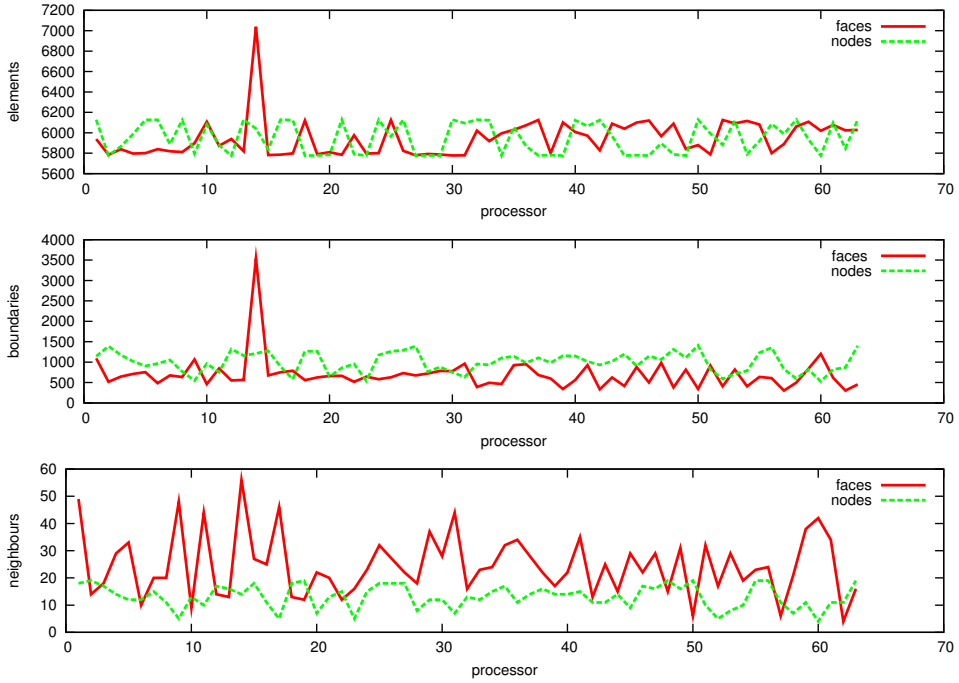


Figure 3.30: (Top) Number of elements by process. (Middle) Number of boundaries by process. (Bottom) Number of neighbors subdomains by processor.

end up lost in the graph, unconnected to any other element. That said, we know that, using the element graph information, METIS tries to balance the work by equalizing the number of elements per process while minimizing the boundaries between the processes. Therefore, for METIS, this lost elements do not communicate, enabling METIS to give lots of these elements to some processes. But, this elements do involve communication, if their nodes are located on the process boundaries. In the present example, we observe that process 14 inherit much more elements than others.

Let us finally note that processes that have hole elements present a load imbalance in the assembly step, as these hole elements are not assembled. This is confirmed by Figure 3.31 (Top). However, the corresponding degrees of freedom do exist and participate to the matrix-vector product, as seen in Figure 3.31 (Bottom). This is why we do not observe such load imbalance in the matrix-vector product (although the corresponding matrix rows are null), and useless work is carried out.

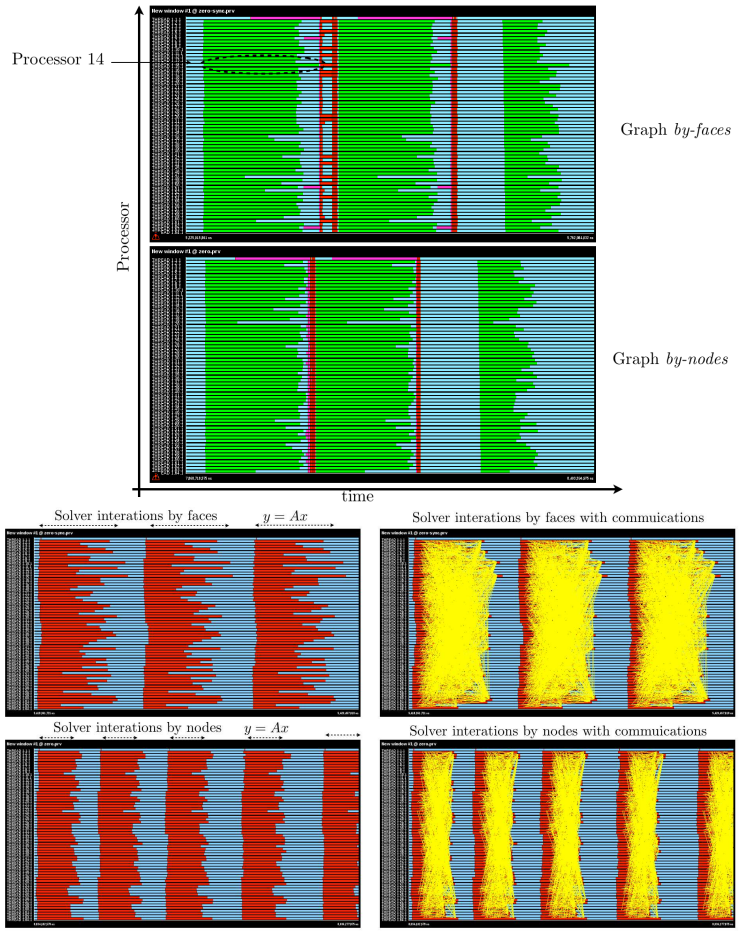


Figure 3.31: Trace extracted from *paraver* program.

3.6.5 Possible Improvements

- As we noted before briefly, it should be pointed out that, due to the way the extension elements are constructed, some extension elements are very likely to be repeated. For example, in Figure 3.13, the leftmost extension triangle would also be an extension element of the gray node located on the left of the white node. Therefore, these elements could be collapsed in order to save memory and time. The assembly as explained in the previous subsection should therefore be slightly modified. In this example, if the leftmost extension element is shared by these two fringe nodes, then the elemental LHS and RHS should be assembled for the two respective rows.

- Another possible improvement from the point of view of parallelization is that of minimizing the communications between the subdomains, at these places where boundary nodes are also fringe nodes. Let us take a look at Figure 3.32. On the left part, the green fringe node of subdomain 2 is connected to subdomain 1 through an extension element belonging to subdomain 1. Remember that the equation coming from an extension element is only assembled for the corresponding fringe node (see section 3.6.1). This means that subdomain 1 will assemble the fringe node equation coming from the extension element while subdomain 2 will assemble the equation coming from the leftmost element of subdomain 2. After a matrix-vector product, an exchange (send and receive) is therefore necessary to obtain a global result on the fringe node, coming from the contributions of both subdomains to the equation of the fringe node for $u_{f,i}$. Now let us see what happens if the extension element belongs to subdomain 1, as depicted in the right part of Figure 3.32. This subdomain is able to assemble the complete equation for the fringe node. It only requires the value of u_i from subdomain 1. On the contrary, the equation of node (i) in subdomain 1 does not depend at all on subdomain 2. This means that subdomain 1 does not require any value from subdomain 2. However, subdomain 2 needs the value of u_i from subdomain 1 in order to perform the matrix vector product of u_f row. Thus, in this case, we only need one communication from subdomain 1 to subdomain 2, while in the first case (left part of the figure), we need two communications. The parallelization in Alya is based on a send/receive as explained in section 2.7. Therefore, in the situation represented by the right part of the figure, we would carry out one unnecessary communication from subdomain 2 to subdomain 1.

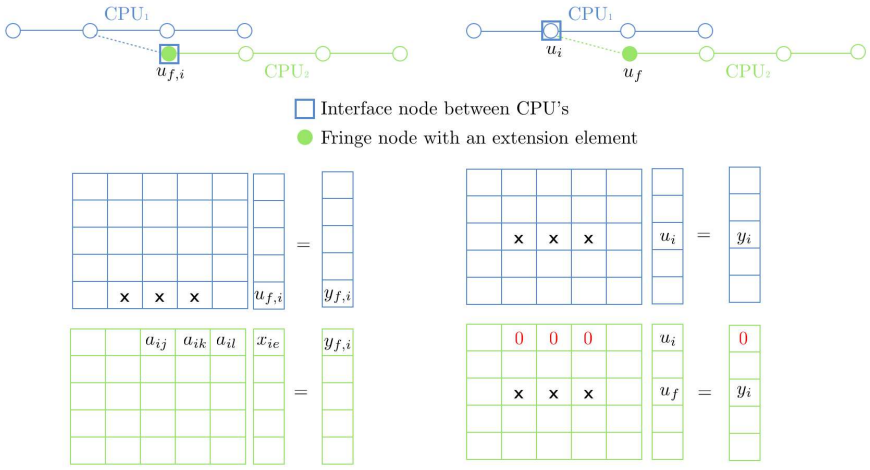


Figure 3.32: Fringe node as a boundary node between processors.

- One possible improvement related to the question of the type of graph used by METIS is a hybrid option, such that METIS divides the mesh *by-faces* if the element is normal (i.e. `letyp(ielem) = 0`) and *by-nodes* if the element is an extension element (i.e. `letyp(ielem) = 1`).

- Another issue concerns the matrix graph. As illustrated in the 1D example (Figure 3.12), the fringe node is connected to the extension nodes through the extension element. However, only the row of the fringe node is assembled. Therefore, we will obtain null coefficients in the row of the extension nodes at the place of the fringe nodes. The nodal graph (used for example for the CSR format of sparse matrices) could therefore be reduced by cancelling these positions.

3.7 Quality Criteria for extension elements

As any other element of a finite element mesh, the priority criteria for constructing the extension elements is the geometric quality although this could be a non-trivial issue. There are several parameters for assessing the quality of an element, and the expression depends on the type of element. A good summary of tetrahedra measures and a global definition of the tetrahedron shape measure is given in Dompierre, Labbé, Guibault, & Camarero (1998). Let us present just a brief mention of these classical tetrahedron shape measures:

- **The Radius Ratio** ρ. The radius ratio ρ of a tetrahedron T is defined to be $\rho = N\rho_{in}/\rho_{out}$ where ρ_{in} and ρ_{out} are the in-radius and the circumradius of T, respectively and N is the dimension of the space.

- **The Solid Angle** Θ_{min}. It is defined as follows:$\Theta_{min} = \alpha \min_{0\leq i\leq 3} \Theta_i$ where $\alpha^{-1} = 6\arcsin(\sqrt(3)/3) - \pi$ and $0 \leq \sum_{i=0}^{3} \Theta_i \leq 2\pi$.

- **The Dihedral Angle**. The minimum dihedral angle is a defined as: $\Phi_{min} = \alpha \min_{0\leq i\leq j\leq 3} \Phi_{ij} = \min_{0\leq i\leq j\leq 3}(\pi - \arccos(n_{ij1} \cdot n_{ij2}))$ where n_{ij1} and n_{ij2} are the two triangular faces adjacent to the edge ij and $\alpha^{-1} = \pi - \arccos(-1/3) = 1.230959$ is the value of the six dihedral angles of the regular tetrahedron.

- **The Edge Ratio**. The ratio between the minimum and maximum edges l_{ij} of the tetrahedron. $r = \min_{0\leq i\leq j\leq 3} l_{ij} / \max_{0\leq i\leq j\leq 3} l_{ij}$. But this is a not a proper tetrahedron shape measure and it fails to detect some degenerated tetrahedra.

- **The Aspect Ratio**. The ratio of two characteristic sizes of the tetrahedron. In particular, in our work we use the ratio between the maximum edge l_{ij} of the tetrahedron and the minimum height h_i calculated as the

distance between a vertex and the center of the gravity of the opposite face.

$$Q = \max_{0 \leq i \leq j \leq 3} l_{ij} / \min_{0 \leq i \leq 3} h_i \tag{3.13}$$

In our work we have also used the next quality measure coming from Gamma Team, see George (1997):

$$Q = \alpha \frac{h_M}{\rho} = \alpha \frac{h_m S}{V}, \tag{3.14}$$

where $h_M = max_{i=1,6} h_i$, h_i being the length of edge i of the element under consideration, ρ is the in-radius, $S = \sum_{i=1,4} S_i$, S_i being the surface of face i, and V is the element volume.
Apart from these nodal-based objective functions, there are also other interpretations using matrices and matrix norms. This matrix point of view suggests several different objective functions such as, for example, the smoothness objective functions in terms of the **condition number of Jacobian matrix**. See references like Freitag & Knupp (1999), Freitag & Knupp (2002),Freitag, Plassmann et al. (2000),Knupp (2000b),Knupp (2000a) related to this question. The expression of the condition number of Jacobian matrix is given by:

$$\|K\|_2 = \left[\sum_{m=0}^{M-1} \kappa_m^2 \right]^{1/2}. \tag{3.15}$$

To evaluate the importance of the quality of the mesh and the role of its different criteria, we have checked the mesh convergence modifying this parameter. For this, we consider the solution of the diffusion equation in a unit cube $\Omega = [0,1] \times [0,1] \times [0,1]$ and we have applied the manufactured solution technique explained in 2.6 with the following analytical exact solution:

$$u_e = \sin(\pi x)\, \sin(\pi y)\, \sin(\pi z)\, \exp(xyz). \tag{3.16}$$

We consider four meshes with lengths $h, h/2, h/4, h/8$ and two subdomains in a Chimera-type problem. The background mesh is Ω_1 with a spherical hole of radius 0.2 and the patch mesh located inside is a spherical subdomain Ω_2 of radius 0.21 to ensure a minimum overlap. Three criteria have been used for the creation of the extension elements. They are the Gamma Team criterion given by Equation 3.14 ; the condition number quality criterion done by Equation 3.15 and the aspect ratio quality criterion done in Equation 3.13. In addition, for the Gamma Team criterion, two qualities have been selected: the "best quality" and the "worst quality". Figure 3.33 shows the extensions obtained for subdomain Ω_2 using both qualities for the Gamma Team criterion. We observe very distorted elements for the "Worst quality" criteria. To have an order of magnitude, the $h/8$ mesh has a total of 6.5M of elements including 2% of extension elements.

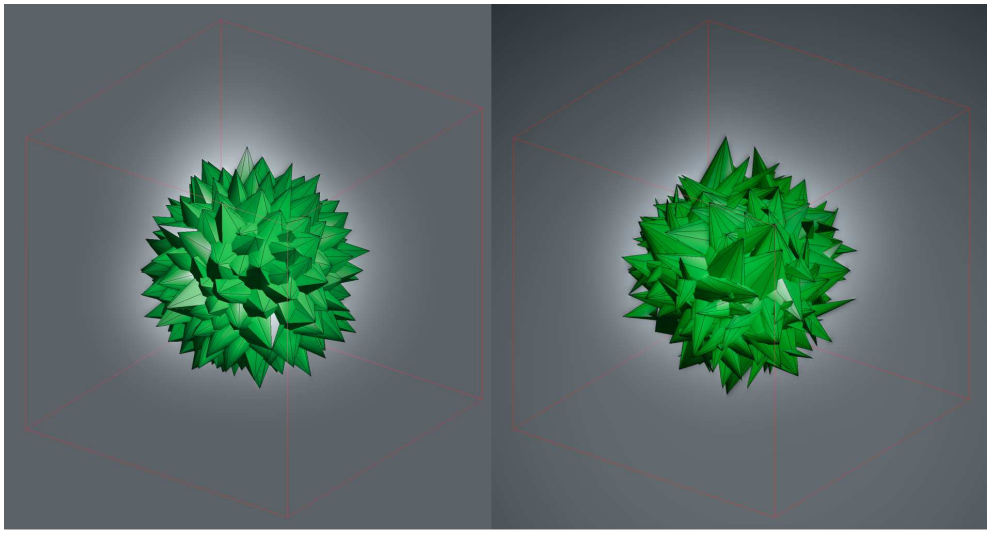


Figure 3.33: Extension elements of interior subdomain Ω_2 using Gamma team criterion. (Left) Best quality criterion. (Right) Worst quality criterion.

Figure 3.34 shows the L^2 error norms of the solution and its gradient. They confirm the good asymptotic behavior of the solution as well as the fact that the Best quality gives better results than the Worst quality criterion for the Gamma team criterion. Finally, the bottom part of Figure shows that for this example, all quality criteria yield similar results.

3.8 Interface smoothing

The domain decomposition coupling we propose is geometrical, as has been shown in Section 3.4. It is therefore important to ensure a minimum regularity of the interfaces and the mesh nearby, as this will affect the quality of the results, as is shown in Figure 3.34. So, it could sometimes be useful to previously smooth the interfaces where the crown of extension elements will be constructed. This requirement is more frequently done in Chimera-type problems when the hole is created, in particular if the mesh where the hole is constructed is structured and the geometry of the patch located inside does not coincide with the regularity of the mesh, as it will be shown below.
The algorithms for mesh improvement can be divided into three basic categories:

- Point insertion/deletion to refine or coarsen meshes.
- Local reconnection or face swapping to change mesh topology for a given set of vertices.

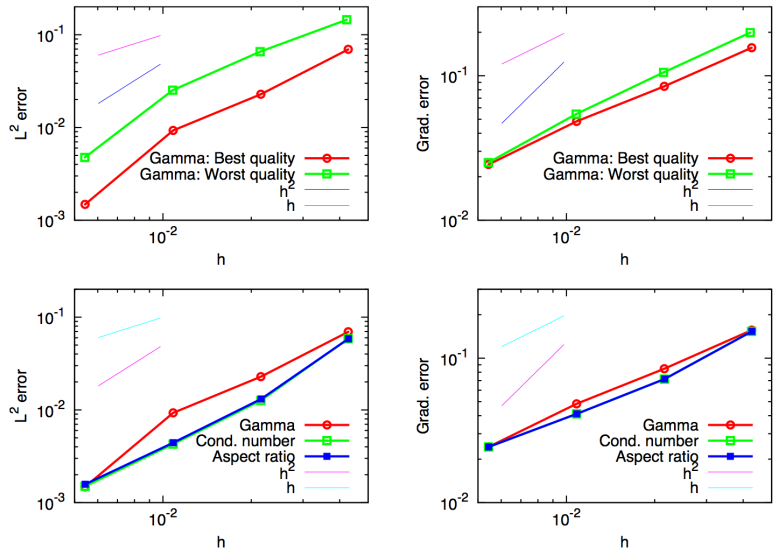


Figure 3.34: Error with respect to exact solution given by Equation (3.16). (Top) Gamma team criterion with best and worst qualities. (Bot.) Gamma with best quality criterion, condition number criterion and aspect ratio criterion. (Left) L^2 error: solution. (Right) L^2 error: gradient.

- Mesh smoothing to relocate grid points to improve mesh quality without changing mesh topology.

In this work, the last category is treated. Basically, mesh smoothing falls into two major groups:

- Local, so that nodes are moved one by one.
- Global smoothing that simultaneously changes all the nodal locations in the mesh.

Local mesh smoothing algorithms have been shown to be effective in repairing distorted elements in meshes and we have used them in our work.

The most commonly used smoothing technique is Laplacian smoothing, see Lo (1985), which moves a given node to the barycenter of all the nodes connected with it. This method is computationally inexpensive but it does not guarantee an improvement in mesh quality, it being possible to create inverted or invalid elements. More drawbacks related to this Laplacian approach include convergence problems and shrinking issues of the mesh. A variant of this algorithm was developed, the so-called constrained-Laplacian smoothing, see Freitag (1997) for more detail. It overcomes this problem by placing a node at a new location only when the mesh quality is improved. This method successfully prevents the degradation of mesh or place nodes at their best locations.

In Taubin (1995b) the author reduces the problem of surface smoothing to low-pass filtering. This approach is based on the observation that the classical Fourier transformation of a signal can be seen as the decomposition of the signal into a linear combination of the eigenvectors of the Laplacian operator. This iterative algorithm is linear both in time and space, simple to implement, and produces smoothing without shrinkage.
Other smoothing strategies include the so-called optimization-based smoothing algorithms. These algorithms integrate certain mesh quality measures into objective functions. These schemes depend on the type of the mesh, the optimization method used as well as the measure of a mesh quality to construct the objective function, this being a key point. One has to define a proper objective function which is closely related to mesh quality measures, treated in Section 3.5.2. Knupp and Freitag have brought about mesh quality improvement algorithms for 2D and 3D linear elements, which have to be treated separately since some matrix norm identities can be varied depending on the rank. Freitag & Knupp (2002) proposed a low cost, optimization-based alternative to Laplacian smoothing that guarantees valid elements in the final mesh if the initial mesh is valid. Most of the existing objective functions used in optimization-based smoothing schemes cannot guarantee a converged solution for an invalid mesh. For this reason, untangling techniques as is shown in Freitag et al. (2000) have been proposed to remove invalid elements from the mesh before executing optimization-based smoothing. The authors use this fact to formulate the solution to $\max\, q(x) = \max \min_{1 \leq i \leq n} A_i(x)$ as a linear programming problem which they solve via the simplex method. On each sweep, m linear programs are solved, which sequentially moves each interior node in the mesh, and this process is made until the mesh is untangled or a maximum number of sweeps has occurred. The problem is well posed if the vertices of the subproblem do not all lie in a lower-dimensional subspace in relation to the original problem and none of the vertices are collocated in the same point in the space. Chen, Tristano, & Kwok (2004) propose an objective function for their Laplacian/optimization smoothing scheme for both linear and quadratic triangular and quadrilateral element that can be used to untangle and smooth a mesh in a single process.

It should be noted that since finite element mesh smoothing is used to improve the quality of finite element analysis (FEA) results, another way to assess mesh quality is to examine the relationship of mesh quality with the FEA results. Shewchuk (2002) has studied this relationship. His work showed that the conditioning of the stiffness matrices depends on element shape. And it also showed the connection between mesh geometry, interpolation error, and stiffness matrix conditioning. The author expresses these relationships with error bounds and element quality measures that determine triangle element fitness for achieving low condition numbers.

In our work, our objective is to achieve a more regular interface to couple different subdomains. For this reason, we have applied a surface Laplacian-smoothing algorithm based on Taubin (1995a) with the same parameters λ, μ and the number of iterations *numiter* chosen is 7. This algorithm is represented in Algorithm 2. Another version of the same algorithm is presented in Algorithm 3 and Algorithm 4 , where we check if the movement of each free node entails a worse condition number than a certain reference value, regarded as the worst condition number of the original mesh.

Algorithm 2 Laplacian smoothing general algorithm

1: Mark to not move points in the interface and located in the real boundary.
2: **(INPUT)**:

- $\lambda = 0.6307$, $\mu = 1.0/(0.1 - 1/\lambda)$
- *numit*: number of iterations

3: **(INITIALIZE)** $k = 0$, $x^k = x$.
4: **while** $k \leq numiter$ **do**
5: **for** Laplacian loop: Points to be moved with free coordinates (x) **do**
6: Compute the barycenter: b_i with the connected points to be moved.
7: Move the coordinate (x^k): $x^k = x^k + \lambda(b_i - x^k)$
8: **end for**
9: **for** Anti-Laplacian loop: Points to be moved with free coordinates (x^k) **do**
10: Compute the barycenter b_i.
11: Move the coordinate (x^k): $x^k = x^k + \mu(b_i - x^k)$
12: **end for**
13: **end while**

As a consequence of this Laplacian surface smoothing, the quality of the volume mesh is perturbed, so we have to relocate the volume-nodes with the aim to repairing the poor quality elements resulting from the above surface smoothing. To do this, we have applied a tetrahedron mesh improvement via optimization of the element condition number developed in Freitag & Knupp (1999) and this algorithm is sketched in Algorithm 5.

This optimization uses a steepest descent framework with a modified line search adapted to the geometrical constraints of the sub-mesh $t_i, i = 1, \ldots, m$ associated to the free point. For more details about this issue, see Nocedal & Wright (2006). The implemented line search satisfies the Armijo rule which guarantees the local convergence of the method. A structured strategy is applied in order to perform the line search. The descent direction is obtained using the gradient of the following objetive function $f(x)$, in which the free

Algorithm 3 Laplacian constrained smoothing algorithm

Mark to not move points in the interface and located in the real boundary.
(INPUT):

- $\lambda = 0.6307$, $\mu = 1.0/(0.1 - 1.0/\lambda)$
- *numit*: number of iterations
- *control*: number of iterations to reduce the moviment.
- $flag$: flag to control if the free vertex can be moved.
- κ_{ref}: Reference condition number.

(INITIALIZE) $k = 0$, $x^k = x, flag = 0$.
while $k \leq numiter$ **do**
  for Laplacian loop: Points to be moved with free coordinates (x^k) **do**
    Compute the barycenter: b_i with the connected points to be moved.
    Move the coordinate (x^k): $x^k = x^k + \lambda(b_i - x^k)$
    (INITIALIZE) $jiter = 0$
    while $jiter \leq control$ **do**
      Move the coordinate with a weight α, (x^k): $x^k = x^k + \alpha\lambda(b_i - x^k)$
      for Tetrahedra $t_i, i = 1, \ldots, m$ connected to the free point **do**
        Compute the maximum condition number of the sub-mesh: κ_{max}
      end for
      if $\kappa_{max} > \kappa_{ref}$ **then**
        Modify the weight to move: $\alpha = \alpha * \frac{(control * jiter)}{control}$
        Move the coordinate with weight α, (x^k): $x^k = x^k + \alpha\lambda(b_i - x^k)$
      else
        $flag = 1$
      end if
    end while
    if $flag = 0$ **then**
      The point cannot be moved.
    end if
  end for
  for Anti-Laplacian loop: Points to be moved with free coordinates (x^k) **do**
    Detailed in Algortihm 4
  end for
end while

Algorithm 4 Continuity of the Laplacian constrained smoothing algorithm

for Anti-Laplacian loop: Points to be moved with free coordinates (x^k) **do**
  Compute the baricenter: b_i with the connected points to be moved.
  Move the coordinate (x^k): $x^k = x^k + \mu(b_i - x^k)$
  (INITIALIZE) $jiter = 0$
  while $jiter \leq control$ **do**
    Move the coordinate with a weight α, (x^k): $x^k = x^k + \alpha\mu(b_i - x^k)$
    for Tetrahedra $t_i, i = 1, \ldots, m$ connected to the free point **do**
      Compute the maximum condition number of the sub-mesh: κ_{max}
    end for
    if $\kappa_{max} > \kappa_{ref}$ **then**
      Modify the weight to move: $\alpha = \alpha * \frac{(control * jiter)}{control}$
      Move the coordinate with weight α, (x^k): $x^k = x^k + \alpha\mu(b_i - x^k)$
    else
      $flag = 1$
    end if
  end while
  if $flag = 0$ **then**
    The point cannot be moved.
  end if
end for

vertex x is the variable:

$$f(x) = \|K(x)\|_2 = \left[\sum_{m=0}^{M-1} \kappa_m(x)^2\right]^{1/2}. \tag{3.17}$$

First, we select the tetrahedron t_j of the sub-mesh which intersects the line defined by the descent direction $p = -\nabla_x f$ and x. After that, we calculate the distance d, between x and the face formed by the rest of the points of this tetrahedron t_j. This distance is expressed in Algorithm 5 by

$$d = \|x^k - y^k\| \quad \text{where} \quad y^k \in \{x^k + sp_k : s \geq 0\} \cap [\cup_{i=1}^m \partial t_i]$$

, where $\cup_{i=1}^m \partial t_i$ represents the boundary of the submesh and the intersection with the mentioned line is the point in the face of the tetrahedron t_j. Using an input parameter n given by the user, we calculate the step $\gamma = d/n$. Finally, the next position of the free vertex is calculated as

$$x_{new} = \text{argmin}_{i=1,\ldots,n}\{f(x + i\gamma p) : x + i\gamma p \in \mathbb{R}^3\}. \tag{3.18}$$

The process is illustrated in Figure 3.35, where the sub-mesh associated to the free vertex to be moved as well as the steepest descent direction is

Algorithm 5 Optimization smoothing algorithm

1: **(INPUT)**:

- (x): free vertex coordinates
- $t_i, i = 1, \ldots, m$: tetrahedra connected to the free point.
- n: partitions
- $maxiter$: maximum number of iterations
- ϵ: convergence tolerance

2: **(INITIALIZE)** $k = 0$, $x^k = x$.
3: **while** $\|\nabla f(x^k)\| > \epsilon$ and $k < maxtiter$ **do**
4: $p_k = -\nabla f(x^k)$
5: $d = \|x^k - y^k\|$ where $y^k \in \{x^k + sp_k : s \geq 0\} \cap [\cup_{i=1}^{m} \partial t_i]$
6: $x^{k+1} = \text{argmin}\left\{ f\left(x^k + l\frac{d}{n}p_k\right) : l = 1, \ldots, n \right\}$
7: $k = k + 1$
8: **end while**

	Maximum	Average	Minimum
Regular Mesh	3.8	3.8	3.8
Perturbed Mesh	160.6	6.1	3.0
Optimized Mesh	7.2	4.0	3.1

Table 3.1: Condition number mesh quality.

illustrated. The red tetrahedron is the tetrahedron of the sub-mesh which intersected the yellow line, defined by the steepest descent direction given by the negative gradient of the objective function.

To verify the smoothing volume technique we have performed the following test. From a regular Cartesian mesh consisting of 3000 tetrahedral elements and 791 nodes in a cube domain we have randomly perturbed the position of all the interior nodes and applied the volume mesh smoothing Algorithm 5, as we can see in Figure 3.36. The statistic of the condition number mesh quality is represented in Table 3.1.

So we observe how the smoothing algorithm corrects the perturbed mesh. In Figure 3.36 these three meshes are represented.
The coupling between the Laplacian surface mesh smoothing and optimization volume mesh smoothing can be done in different ways. On one side, we can perform the Laplacian smoothing as in the Algorithm 2 and after that, apply

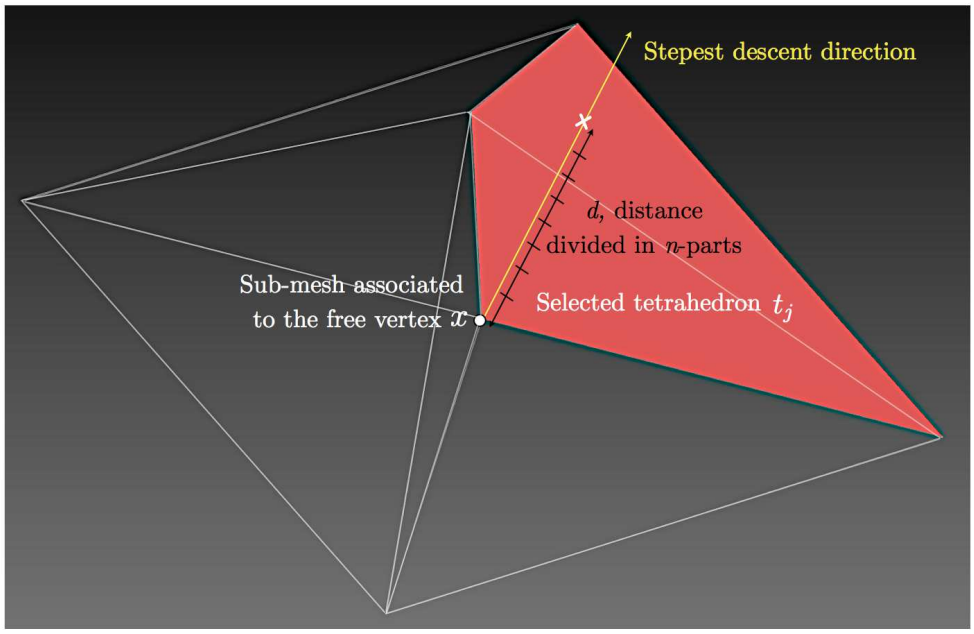


Figure 3.35: Steepest descent optimization process.

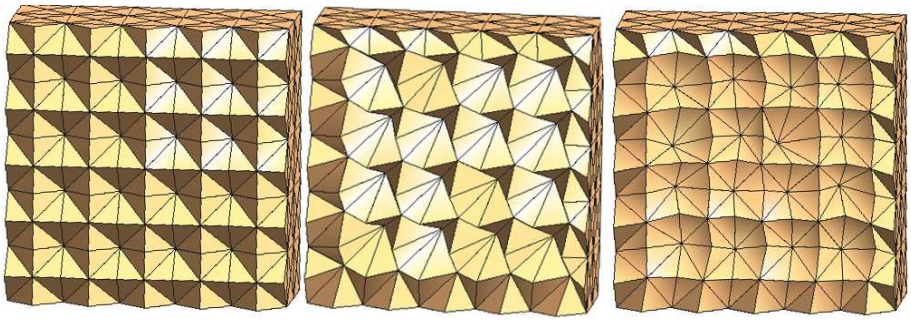


Figure 3.36: (Left) Regular mesh. (Middle) Perturbed mesh. (Right) Smoothed mesh.

an optimized volume smoothing to volume points from the interface to the fourth layer. This parameter can be modified adding or removing points to be smoothed in the volume mesh. This algorithm is described in Algorithm 6.

On the other side, we have applied another strategy to couple the subdomains so that the volume mesh smoothing is done inside the surface smoothing iterations. This is described in Algorithm 7.

Algorithm 6 Coupling surface-volume smoothing version 1

1: Call Algorithm 2 or 3+4
2: **(INPUT)**: n_{layer}
3: **for** Volume points untill the n_{layer} from the interface **do**
4: Tetrahedra $t_i, i = 1, \ldots, m$ connected with the free point.
5: Call Algorithm 5
6: **end for**

Algorithm 7 Coupling surface-volume smoothing version 2

1: Mark not to move points in the interface and located in the real boundary.
2: **(INPUT)**:

- $\lambda = 0.6307$, $\mu = 1.0/(0.1 - 1/\lambda)$
- *numit*: number of iterations

3: **(INITIALIZE)** $k = 0$, $x^k = x$.
4: **while** $k \leq numiter$ **do**
5: **for** Laplacian loop: Points to be moved with free coordinates (x) **do**
6: Compute the barycenter: b_i with the connected points to be moved.
7: Move the coordinate $(x^k) : x^k = x^k + \lambda(b_i - x^k)$
8: Tetrahedra $t_i, i = 1, \ldots, m$ connected with the free point
9: Call Algorithm 5
10: **end for**
11: **for** Anti-Laplacian loop: Points to be moved with free vertex coordinates (x^k) **do**
12: Compute the baricenter: b_i of the coordinates of the rest of the points connected with the point to be moved.
13: Move the coordinate $(x^k) : x^k = x^k + \mu(b_i - x^k)$
14: trahedra $t_i, i = 1, \ldots, m$ connected with the free point
15: Call algorithm 5
16: **end for**
17: **end while**

As we noted before, one of the worst scenarios in which the smoothing interface technique helps us to a better construction of the extension elements is represented in Figure 3.36. It corresponds to a regular mesh in a cube with a spherical patch inside. When the hole is constructed, the interface surface contains the boundaries forming ninety degrees between them which complicates the construction of the extension elements. In Figure 3.8 we have illustrated the interface of the hole, smoothed and non-smoothed. If we do not apply the smoothing of the interface we are in trouble when the extension elements are

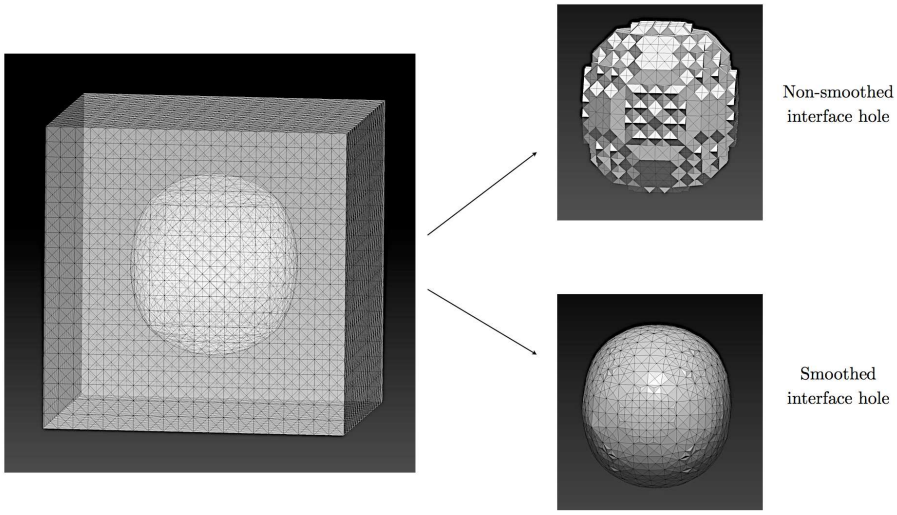


Figure 3.37: (Left) Geometry of the domain. (Right) Non-smoothed and smoothed interface.

constructed, bearing in mind that these new elements for this interface have to be constructed inwards to the surface.

3.9 HERMESH numerical properties

The numerical properties of HERMESH will be shown in the different applications which are valid for the method; Chimera-type problems and joining meshes. They will be presented in the next chapters, 4 and 5, respectively. In Table 3.9 we present which are these numerical properties, for which equations they have been tested and for which particular application.

	Navier-Stokes	Solid mechanics	ADR
Linear	-	-	Joining
Mesh cvg	Joining & Chimera	Chimera	-
Mass cons.	Chimera	-	-
Matching	Chimera	Joining	-

where the definitions of the different properties are the following:

- *Linear*: The HERMESH method gives error zero for linear solutions. Manufactured solution technique with imposed linear solution is applied to prove it.

- *Mesh convergence*: The order of convergence of the HERMESH method in different equations. Manufactured solution technique with different imposed solutions are applied in each case.
- *Mass conservation*: Quantify the mass non-conservation in Navier-Stokes equations.
- *Matching overlap*: To show that if the union of the subdomain meshes and their extensions matches with the one-domain mesh then we recover the same solution in both problems, two-subdomains and one-domain.

Chapter 4

Hermesh method for Chimera strategy

As we have explained in first chapter, the main idea of the Chimera method is to generate independent meshes for the components present in the computational domain (e.g. fuselage, airfoil, flap, slap, etc.) and to couple them via a coupling strategy in order to obtain a global solution. The appealing characteristics of the Chimera method have permitted many applications such as simplified mesh generation, local refinement, moving components or optimization problems saving a lot of computational resources in all cases. HERMESH also offers the possibility of applying this technique, which will be given in detail below.

4.1 Introduction

The Chimera jargon. When the Chimera Method is taken into account, the mesh is divided into a *background mesh*, which covers all of the computational domain, and *patch or overset meshes* attached to the different components (objects) which are located on the background mesh. First, a proper preprocess is applied to create the interface of the background mesh with the overset meshes. This can be achieved by removing elements (in Finite Element jargon) of the background mesh located inside the patch meshes to create apparent interfaces between the background and the patches: this is the *hole-cutting*. If the patch meshes are stationary, the elements of the hole can be permanently removed from the mesh definition. Then a domain decomposition (DD) *coupling* algorithm is carried out in order to obtain a "continuous solution" across the interfaces.

Different implementations. In the literature, all the Chimera methods have a common feature, the hole-cutting. They differ in the way the coupling of the background with the patches is carried out. Figure 4.1 illustrates three options.

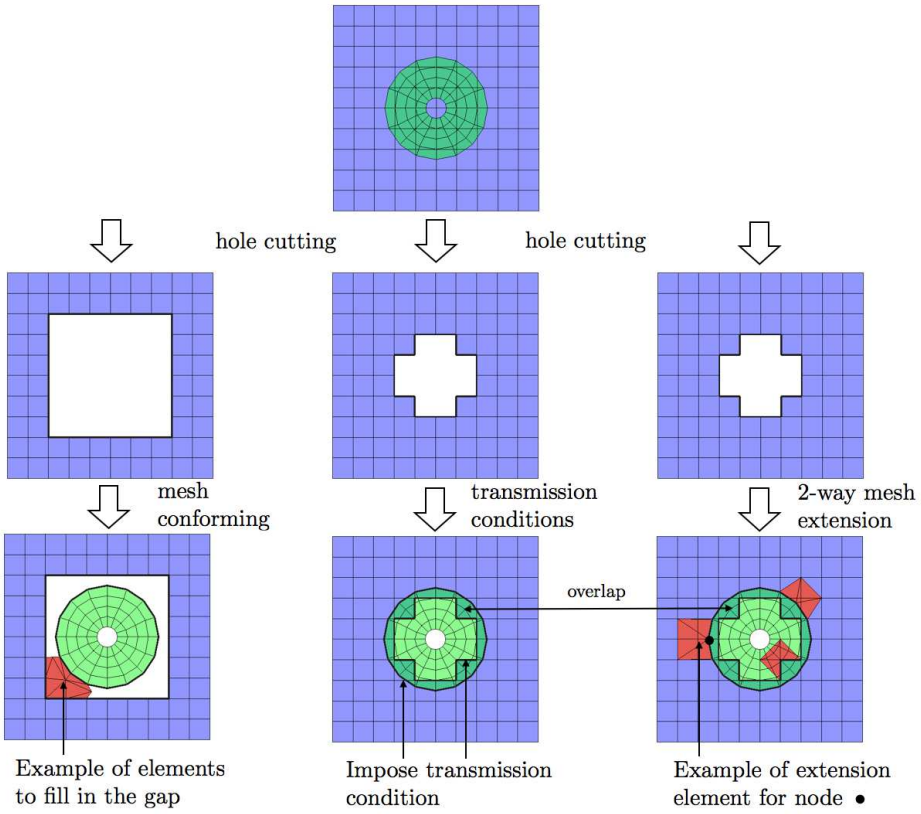


Figure 4.1: Some Chimera methods.

In the first method shown in Figure 4.1 (Left), the hole-cutting is created in such a way that a gap between the two subdomains is obtained. Then, the gap is meshed in such a way that a conforming mesh connects the subdomains (see e.g. Chan & Buning (1996)). This technique is commonly used as a meshing strategy to glue octree meshes to boundary layer meshes (see e.g. Park, Jeong, Lee, & Shin (2013)). The two main advantages of this method are as follows: on the one hand, it consists of a preprocess technique that can be taken out of the simulation code; on the other hand, the resulting mesh is conforming. The main drawback is that some nodes may be added if the mesh sizes are different, and the meshing can be problematic. The Shear-Slip Mesh Update Method (SSMUM), see Behr & Tezduyar (1999), is a particular

case of this method where the nodes on the interfaces are reconnected to the other interface nodes without adding any nodes in the gap, which restricts the application of the method to meshes of similar sizes on both sides of the gap. The Chimera method shown in Figure 4.1 (Middle) consists in imposing transmission conditions to obtain continuous variables and continuous fluxes across the interfaces. In a finite element context, the transmission condition on one interface could be of Dirichlet type and on the other one of Neumann type, in order to impose continuity of the solution and its flux, respectively. Usually, the coupling is imposed iteratively, imposing an additional iteration loop to the original algorithm. In addition, the flux continuity is PDE-dependent and a specific implementation would be required for different physics. The latter method, illustrated by Figure 4.1, (Right) is the one proposed in the present work.

The proposed parallel implementation of the Chimera method is only valid for fixed components. In fact, the coupling between the meshes is done as a preprocess, before the mesh partitioning is carried out for parallelization purpose. This is not an inherent restriction of the method but a deliberate choice on the authors' part, due to the specific requirements of the applications envisaged in this work. It should be mentioned that the parallelization of the Chimera method in a distributed memory context is not an easy task whenever the method is meant to be implicit. In common explicit implementations of the method, the transmission conditions are usually imposed iteratively, as we have noted before. This option means that the subdomains can be solved in a staggered manner. In this case the parallelization is more straightforward as one could use as many parallel instances of the code as subdomains and the only difficulty consists in the exchanging of the transmission conditions through MPI. In the case of an explicit coupling, each subdomain is independently solved with different parallel instances of the code and the coupling is carried out between these instances in an iterative way. As we have seen in previous chapter, in the implicit case, the transmission conditions are included in the matrix of the algebraic system. So in this case, the global coupled solutions are solved in parallel, regardless of the existence of background and patch meshes.

If the subdomains are moving, the number of degrees of freedom as well as the connectivity in each subdomain vary in time, and the parallel implementation is cumbersome. The method proposed, implemented as a preprocess, can therefore be applied for simplifying the mesh generation, for local refinement and optimization.

The following section deals with the hole-cutting, which leads to the creation of the background interface.

4.2 Hole cutting

For the sake of clarity we will consider only one patch mesh, although the discussion still holds for multi-components meshes. The hole-cutting task consists in removing some elements, the hole elements, from the background mesh. The resulting background mesh is therefore made of the original elements without the hole elements and the interface is the boundary mesh formed by the hole. This interface will be used later on to couple the background with the patch. One key point when creating the hole is that the boundary of the new mesh is a manifold boundary mesh.

> *A triangle mesh is a 2-manifold if it contains neither non-manifold edges nor non-manifold vertices, nor self-intersections. A non manifold edge has more than two incident triangles and a non manifold vertex is generated by pitching two surface sheets together at that vertex such that the vertex is incident to more than one fan of triangles.*

More details about the requirements for ensuring mesh validity can be found in Dey, Shephard, & Georges (1997). The purpose is to avoid the situation depicted in Figure 4.2 (Right), where a node (vertex) is connected to four boundary edges and its connected elements (faces in 2D) do not form neither a closed nor an open fan. To ensure that we obtain a manifold mesh, the idea is

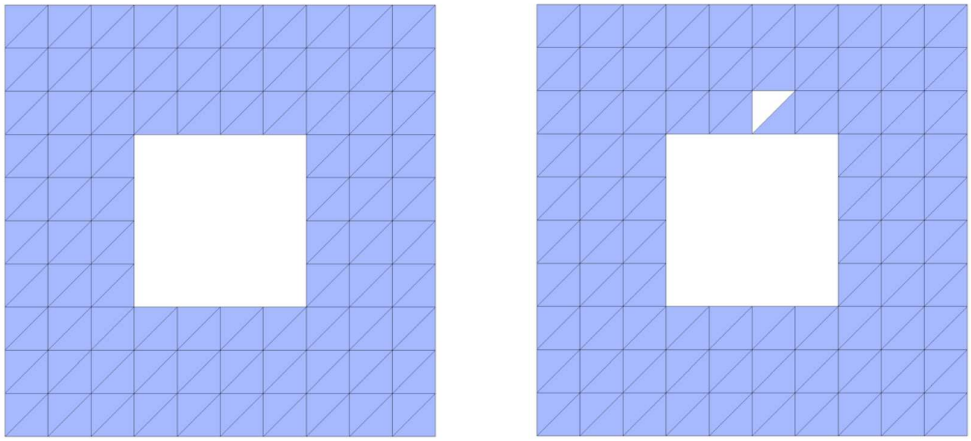


Figure 4.2: Manifold and non-manifold mesh after hole cutting.

to recursively mark some candidate hole elements, using the common faces as a criterion. Therefore, we will never mark a candidate element if it connects to

another marked element through only an edge or a node. The method consists therefore of two steps:

1. Select the candidate hole elements.
2. Mark the candidate hole elements recursively using the common face criterion.

Candidate hole elements. We propose two methods for selecting the candidate elements, as illustrated in Figure 4.3. The first one consists in first marking the elements crossed by the patch interface. The candidate elements are therefore all the elements until the previously marked elements are reached by the recursive algorithm. The second method consists in using the signed distance between the background nodes and the patch interface to identify the hole nodes located inside the patch. In this case, the candidate elements are those elements all of whose nodes are hole nodes. To evaluate the signed distance, we have used a *skd-tree* strategy, as explained in 2.4. In this case, the surface is formed by the patch outer boundary elements.

Hole elements from candidates. Once candidate elements have been selected, a recursive algorithm is executed to create a manifold hole boundary. The seed element of the recursive algorithm is selected near the center of gravity of the hole. Figure 4.4 represents the recursive algorithm where elements are chosen from top to bottom and black to white. The red element is the seed.

The recursive algorithm is summarized in Algorithm 8. We will not give too many details here as the complete algorithm largely depends on the data structure in question. We shall simply mention that in the current implementation, when marking an element, the associated faces and nodes can also be marked, in order to identify critical edges and vertices more easily.

The algorithm works as follows. From a marked element `ielem` in the stack, we loop over its neighbors `jelem,` which share common faces. In order to accept and mark the new element `jelem`, one should check that it only connects to other marked elements `kelem` through faces. This is necessary in order to avoid having critical edges or vertices, as illustrated in Figure 4.5, and thus to obtain a manifold mesh.

4.2.1 Hole cutting plus HERMESH coupling

We now illustrate the complete strategy for setting up the Chimera method using two one-dimensional and overlapping meshes, as shown in Figure 4.6.

The complete strategy consists of the following steps:

- *Step 1: Hole cutting.* From the top meshes, perform the hole cutting by identifying the hole elements. In Figure, they are elements 4, 5 and 6. We

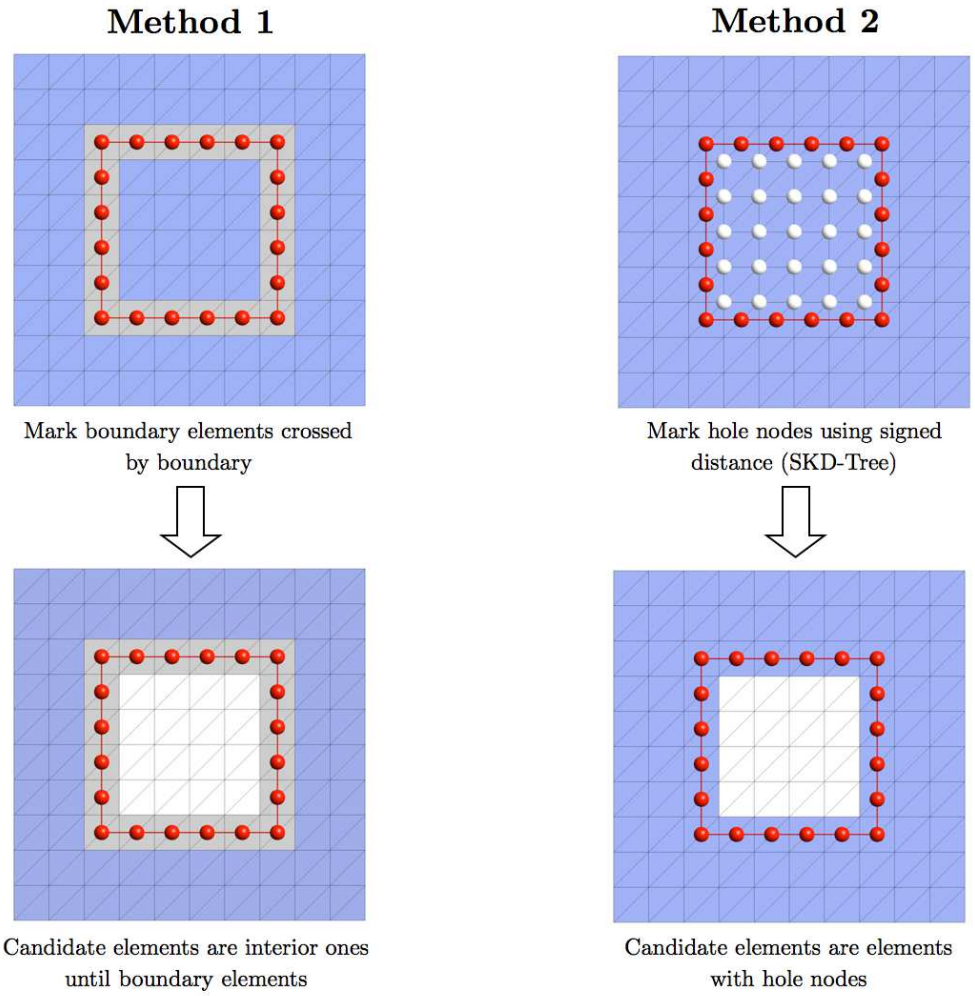


Figure 4.3: Two methodologies to select hole element candidates. (Left) Marking the elements crossed by patch interface. (Right) Marking the hole nodes using the signed distance to the patch interface.

observe that we end up with an overlap between the subdomains. The overlap is the zone comprised between nodes 4 and 10 on the left-hand side, and between nodes 7 and 14 on the right-hand side.

- *Step 2: Identify fringe nodes.* The fringe nodes are the nodes located on the boundary of the hole of the background (blue subdomain), which are nodes 4 and 7, and the outer boundary of the patch (green subdomain), which are nodes 10 and 14.
- *Step 3: HERMESH coupling.* Create the extension elements (in red),

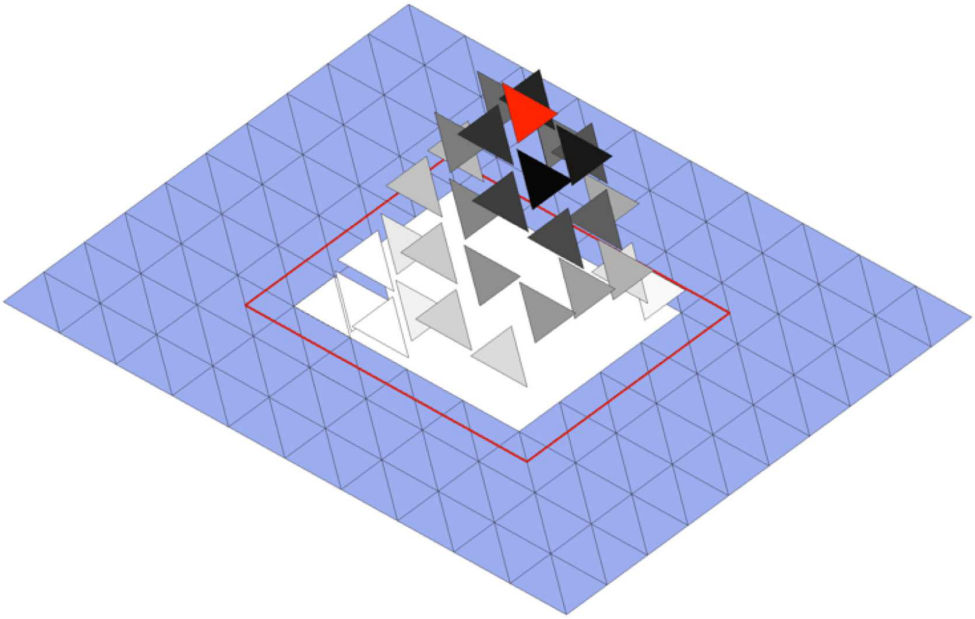


Figure 4.4: Recursive algorithm to create the hole from the candidate elements. The red element is the seed of the algorithm.

Algorithm 8 Recursive algorithm to create hole from candidate hole elements.

```
Find seed hole element ielem
Initialize element marker lmark(:) = 0
lstack(1) = ielem
lmark(ielem) = 1
istack = 1
while istack ≠ nstack do
  istack = istack + 1
  ielem = lstack(istack)
  for Candidate neighbors jelem of ielem with common face do
    if Critical edge or critical vertex then
      Do not mark jelem
    else
      Mark element: lmark(jelem) = 1
      nstack = nstack + 1
      lstack(nstack) = jelem
    end if
  end for
end while
```

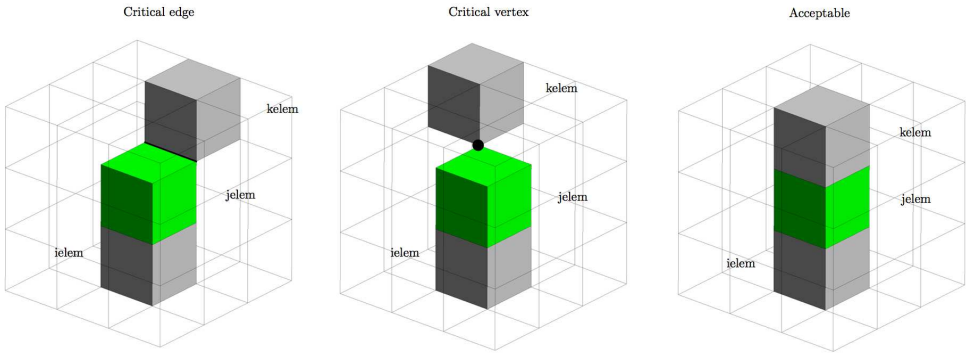


Figure 4.5: Marking or not a neighbor `jelem` (green) of `ielem` (grey) according to the previously marked elements `kelem` (grey).

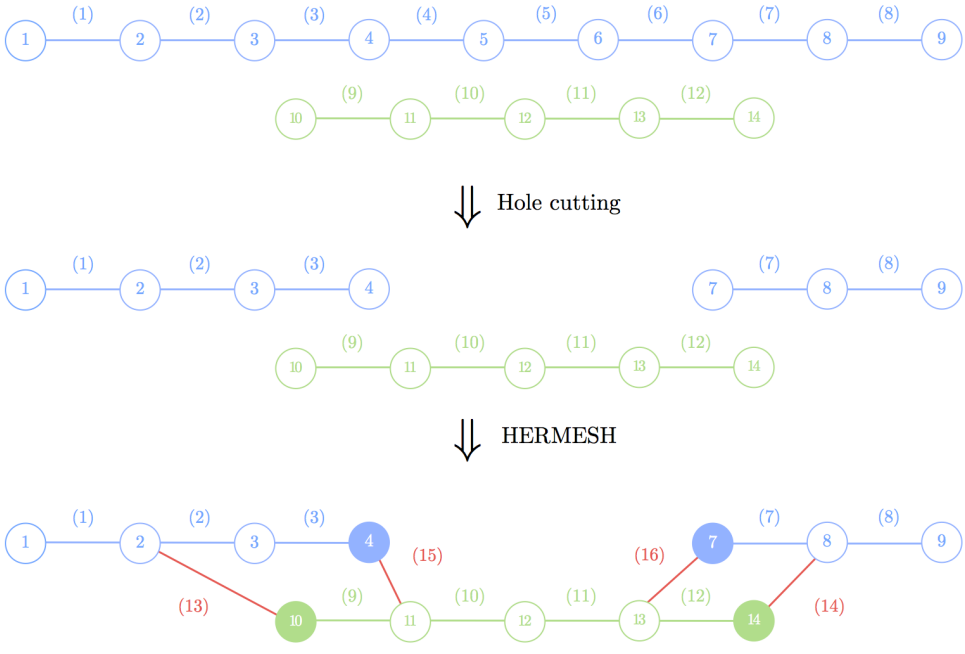


Figure 4.6: Chimera method: hole cutting plus HERMESH coupling.

which are elements 13, 14, 15 and 16, which connect one subdomain to the other. The final overlap is now the zone comprised between nodes 2 and 11 on one side and nodes 7 and 13 on the other side.

4.3 Implementation aspects

We are going to describe some details of the particular case of the Chimera problem in relation to the implementation aspects, although some of them have already been partially explained in the general Chapter 3.

4.3.1 Equation assembly

Once the extensions have been created, a very simple implementation is possible. We have to deal with the new types of elements, namely the hole elements and the extension elements.

Hole elements. The governing equations should not be assembled in the hole elements as they are no longer part of the solution process. We can choose to eliminate the hole elements and hole nodes from the mesh definition and apply a node and element renumberings before going any further. However, if the Chimera method is implemented to treat moving components, this option is no longer valid. In the present work, we do not eliminate the hole elements and hole nodes to show how the implementation should proceed.

Extension elements. As described in the previous chapter, the extension elements are attached to a given fringe node and should only be used to assemble the fringe node equation. Therefore, something must be done during the assembly process in order not to assemble the complete contribution of the extension elements. The following strategy treats both the extension and hole elements and is specific to the finite element method.

Let us once again consider the array `letyp(nelem)` where `nelem` is the total number of elements (including extensions and holes). Say this array is:

$$\begin{cases} \text{Normal element:} & \texttt{letyp(ielem)} = 0, \\ \text{Extension element:} & \texttt{letyp(ielem)} = 1, \\ \text{Hole element:} & \texttt{letyp(ielem)} = -1. \end{cases}$$

4.3.2 Hole nodes treatment

We have already mentioned that hole elements could be removed from the mesh. However, they are kept in the present implementation. Therefore, as these elements do not participate in the assembly of the global matrix, we are left with empty rows in the global matrices and RHS's. These rows correspond to the hole node degrees of freedom. We have basically two alternatives. On the one hand, the hole nodes can be eliminated from the matrix graph (in the CSR format). On the other hand, we have the option to let the matrix as it is; in fact, no free node is connected to a hole node, so any nodal value on a hole

node is irrelevant. If this last option is selected, one should only remember to put a non-zero value on the diagonal if a diagonal preconditioner is used in the algebraic solver. The drawback is that useless operations are carried out in the matrix-vector operations of the iterative solvers.

4.4 Numerical propierties of the HERMESH Method for Chimera problems

As is illustrated in Table 3.9, , Chimera problems have been tested with HERMESH for different PDE's and certain numerical properties have been proved. They will be presented in this section. In particular, we are going to show the matching property, i.e., if the independent coupled meshes plus its extension elements replicate the same mesh as in one domain, the solution is exactly the same. This is done for Navier-Stokes and Solid Mechanics in two-dimensional problems. Then a three-dimensional problem where the advection-diffusion-reaction equation is solved will be presented and we will show that the solution is not affected by the direction of the flow. A mesh convergence test proves that the method is of order two for Navier-Stokes and Solid Mechanics problems. Finally, we shall quantify the loss of mass for the Navier-Stokes equations in the last example.

4.4.1 Independence of flow direction in the ADR problem

As we have mentioned in the first chapter, classical domain decomposition techniques are flow-direction-dependent when they solve an advection-diffusion-reaction equation. Here we present an example in three dimensions for a Chimera-type domain solved with the HERMESH method. We will show that the solution is not affected by the direction of the flow across the interface. Figure 4.7 shows the problem to have been solved with the boundary conditions for unknown $\mathbf{u}$, and in Figure 4.8 we can see the meshes used in this example. The gray extensions are the extensions of the patch and the green ones are those of the background. On the right and bottom part of the figure, a zoom of the corner is shown, as are the extension elements of a particular fringe node of the patch.

4.4.2 Mesh convergence

This section is devoted to solving the problem with the manufactured technique, explained in Section 2.6. With these examples, we want to study the mesh convergence and compare the results with the one-domain solution for Navier-Stokes equations and solid problems.

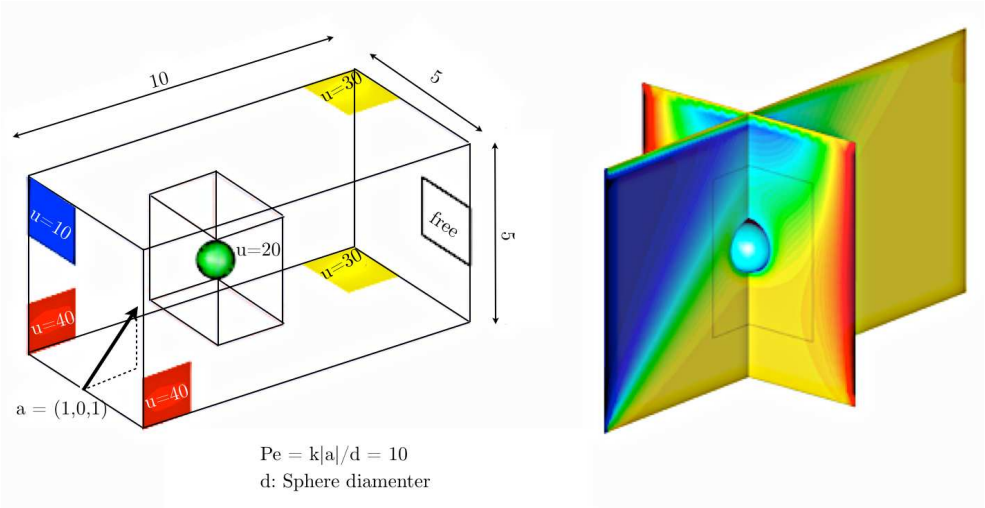


Figure 4.7: Advection-Diffussion-Reaction problem. Presentation of the physical problem and the solution not affected by the flow direction.

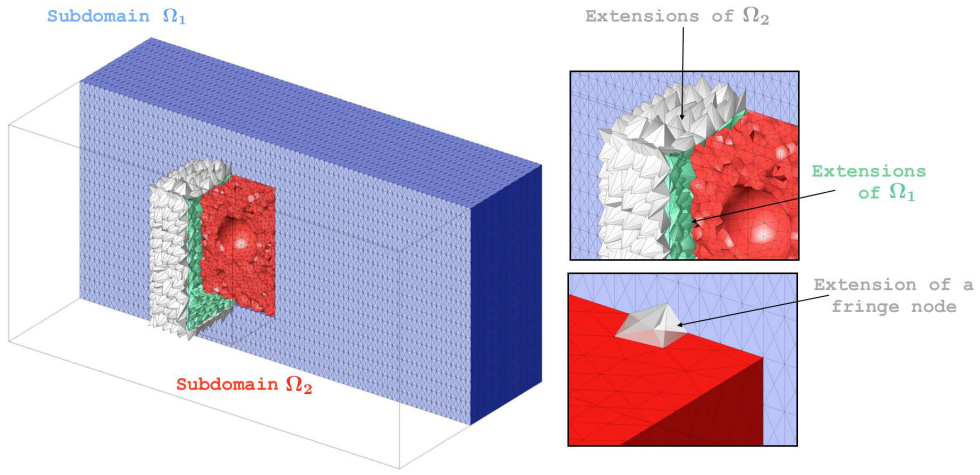


Figure 4.8: Advection-Diffussion-Reaction problem. Mesh and extension elements

Chimera in solid mechanics for a plate

The two-dimensional exact solution $\boldsymbol{u}^{(e)}$ considered here for the manufactured technique described in 2.6 is:

$$u_1^{(e)} = x^3 \times y^4 \tag{4.1}$$
$$u_2^{(e)} = x^3 \times y^3 \tag{4.2}$$

solved on a unit square. Figure 4.9 shows the resulting mesh convergence results, namely the following definition of the errors:

- $\epsilon(\boldsymbol{u})$: error of the displacement;
- $\epsilon(\boldsymbol{\nabla} u)$: error of the gradient of the displacement.

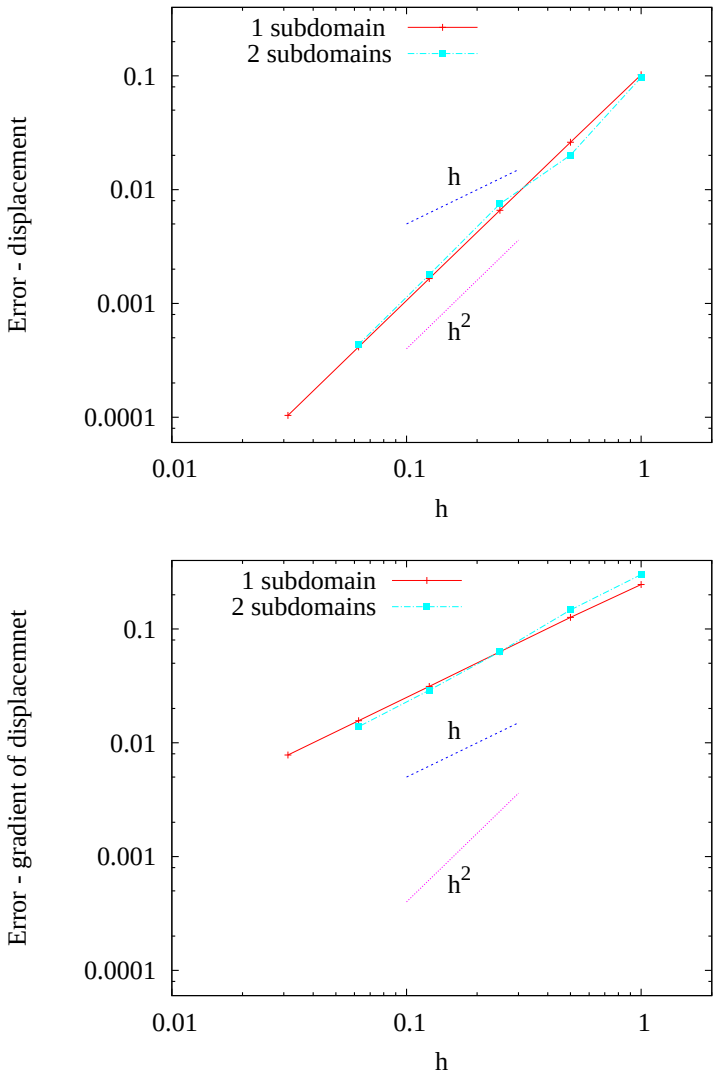


Figure 4.9: Mesh Convergence study for solid mechanics; $\epsilon(\boldsymbol{u})$-error (left) and $\epsilon(\boldsymbol{\nabla} u)$-error (right).

Figure 4.10 shows the three finest meshes of the convergence study. The subdomains are meshed with $Q1$ (quadrilateral) elements while the extensions are $P1$ (triangular) elements. For the one domain solution, $Q1$ regular meshes have been used.

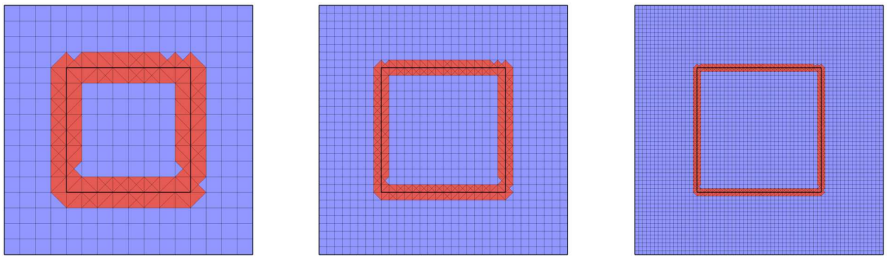


Figure 4.10: Mesh Convergence study for solid mechanics; the three finest meshes used for the Chimera method (blue and red elements are, respectively, the regular and extension elements).

The results successfully confirm a first order convergence in $\epsilon(\boldsymbol{\nabla} u)$ and second order in $\epsilon(\boldsymbol{u})$ with respect to h for the Chimera method with almost the same error as for one subdomain.

Chimera in CFD for two-dimensional example

To assess the mesh convergence, we shall consider the following exact solution to have been solved with the manufactured technique strategy described in 2.6 on the unit square:

$$\begin{aligned} \boldsymbol{u}_e &= [1 + x + y^2, -1 - y - x^2]^t, \\ p_e &= x^2 + y^2. \end{aligned}$$

Both the one-domain and Chimera method with two subdomains are solved, and two different scenarios are studied. In the present case, the two subdomains used for the Chimera method have the same mesh size and their interfaces end up being matching. The overlap comes from the extension elements, after connecting the subdomains. On the one hand, the problem is solved with a Dirichlet condition for the velocity on all the boundary. On the other hand, the right-hand side wall Dirichlet condition is substituted by a Neumann condition. Figure 4.11 shows the meshes and boundary conditions. As shown in chapter 3, the HERMESH coupling is equivalent to extending the subdomain and imposing a Dirichlet condition on the extension nodes. Therefore, we may not conserve mass at the subdomain interfaces. On the one hand, if the computational domain is confined, it is well known that the continuous problem

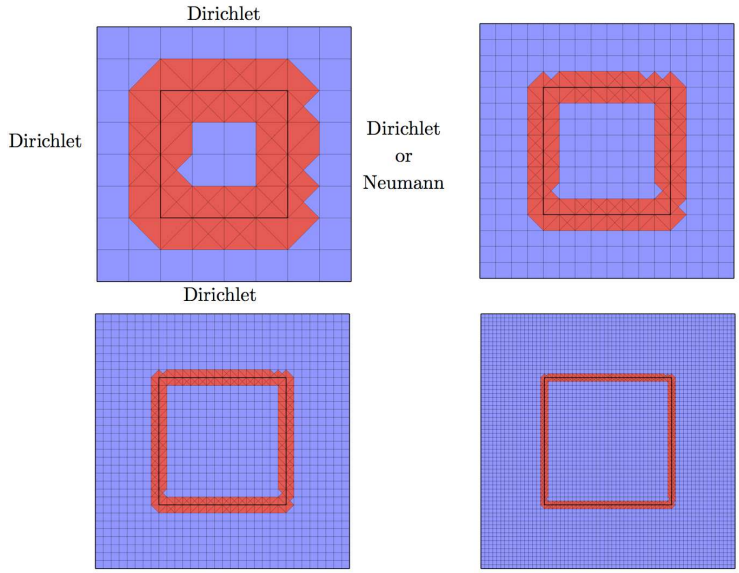


Figure 4.11: Mesh convergence study for Navier-Stokes equations. Meshes and boundary conditions.

does not have a solution (as the Dirichlet condition violates the data compatibility which establishes the global mass conservation). On the other hand, the numerical problem has a solution, but the pressure solution carries an error according to the magnitude of the mass imbalance. The problem can be avoided by imposing a Neumann condition on one of the walls. Figure 4.12 shows the errors obtained in velocity and pressure using the one-domain and the Chimera method, and for the Dirichlet and Neumann conditions. The expected rates of convergence are obtained, with low differences in velocity between the Chimera and one-domain solutions. The difference is much greater in pressure in the case of the Dirichlet condition on all the boundary. This difference is mitigated by imposing the Neuman condition.

4.4.3 Mass conservation

A good review of this issue can be found in reference Liu & Shyy (1996). This matter is also dealt with in detail in Ahusborde & Glockner (2010). In order to quantify the mass non-conservation, the Navier-Stokes equations are solved on the same geometry as in the mesh convergence example of the previous subsection, imposing a parabolic profile on the left wall (inflow), zero velocity on the top and bottom walls, and a zero traction condition on the right wall (outflow). Due to the not-strictly-conservative property of the HERMESH coupling, we expect the background to interpolate a global non-zero mass from the patch.

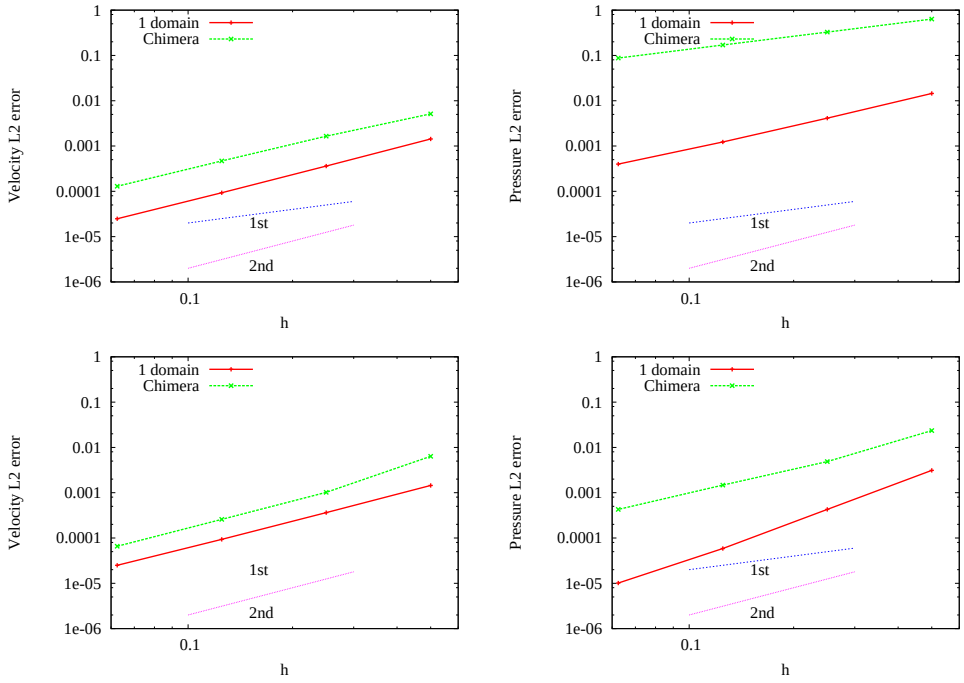


Figure 4.12: Mesh convergence study for Navier-Stokes equations. (Top) All Dirichlet conditions. (Bot.) Right wall is a Neumann condition. (Left) Velocity L2 residual. (Right) Pressure L2 residual.

This is shown in Figure 4.13.

The mesh convergence of the velocity is quadratic so it is expected that the mass imbalance is expected to be quadratic as well. This comment is confirmed by Figure 4.14, where we depict the difference in mass between inflow and outflow as well as the percentage of error.

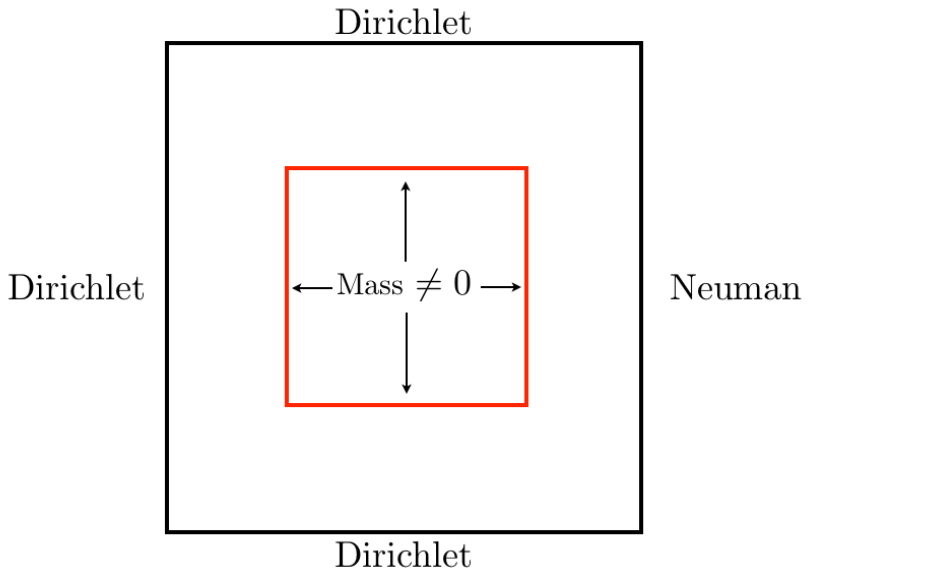


Figure 4.13: Non-conservative interface between background and patch meshes.

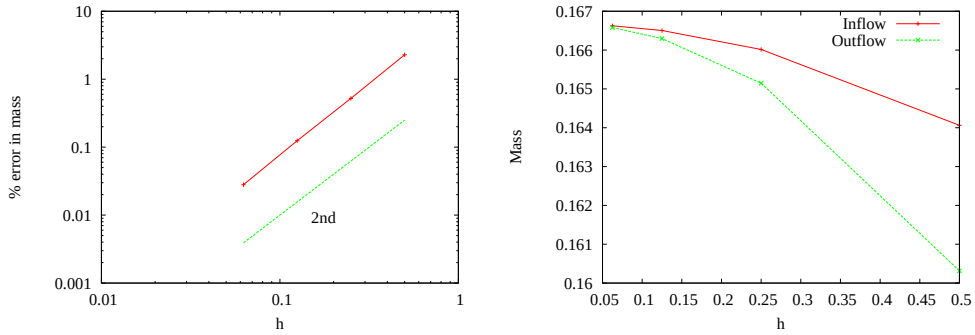


Figure 4.14: Mass conservation.

Chapter 5

Hermesh Method for mesh gluing strategy

As we have discussed in the first chapter, the question of the non-conforming meshes is very useful in many circumstances. One of the strategies in complex geometries is that of dividing the geometry into meshable parts so as to simplify the meshing process. Also, in fluid-structure interaction, we can find problems in which the geometry does not match perfectly and gaps may appear between different parts of the domains. In all such cases, it is very important to be able to join these non-matching meshes with some flexibility in the relative position of the different parts composing the whole geometry or the relative mesh size, and in the most automatic way as possible. We have developed the HERMESH method to be able to deal with non-conforming meshes. In this chapter, we will describe the capacities HERMESH has for automatically joining meshes as a *lego*, as well as show the numerical properties with different examples.
To describe the gluing method developed in this work, we shall consider four different situations illustrated in Figure 5.1 to which the the HERMESH method can be applied. These situations correspond to the main combinations that appear in practice. When meshes come from different sources, their interfaces are likely not to match perfectly. Therefore, some parts can present a gap (Figure 5.1 (a,b,c)) and others can be overlapping (Figure 5.1 (d)). In addition, we need to define where the subdomains should connect to each other. In other words, we need to define the interfaces where HERMESH coupling is to be applied. If the user knows the interface a priori, this interface should be depicted; such is the case of Figure 5.1 (a) as well as partially the case of Figure 5.1 (b). If the user has no means to act on the mesh and describe these interfaces, one would like them to be automatically detected, whenever possible, as shown in Figure 5.1 (c,d) and partly in (b). In the following, we shall confine our discussion

to two subdomains, although the methodology still holds for multi-component meshes.

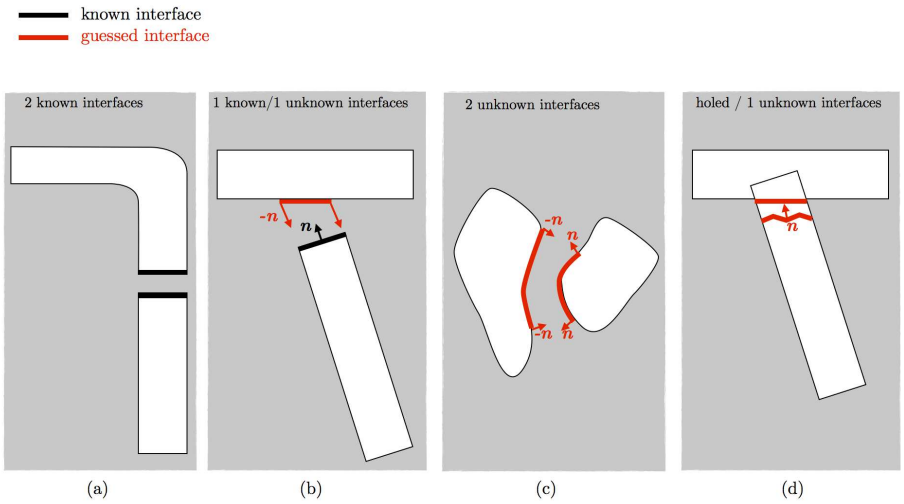


Figure 5.1: Lego with independent components.

The aforementioned possibilities come from the combinations of three different types of interfaces. We refer to these interfaces as: *Known*, *Unknown* and *Holed-Unknown*.In accordance with this nomenclature we have the following four possibilities illustrated in Figure 5.1:

	Interface i	Interface j
Case a	*Known*	*Known*
Case b	*Known*	*Unknown*
Case c	*Unknown*	*Known*
Case c	*Unknown*	*Holed-Unknown*

At the mesh level, an interface consists of a list of element boundaries which, in turn, defines a list of nodes on which the HERMESH method will be applied. These nodes will be referred to as fringe nodes in the next Section.

5.1 Description of the types of Interfaces

Let us now describe the different types of interfaces.

- The *Known* interface of a subdomain i is the one that is prescribed by the user as an input data.

- The *Unknown* interface of a subdomain i is constructed by using some geometrical information of the adjacent subdomain j in different ways, according to whether subdomain j has a known or unknown interface with i.

- The *Holed-Unknown* interface of a subdomain i is constructed with a previous hole-cutting step.

According with this definitions we have the four cases represented in Table 5 and explained next:

- Case *a*. Both interfaces are prescribed by the user and the extension elements can be constructed directly.

- Case *b*. If the interface of j is known (Figure 5.1 (b)), the geometrical information is the average normal $\boldsymbol{n}$ of the interface of j. Then, we guess the interface as being the list of boundaries of i that are intercepted by the projection of the interface of j along the direction of $\boldsymbol{n}$.

- Case *c*. when both interfaces are unknown, the procedure is quite risky and its usefulness largely depends on the case considered. Figure 5.1 (c) shows an example of both unknown interfaces. Let us first select a Master subdomain, for instance i; the procedure is illustrated in Figure 5.2. Boundaries b_1 and b_3 are not interface boundaries, whereas b_2 is. The procedure consists of the following:

  - Loop over the boundaries b of i, and compute the line going from the center of gravity of b along the normal of b; if the line does not intersect with j, do not consider the boundary (boundary b_3 in the figure). If this line intersects subdomain j (triangles in the figure), check if it intersects with its own boundary:
    - If the line intersects with its own subdomain boundary (in i), do not consider b. In the figure, this is the case of boundary b_1.
    - If the line does not intersect with its own subdomain boundary, then mark this boundary b as an interface boundary. In this Figure, this is the case of b_2.
  - Once the process is finished, mark this interface (union of all the b's) as known.
  - Use the procedure described before for a known-unknown interface corresponding to Figure 5.1 (b), where now j is the unknown subdomain.

  Roughly speaking, the interface of an unknown interface is created by the shadow of the neighbor mesh.

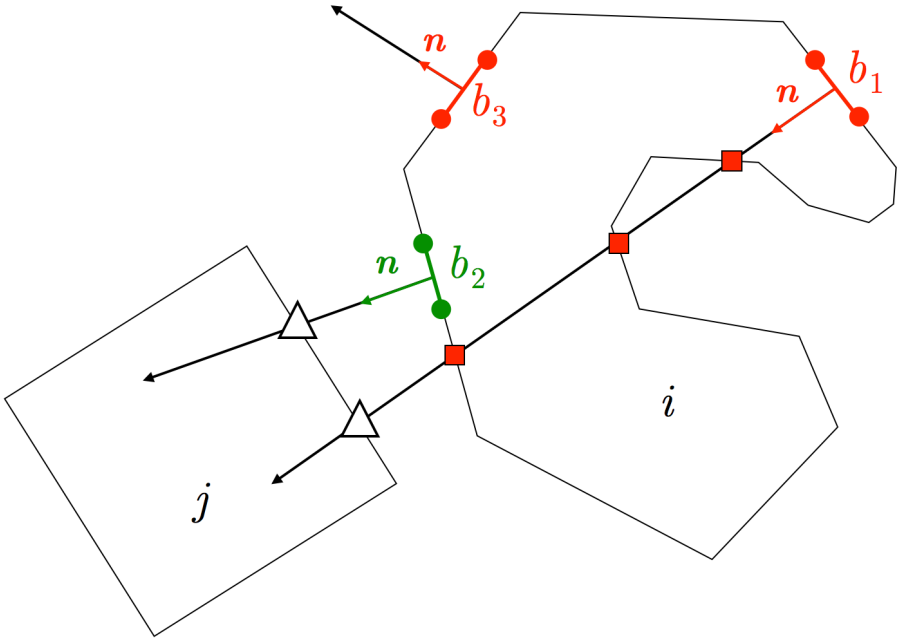


Figure 5.2: Unknown-unknown interfaces procedure.

- Case *d*. The *Holed-Unknown* case (Figure 5.1 (d)) means that a previous step is necessary for defining the interface, that is, the hole-cutting process. The hole-cutting task consists in removing some elements from the mesh, referred to as the hole elements, and defining the *holed-unknown* interface. The interface will be the boundary mesh formed by the hole. This process is the same as the hole-cutting process of the Chimera problem explained in Section 4.2. However, in this case we do not locate the seed of the recursive algorithm in the hole candidate elements because we could find a *holed-unknown* interface with two non-connected parts. An example is shown in the next chapter, corresponding to the simulation of a by-pass in a stenosed artery problem solved with HERMESH.

Once this is done, we identify the fringe nodes as the nodes that form the interface to be connected to the adjacent mesh. With all the different interfaces previously defined, we are able to couple disjoint and non-overlapping subdomains (case a, b, c) or overlapping subdomains (case d).

Finally, when the interfaces are defined, the next step consists in coupling the meshes applying the HERMESH technique, explained in chapter 3.

5.2 Numerical properties of the HERMESH Method for mesh-gluing problems

In this section, as in the Chimera chapter, we present the main numerical properties of the HERMESH method in different examples. We show that in problems of this kind we also obtain an error zero if the exact solution is linear for the manufactured strategy, as see Section 2.6. Also, we are going to prove that the solution is nodally exact when the composed mesh with HERMESH and the mesh in one-domain problem are the same. This test will be performed for the solid mechanics equations. Finally, we show a mesh convergence test for the Navier-Stokes equations.

5.2.1 Linear solution in gluing meshes

We consider the four geometries depicted in Figure 5.1. The first manufactured solution is linear in space. The automatic extension elements constructed by this *lego-tool* and the solution obtained are illustrated in Figure 5.3.

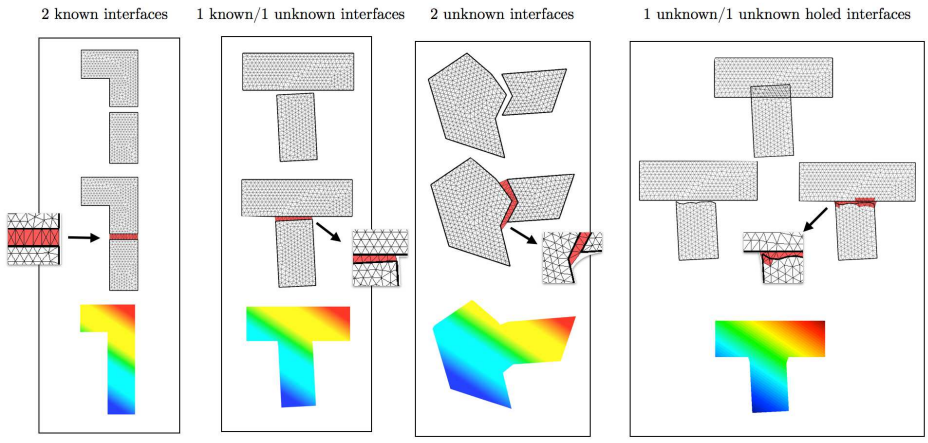


Figure 5.3: Results of the coupled meshes.

The results show that the error is null in the four cases, so, as we mentioned in the previous chapter, the error is null when the solution belongs to the finite element space.

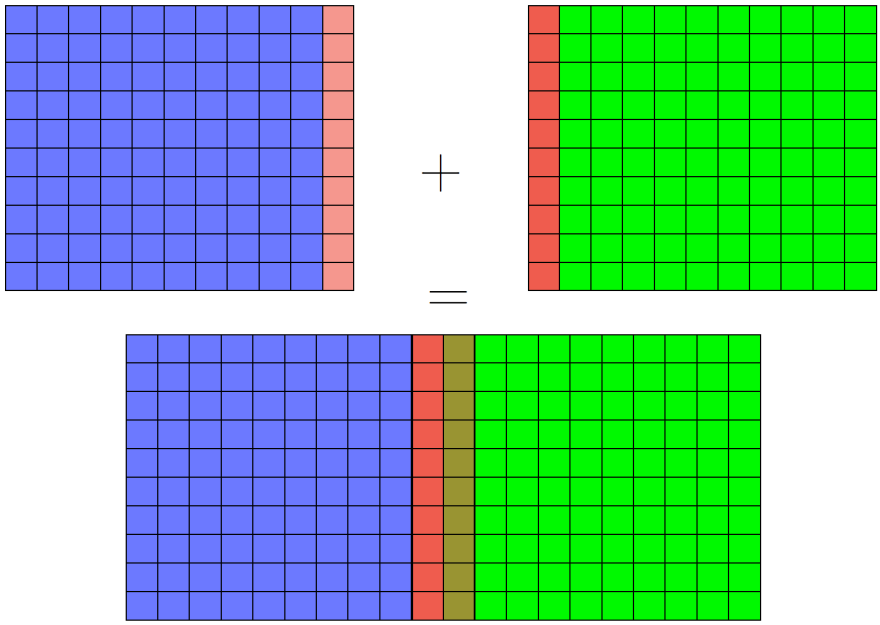


Figure 5.4: Matching overlap. Extension elements (in red) coincide with the adjacent elements.

5.2.2 Matching overlap

With this example, we want to show that when the subdomain elements (including extension elements) coincide everywhere including the overlapping zone, the solution using the HERMESH coupling is the same as with one subdomain. In this case, a two-dimensional end-loaded cantilever beam clamped on the other side is considered. We choose two disjoint subdomains and then construct the extension elements so that they coincide with the elements of the adjacent subdomain, as shown in Figure 5.4.

As we only want to show that the solution on one and two subdomains are the same, we do not provide any physical and numerical descriptions of the simulation. Figure 5.5 compares the solution, namely the vertical displacement and vertical normal stress, along a horizontal cut and at a given time step. Figure 5.6 compares the vertical displacement at some given point versus time. The two figures confirm that the solutions are identical.

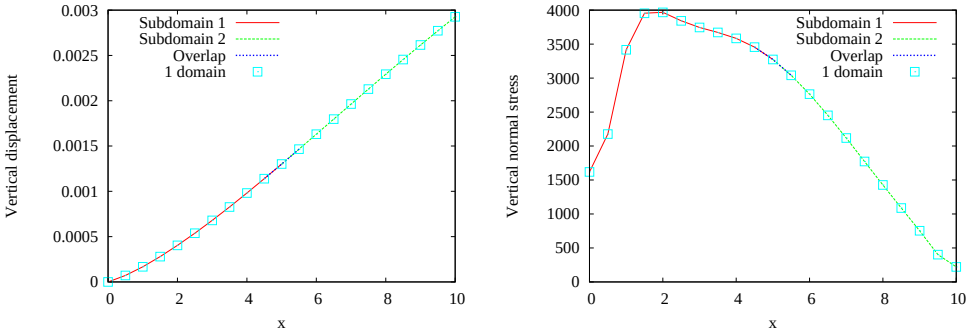


Figure 5.5: Matching overlap. Comparison of solutions with 1 and 2 subdomains. (Left) Vertical displacement along the beam. (Right) Vertical normal stress.

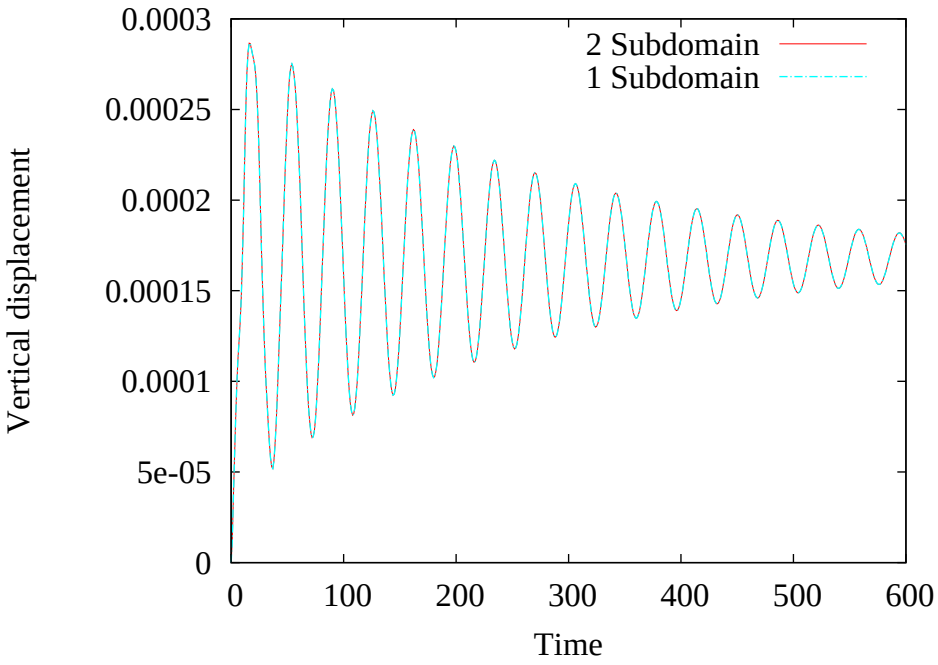


Figure 5.6: Matching overlap. Comparison of Vertical displacement vs time with 1 and 2 subdomains, at an arbitrary position.

5.2.3 Mesh convergence in gluing meshes for Navier-Stokes problem

With these examples, we want to study the mesh convergence and compare the results with the one-domain solution for Navier-Stokes equations and solid

problems.

We consider a unit square domain on which we solve the Navier-Stokes equations with a quadratic manufactured solution, as see in Section 2.6 with the following manufactured solution: $u_e = 1 + x + y^2$, $v_e = -(1 + y + x^2)$ and $p_e = x^2$, where (u_e, v_e) are the manufactured velocity components and p_e is the manufactured pressure. We compare the solutions obtained on one domain and using the HERMESH method for three different mesh sizes. The meshes and extension elements are shown in Figure 5.7. Note that the gap between the subdomain is of the same size as the elements of the finest mesh.

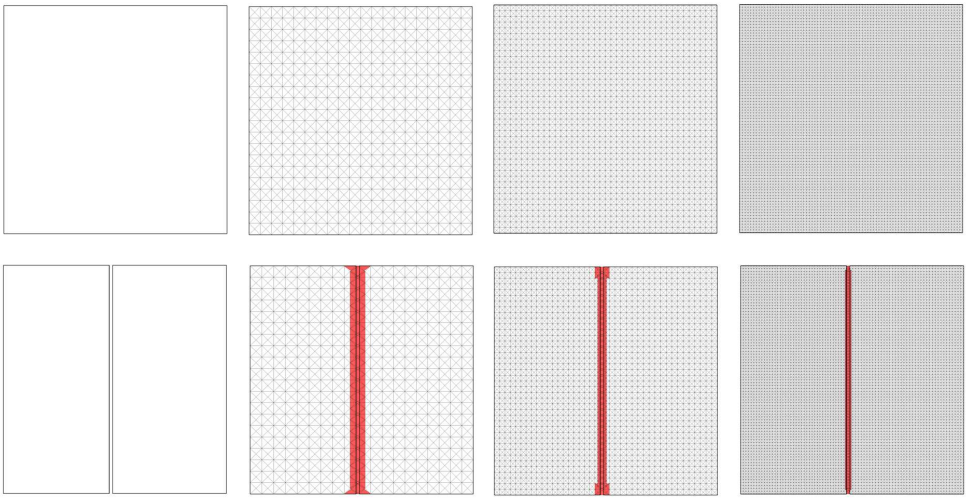


Figure 5.7: (Top) Meshes and geometries. (Top) One domain. (Bot.) HERMESH method.

The L^2 errors (represented for Equation 2.88) of the velocity and pressure are shown in Figure 5.8, and confirm that we obtain a quadratic convergence for the velocity when using the HERMESH method. Let us remark that the velocity error is of the same order for both approaches, whereas in the case of the pressure, it is one order of magnitude higher for the HERMESH method. As noted in Section 4.4.2, this bears on the lack of conservation of mass which introduces additional errors in the pressure. However, this error is consistent with the scheme accuracy.

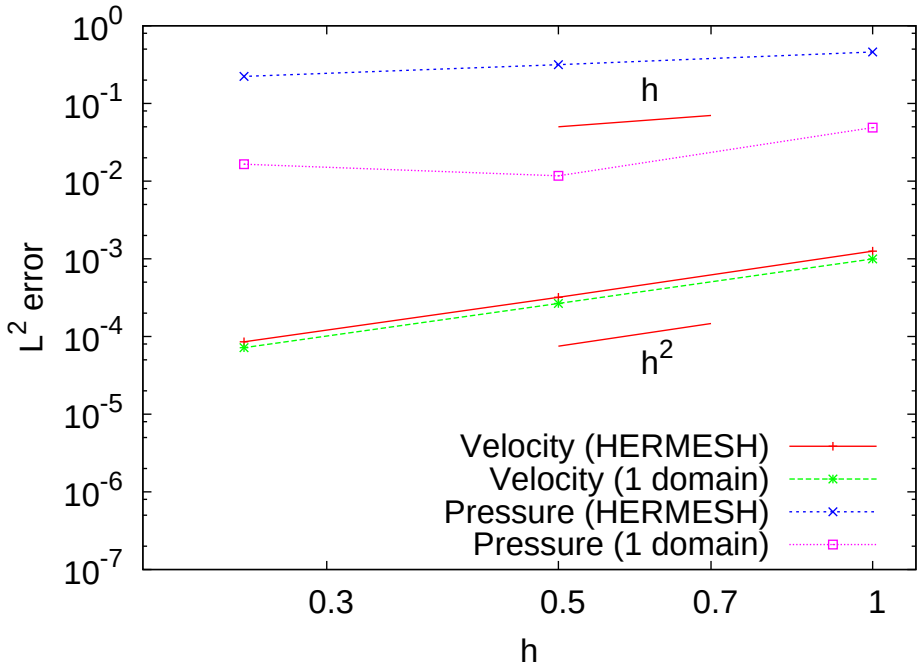


Figure 5.8: Velocity and pressure convergence for one domain and HERMESH method.

Chapter 6

Complex problems solved with Hermesh Method

In this chapter we want to present the real applications solved with the HERMESH method. The presentation of the different cases will be in a chronological way. Since the validity of the method has been demonstrated with different examples during the writing of this manuscript, in this chapter we will just focus on showing real problems solved with the method. As we will see, there are many different situations in which the HERMESH method could be useful. First, we simulate the free surface flow around a ship hull. This is a good example for proving the versatility of the method since three coupled systems are solved in the same problem. Next, we present the simulation of a neuron, corresponding to the solution of the solid mechanics equations. So, once again, the versatility of the method is proved. The neuron is divided into a background mesh corresponding to the body of the neuron and the nucleus is divided into another mesh. This problem could be useful for measuring some properties of the behavior of the neuron submitted to certain pressure conditions. We have proved that the method in solid mechanics equations has a good performance in complex geometries and with different materials, as is the case of the neuron presented in this work, following a previous test in a simple cube domain. Another real problem solved with the HERMESH method is the simulation of the air flow in a wind farm. The wind turbines are approximated by the actuator disk theory and we mesh each disk with a patch mesh and the background is the whole park, so we can simulate different configurations of the turbines inside the park, without remeshing in each test. Another advantage is that we can capture the wake with much less computational effort just by locally refining in the patch but not in the rest of the park. The next problem to be shown lies in small and large human airways simulation. With the HERMESH

method we have been able to join both parts stemming from different research centers. The last application presented in this work also bears on the biomechanical framework; more specifically, it consists of the numerical simulation of the hemodynamics in the cardiovascular system. The idea is to study the viability of the HERMESH method to study the hemodynamic effects of the bypass diameter and incidence angle implanted in the stenosed vessel. If this strategy is valid, the different by -pass could be tested rapidly without having to remesh in each test.

6.1 Ship Hull

In this first real problem, we simulate the free surface flow around a ship hull with the RANS equations together with the Spalart-Allmaras turbulence model, described in Section 2.3 and a hyperbolic equation for the level set, see Section 2.4. We therefore have three coupled systems solved in a staggered manner, as described in Owen, Houzeaux, Lesage, Samaniego, & Vázquez (2013): Navier-Stokes equations, Spalart-Allmaras turbulence model and level set equation, all presented in chapter 2.
The domain has been created with two subdomains in a Chimera-type problem. As we have explained before, the main advantage of the Chimera method in optimization problems or moving components with respect to a complete automatic remeshing stems from the fact that the component meshes move like rigid bodies from one time step/configuration to another. The coupling only affects the vicinity of the interfaces and, if the interfaces are far enough from the body, the impact of the coupling is expected to be minimal. As an example, a boundary layer will remain the same boundary layer. It enables one to respect the tendencies. In fact, a CFD solution always carries an error. By moving the meshes as rigid bodies, without remeshing the whole geometry, it is expected that the error will go the right way. In this sense, let us consider the hydrodynamics of a yacht for which we would want to test two different keels. We would have two options: building two different meshes from scratch for the two configurations, or using the Chimera method in the following way.

- Generating a mesh for the hull including the boundary layer, ignoring the presence of any other object.
- Generating a mesh for each one of the two keels.
- Using the Chimera method to couple the hull with the first keel and, in another simulation, to couple the hull with the second keel.

The drawback of the first strategy, which consists in generating complete meshes for the two configurations is that we may not be able to obtain exactly the same boundary layer mesh around the hull in both cases. Using the

Chimera method, we make sure that the boundary layer will be the same and we do not have to worry about the difference in accuracy from one configuration to another. We are more likely to respect the tendencies of the solutions. Although in this work we have not tested different keels, we have applied our method to a Chimera problem in such a way that the computational domain is composed of two independent meshes. One the one hand, we have generated a mesh for the hull including the boundary layer. On the other hand, we have generated a mesh for the background in which the hull is embedded. We show the successful results next by showing the viability of the HERMESH method for studying the hydrodynamics of a yacht.

Figures 6.1 show the extension elements as well as the mesh. The mesh has been refined around the water-free surface. This makes the creation of extension elements more difficult as the size of the elements from one subdomain to another can be quite different. In Figure 6.2, we can also observe the free surface level with the pressure contour on it.

Figure 6.3 shows the partition obtained with METIS and selecting by-nodes (and not by-faces) to construct the element graph, as we have discussed in 3.6.4.

Finally, Figure 6.4 shows the results obtained for the four unknowns of the problem. We can observe a successful continuity of the solution across the subdomain interfaces despite the large difference in element size from one subdomain to the other.

6.2 Neuron test

This test allows one to prove that the application of the HERMESH method to the solid mechanics equations is valid for a real and complex geometry. Up till now, the HERMESH method applied to these equations has been tested in an end-loaded cantilever two-dimensional beam clamped on the other side or a holed plate submitted to a load in the 0Y direction from the top side and clamped on the base, and has been presented in chapter 5. Before presenting the results obtained in a real neuron test, we have proved the reliability of HERMESH method for a three-dimensional problem with different materials comparing it with a one-domain reference solution.

Preview: Two material cube. The Chimera method is applied here to the solution of a three-dimensional example shown in Figure 6.5. The geometry is composed of two different Neo-Hookean materials, one of which corresponds to the spherical patch mesh which includes a second material. In this example, arbitrary constitutive parameters have been considered for both materials, but material 2 (within the sphere) is stiffer than material 1. The cube is submitted

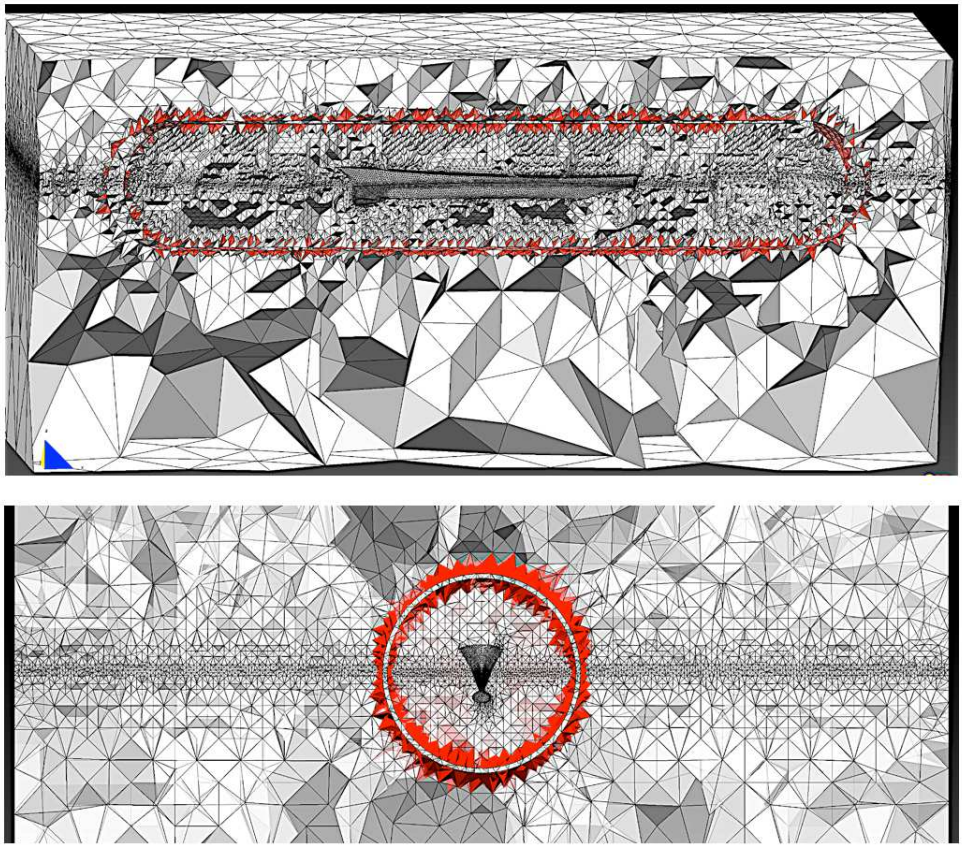


Figure 6.1: Ship Hull. Extension elements and mesh.

	Elements Total	Elements Extension	Elements Hole	Elements Size
Mesh 1	374.848	14.848	17.120	2.910^{-3}
Mesh 2	2.939.664	59.664	151.448	3.610^{-3}
Reference mesh	10.790.400			9.310^{-6}

Table 6.1: Example 2: Number of elements and mesh size.

to an increasing y-displacement on the top face, the bottom face being constrained in the y-direction. Additional lateral boundary conditions are applied to avoid rigid body motion (not shown for clarity).

The problem was solved on two different meshes with the Chimera method: a coarse one of 400,000 elements and a fine one of 3 million elements. A reference solution without the Chimera method was also computed on a refined mesh of 10 million elements. Table 6.1 shows the geometrical details of the three

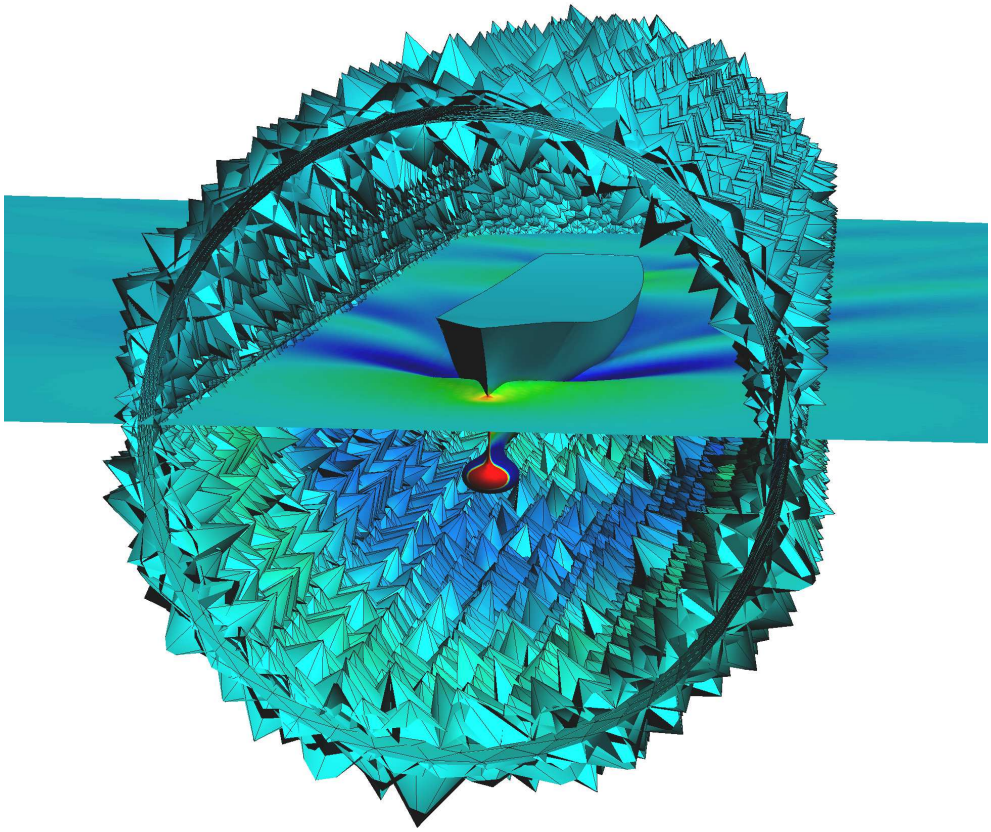


Figure 6.2: Free surface, pressure contours and extension elements.

different meshes. Figure 6.6 shows the displacement and normal stress in the y direction as well as the corresponding mesh obtained on the mid yz-plane for the coarse and fine meshes of the Chimera method. For the Chimera meshes, the extension elements used for coupling the patch and background mesh are demarcated in red. Figure 6.7 shows the solution for an additional cut along the vertical displacement and stress. The solution on both meshes with the Chimera method is compared to the reference solution. A good qualitative agreement is observed for the displacement even for the larger mesh. The stress exhibits a lower convergence, but this should be tempered by the fact that the meshes are only first-order for the solution derivatives. In the overlapping zone, covered by the extension elements, two solutions coexist: the one obtained on the patch and the one obtained on the background. The discontinuity in the stress inside the overlapping zone is appreciable for the coarsest mesh, but this jump decreases when refining the mesh. Nevertheless, the maximum error between the solution with HERMESH computed with the fine mesh and the

Figure 6.3: Mesh partitioning.

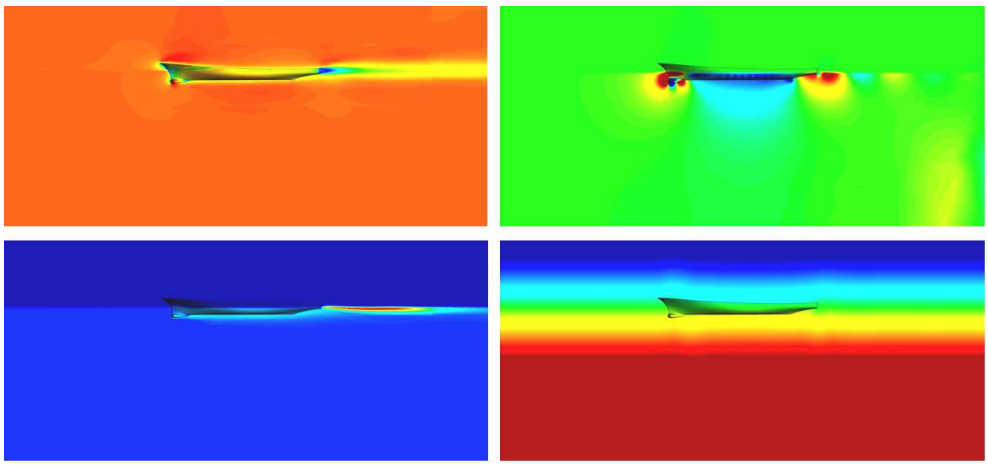


Figure 6.4: (Top) (Left) Velocity. (Top) (Right) Pressure. (Bot.) (Left) Turbulent viscosity. (Bot.) (Right) Level set.

reference solution is of an order of less than 10^{-4}.

Results of neuron test The following problem corresponds to the real geometry of a neuron cell with a soma, an axon, several dendrites and a nucleus under hydrostatic pressure. Figure 6.8 shows the described problem.

The geometry of the neuron-cell was obtained from a real cell using electron microscopy. The two-domain problem is composed of the background, corresponding to the whole body of the neuron and a patch mesh consisting in the nucleus of the neurone inside (Figure 6.9).

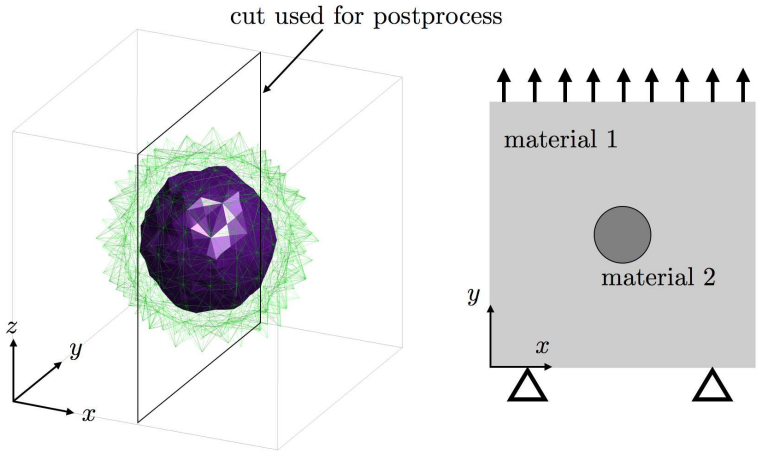


Figure 6.5: Example 2: extension elements and hole (left) and boundary conditions (right).

Nucleus and soma material models are different and representative for a neuron cell. In particular, material parameters representing a (homogenized) neuron cell used in this work for the soma material are given as follows: 1^{st}-Lamè constant $\lambda_0 = 12$ kPa, shear modulus (2^{nd}-Lamè constant) $\mu_0 = 0.5$ kPa, and mass density $\rho_0 = 10^{-12}$ g/μm^3 (taken the same as water, $\rho_0 = 10^3$ kg/m^3). And the properties used for the nucleus are given al follows: 1^{st}-Lamè constant $\lambda_0 = 36$ kPa, shear modulus (2^{nd}-Lamè constant) $\mu_0 = 1.5$ kPa, and mass density $\rho_0 = 10^{-12}$ g/μm^3 (taken the same as water, $\rho_0 = 10^3$ kg/m^3). In this case, HERMESH method creates a hole (nucleus) in the background mesh (soma) and construct the extension elements in both interfaces, the background interface and the patch interface (Figure 6.10).

The problem was aimed at studying the deformation of neuron-cells under mechanical loading, e.g. pressure. With this kind of analyses, the mechanical behavior of the cell under external forces can be studied, to assess whether a certain load may induce damage to the cell or not. The results obtained in this problem are shown in Figure 6.11. We can observe the continuity of the stresses and displacements between two meshes.

Acknowledgements This test has been possible thanks to the collaboration of two research centers. The geometry has been extracted from confocal microscopy images by Prof. Antoine Jerusalem's group (Computational Mechanics of Materials Group, University of Oxford) and has been provided by the Laboratorio Cajal de Circuitos Corticales, CTB-UPM/Cajal-CSIC, which

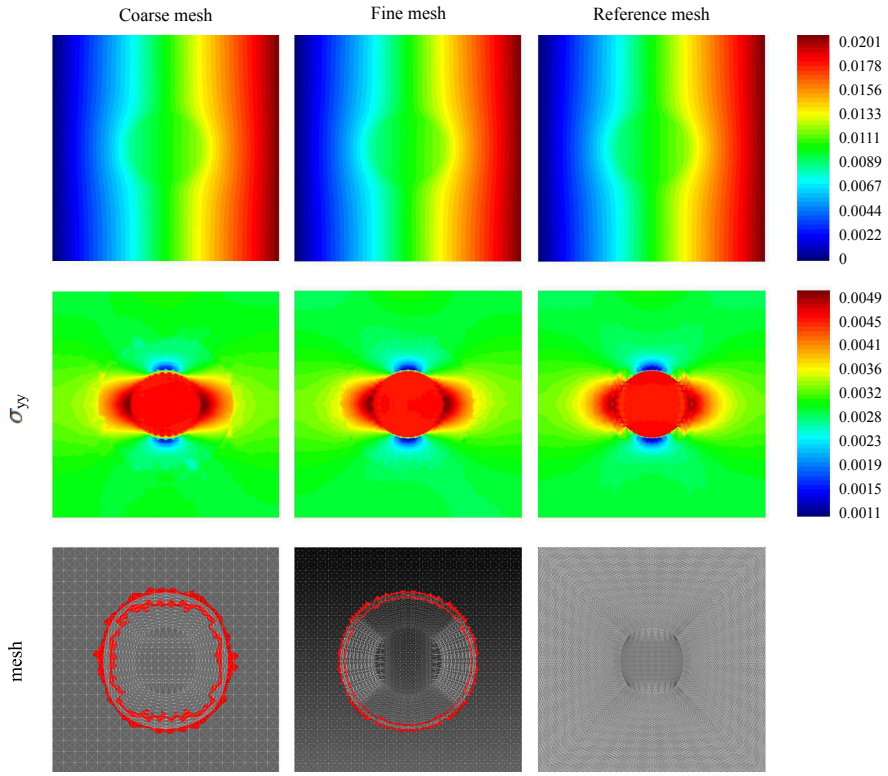


Figure 6.6: Example 2: solution on mid yz-plane for the three meshes: y-displacement (top) and stress along the y-direction (middle) and computational mesh (bottom).

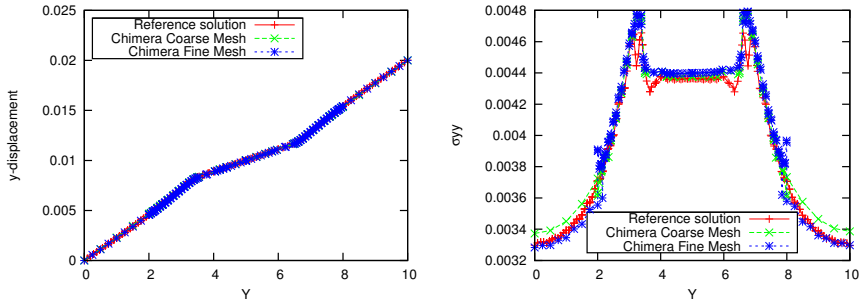


Figure 6.7: Example 2: solution along mid line of mid yz-plane for the three meshes: y-displacement (left) and stress along the y-direction (right).

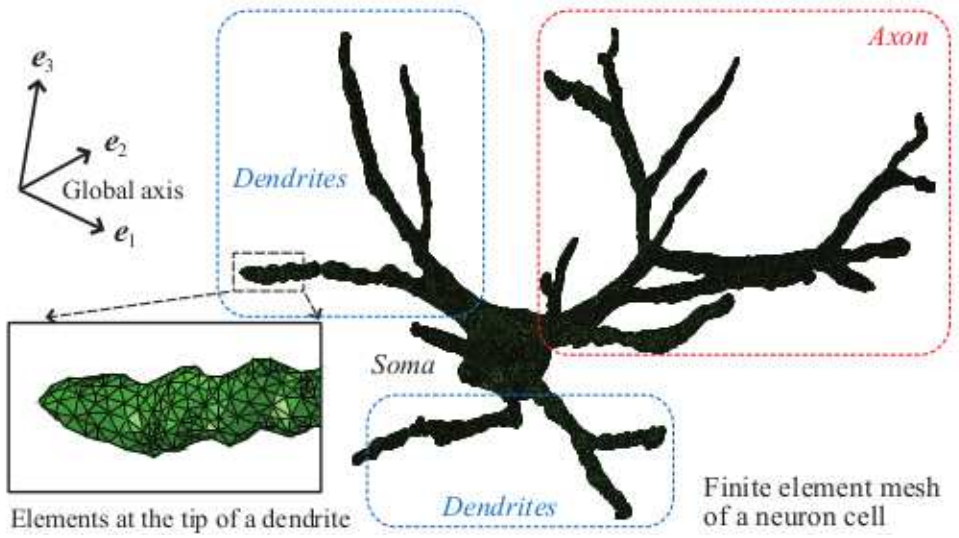


Figure 6.8: Finite element mesh with 139921 linear tetrahedral elements, discretizing a neuron cell (a soma, an axon and several dendrites).

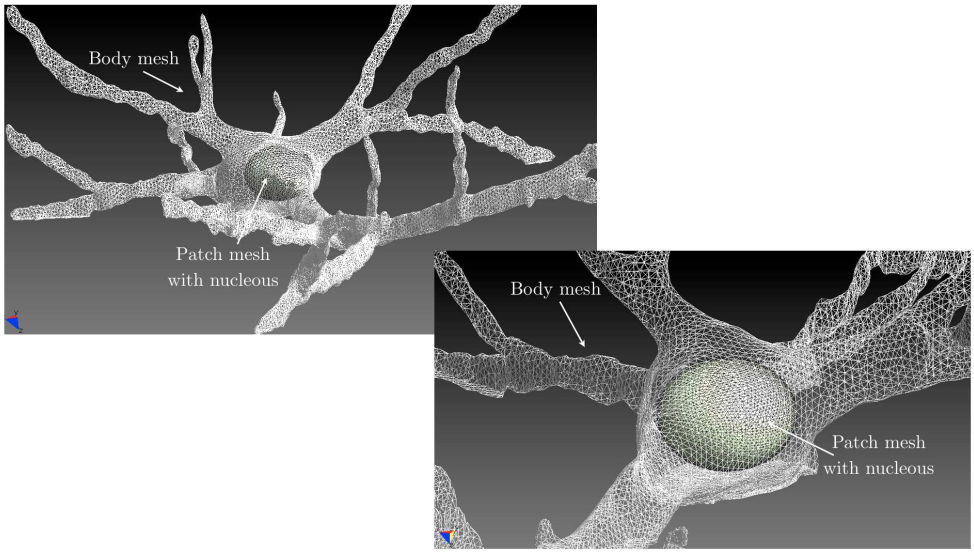


Figure 6.9: (Left) White mesh: background mesh containing the body. Green Mesh: patch mesh containing the nucleus. (Right) Zoom

is involved in the Blue Brain Project (BBP) with an initiative named Cajal Blue Brain (http://cajalbbp.cesvima.upm.es/)

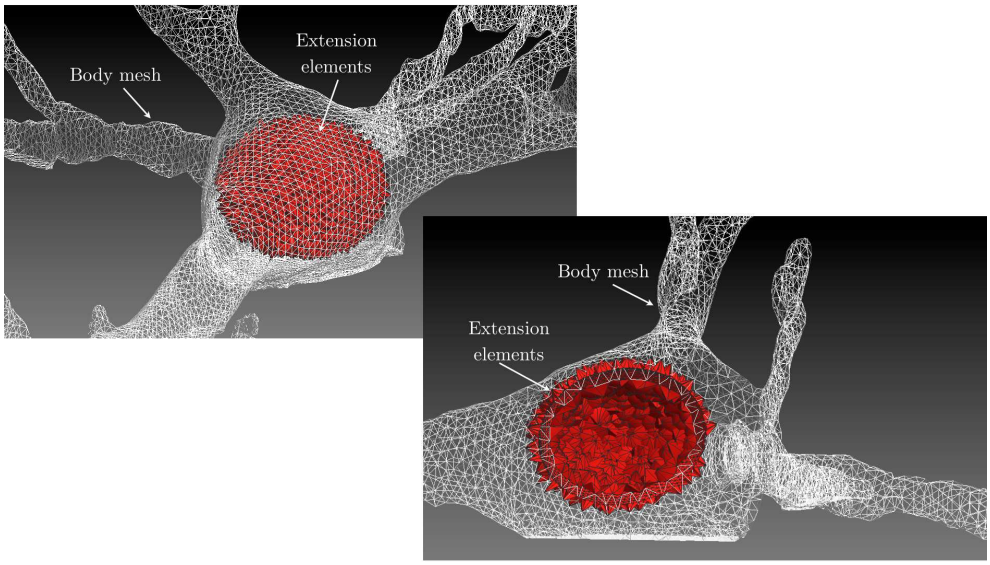


Figure 6.10: (Left) White mesh: background mesh containing the body Red Mesh: Extension elements from the patch to the background. (Right) Cut showing both extension elements from the patch to the background and from the background to the patch.

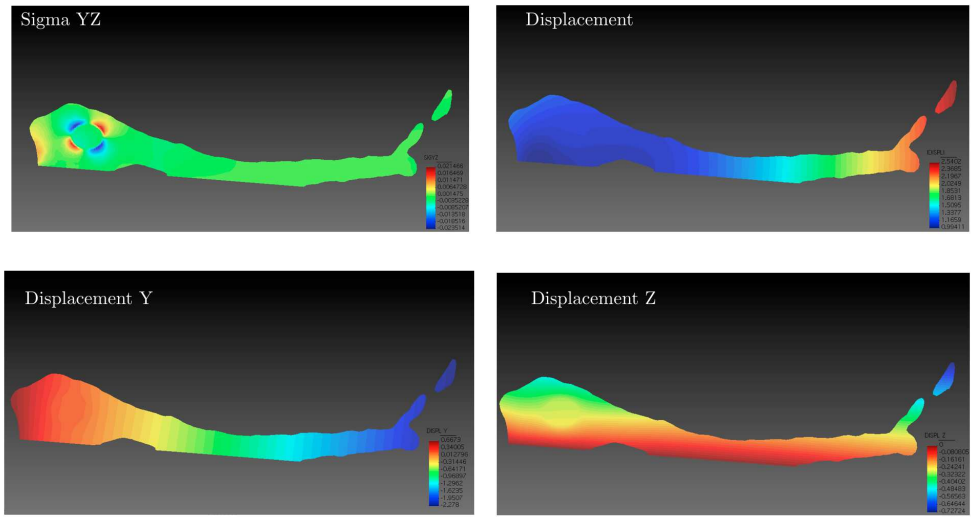


Figure 6.11: Neuron results in a cut.

6.3 Wind Farm

We present a CFD modeling strategy for wind farms aimed at predicting and optimizing the production of farms. The CFD model includes meteorological

data assimilation, complex terrain and wind turbine effects. The model involves the solution of the RANS equations together with a k-ε turbulence model especially designed for the Atmospheric Boundary Layer. As the integration of the model up to the ground surface is still not viable for complex terrains, a specific law of the wall including roughness effects is implemented. The wake effects and the aerodynamic performance of the wind turbines are accounted for using the actuator disk model, upon which a volumetric force is included in the momentum equations. The placement of the wind turbines and a mesh refinement for the near wakes is achieved by means of a Chimera method based on HERMESH coupling.
In order to reliably evaluate wind farm efficiency, simulating the turbulent wakes generated from the turbines is a necessary condition. This turbulent wakes produce changes in the flow received by the neighbor turbine, meaning that efficiency is deteriorated. The CFD technique is used to capture this microscale phenomenon since it gives a detailed description of the wind flow passing through a wind farm. The problem is that the application of this technique entails discretizing the computational domain with a very fine mesh size, the cost of which could eventually become computationally unaffordable.
With the application of the HERMESH method in the simulation of wind farms, we are able to locally discretize around the turbines with a finer patch mesh located inside the background mesh that covers the whole computational domain. So, with our method we can combine a structured mesh which represents the background and non-structured and fine patch meshes representing the turbines. We want to stress that one of the best achievements of the application of the HERMESH method to the wind farm simulation lies in the optimization purpose, as in each configuration of the computational domain composed of the turbines, remeshing is not needed.
The first result that we want to show before carrying out the simulation described earlier is that the HERMESH method could be valid as a refinement technique. As we have explained, the patch mesh in the wind farms simulation contains a finer mesh than the background. So we have had to test how sensitive the method was to the difference in size in the coupled meshes. This is shown in the next section with a simple domain for the cavity flow problem.

Preview: Local refinement in Navier-Stokes problem. In this example we present the application of the HERMESH method as a local refinement technique. To illustrate this application, the cavity flow at a Reynolds number 5000 is considered, on a unit square $[0,1]\times[0,1]$, solved on uniform and rather coarse meshes (finest one has only 50×50 elements). The Chimera-type problem solved with HERMESH is compared to some one-domain solutions and with Ghia's results; see the reference Ghia, Ghia, & Shin (1982). For the HERMESH method, the patch mesh is located on the bottom right corner, where the flow

exhibits a strong recirculation. Four solutions are going to be compared:

- Mesh 1: one-domain solution on a coarse mesh (10×10 elements).
- Mesh 2: one-domain solution on a fine mesh (50×50 elements).
- Mesh 3: HERMESH solution where the background and patch meshes are of the same size as Mesh 1 and Mesh 2, respectively.
- Mesh 4: HERMESH solution where the background mesh is of the same size as Mesh1 and the patch is twice as fine as Mesh2.

The four meshes are shown in Figure 6.12.

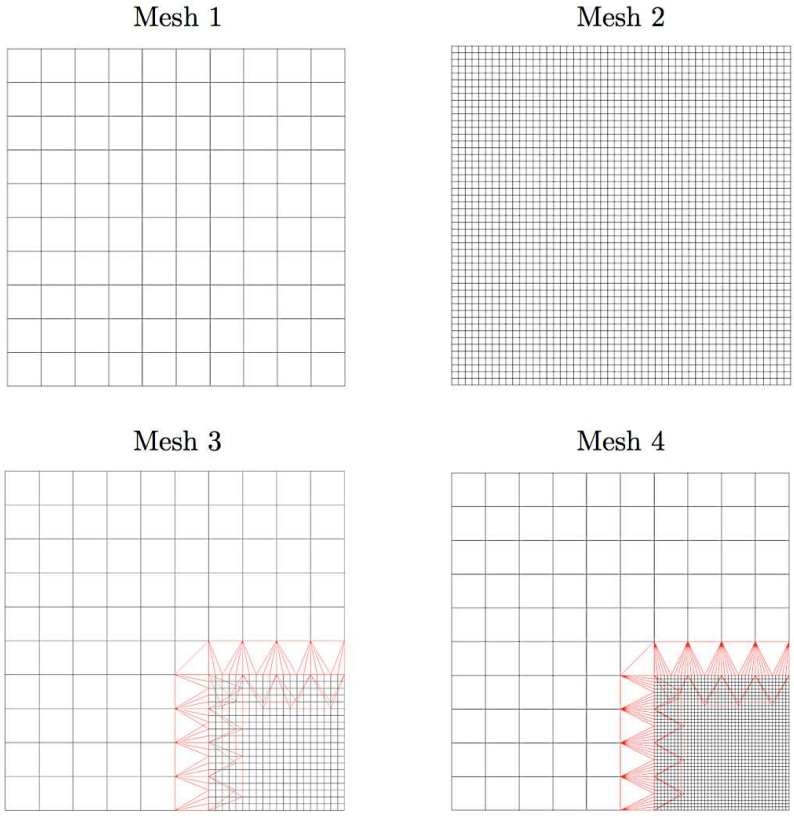


Figure 6.12: Local refinement. (Top) One-domain meshes (Mesh 1 and Mesh 2). (Bot.) Meshes for HERMESH with extensions in red (Mesh 3 and Mesh 4).

Figure 6.13 shows the positive effects of the patch mesh for capturing recirculation. The solution was obtained with Mesh 3. Without using a patch

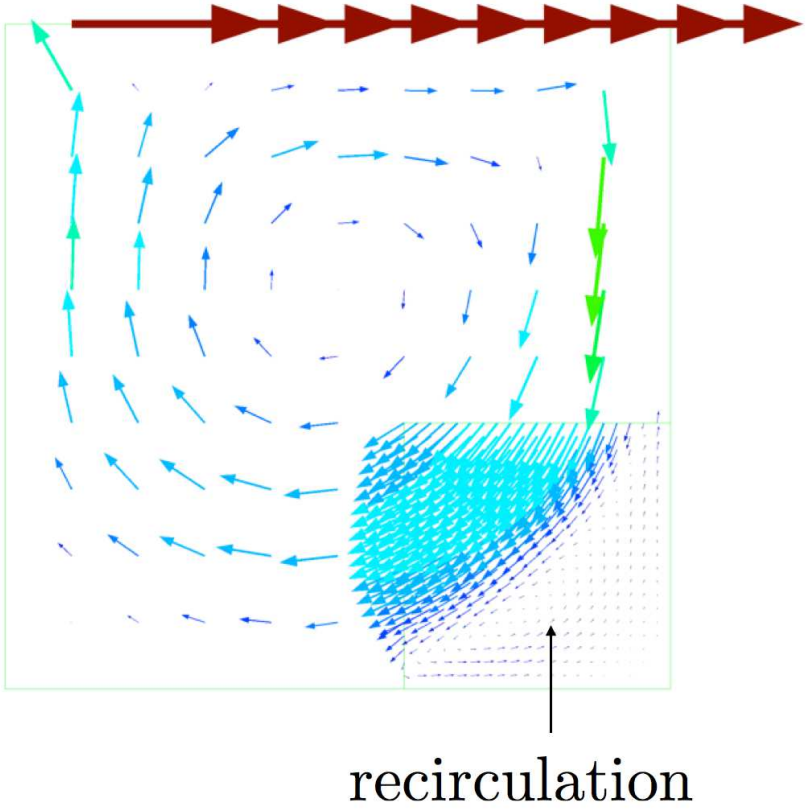


Figure 6.13: Local refinement. Recirculation captured by the patch mesh on Mesh 3.

mesh as a local refinement, no recirculation at all could be obtained, as only very few nodes were located in the recirculation area.

Figure 6.14 compares the solutions obtained on one horizontal cut and one vertical cut. The coarse mesh (Mesh 1) does not capture the recirculation, partly captured by the fine mesh (Mesh 2). Both HERMESH methods enable one to rectify the solution not only in the recirculation zone but also at the core of the cavity. We can also observe that the very fine patch of Mesh 4 gives almost the same solution as Mesh 3: the error is dominated by the coarse mesh.

Turbine turbulent wakes. Once the local refinement application of the HERMESH method is proven, we are going to show the results of the method obtained in the simulation of turbines with an actuator disk theory.
As noted before, in the current work the wind farm simulation consists in solving the Reynolds Averaged Navier-Stokes equations with a specific k-ε turbulence model described in Section 2.3, on a real topography, using real

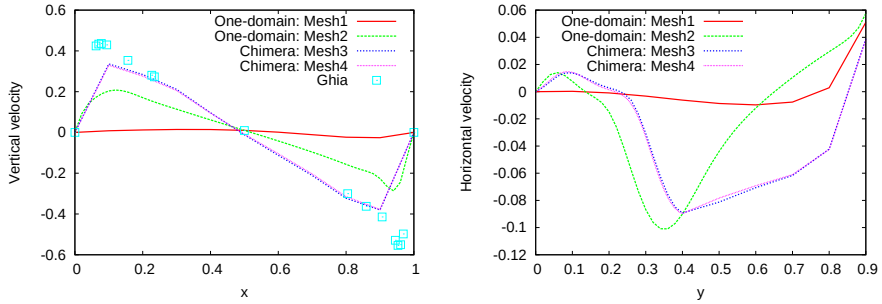


Figure 6.14: Local refinement. Comparison of solutions obtained on Mesh 1, Mesh 2, Mesh 3 and Mesh 4. (Left) Vertical velocity along horizontal cut at $y = 0.5$. (Right) Horizontal velocity along horizontal cut at $x = 0.9$.

wind measures as boundary conditions. Although other approximations, such as the virtual blade model are possible, see for example Hussein & El-Shishiny (2012), in this work the actuator disk theory is applied. This theory is used to account for the wind turbine effects. Using this approximation, the momentum needed for putting the wind turbine in motion is extracted from the Navier-Stokes equations. The real rotor is approximated by a permeable thin cylinder of equivalent area A, and the total linear momentum sink is distributed uniformly within a volume V. The rotor model used in this work is based on the one-dimensional axial momentum theory for a uniformly loaded rotor and non-rotating flow in which the change of momentum is only due to pressure differences across the actuator disk. The expression of the force sink module is:

$$F = \frac{1}{2}\rho A C_t U_\infty^2. \tag{6.1}$$

In the last equation, the force is expressed in terms of the upstream velocity module U_∞, and the rotor thrust coefficient C_t. This coefficient is rotor dependent and determined experimentally. Taking into account the actuator disk, the resulting Navier-Stokes equations read:

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nabla\cdot\left(\left(\mu+\mu_t\right)\boldsymbol{\varepsilon}(\boldsymbol{u})\right) + \nabla p + 2\rho\omega\times\boldsymbol{u} = \rho\boldsymbol{f} + \delta_V F/V\boldsymbol{n}, \tag{6.2}$$

where δ_V is the delta Dirac function with support in the cylinder V and $\boldsymbol{n}$ is the cylinder unit normal pointing towards the incoming flow.
To implement the actuator disk theory in a finite element code, we face two difficulties. The first one entails the geometrical representation of the rotor inside the mesh. The second one concerns the accuracy of the solution in the wake of the rotor. Regarding this last point, accurate calculations in the neighboring of the rotor are essential for a proper estimation of the wind farm power. This

requires mesh refinement in the wake, which is computationally problematic when dealing with structured or Cartesian meshes. As far as the geometrical representation of the cylinder is concerned, we have several options. On the one hand, the mesh can be locally adapted to represent the disk in which the momentum is extracted. This technique requires adapting the mesh for each configuration and is quite difficult to implement for structured and Cartesian meshes. In fact, one direction should necessarily be aligned with the cylinder. On the other hand, the Chimera technique offers a viable alternative for facing the two aforementioned difficulties.
A good review of computational fluid dynamics for wind turbine wake aerodynamics simulations can be found in Sanderse, der Pijl, & Koren (2011). To the author's knowledge, the application of Chimera methods to the simulation of micro-scale wind farms is new, although the Chimera method has been used to simulate single wind turbines, as is the case of the work: Zahle, Sørensen, & Johansen (2009). The application of the Chimera method in the optimization of wind farm problem consists in having an independent patch for each turbine and coupling all of them to the background mesh which contains the topography. Due to the log profile of the flow velocity and rapidly decreasing ε away from the wall, a boundary layer mesh is unavoidable. Figure 6.15 illustrates the idea, where 28 patches were superimposed on a structured background mesh attached to the topography.

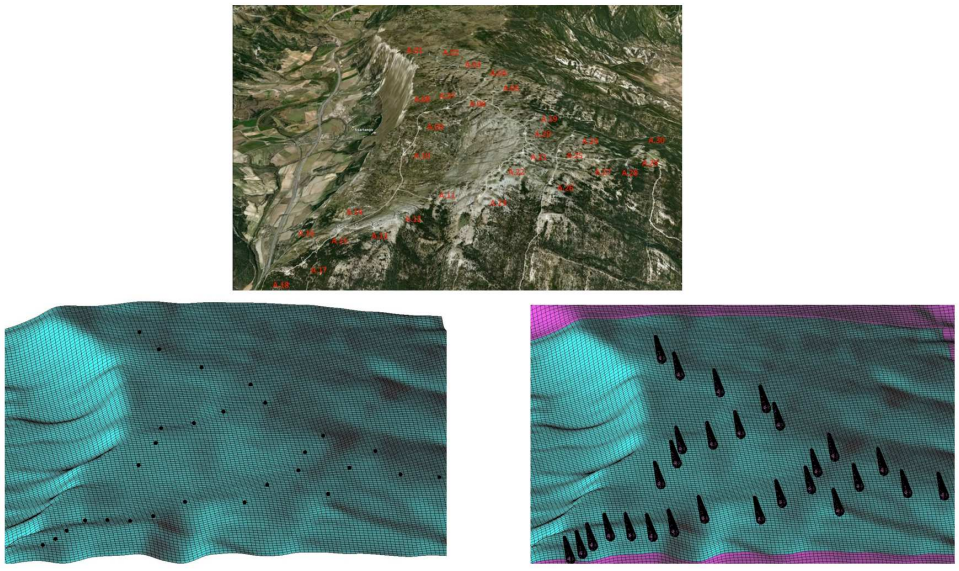


Figure 6.15: Wind Farm. Example of topography and patches with actuator disks.

To see with more detail the composed mesh, Figure 6.16 illustrates a cut of this kind of meshes showing the disk, the patch interface as well as background

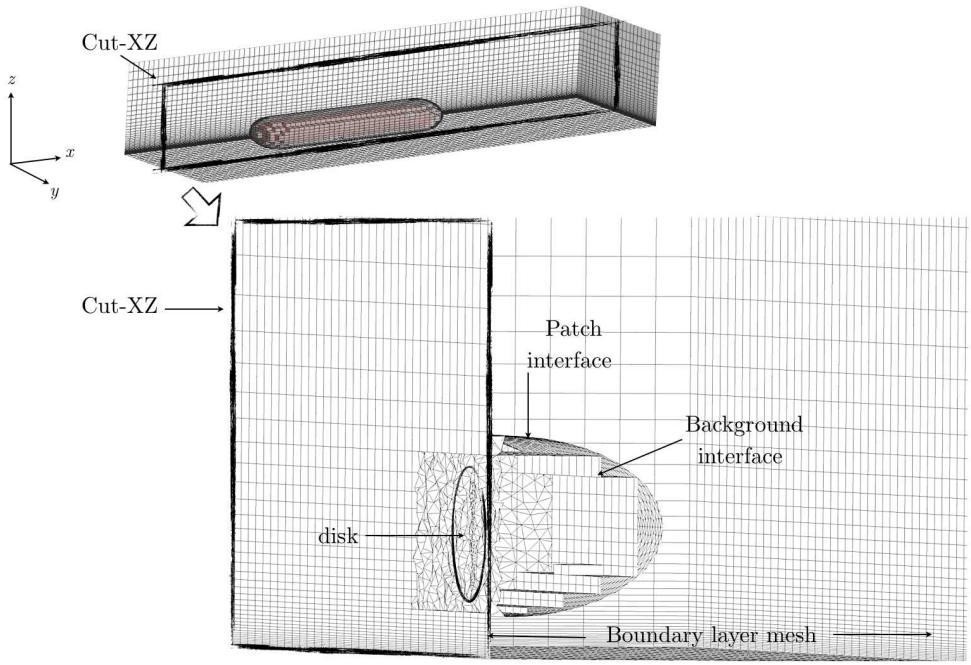


Figure 6.16: Cut-XZ: Disk inside the patch mesh and hole background interface.

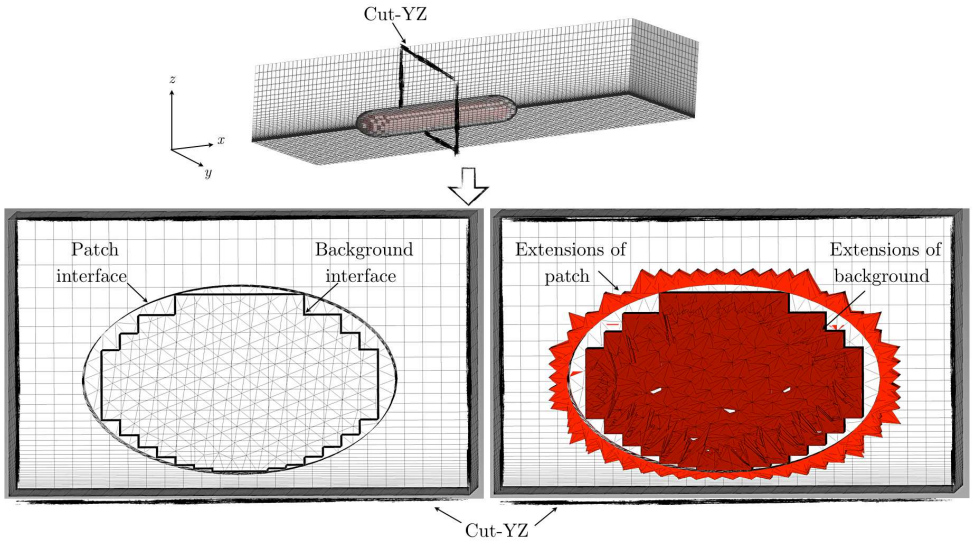


Figure 6.17: Cut-YZ. (Left) Patch and background interfaces. (Right) Extension elements of patch and background interfaces.

interface. Figure 6.17 shows the orthogonal cut to the previous one with and without the extension elements in both interfaces. As we can see in previous figures, the background mesh for these problems consists of hexahedra with a boundary layer refinement near the ground. This entails another issue in the creation of the extension elements. First, the extension elements that connect the background mesh with the patch meshes are pyramids because the outer boundary of the hole is formed by quadrilaterals, as was explained in Section 3.5.2. Second, as some quadrilaterals of the hole boundary may come from anisotropic hexahedra, one could require the pyramids to inherit the aspect ratio of the hexahedra, as is also shown in Section 3.5.2.
We have compared the one-domain solution with the Chimera method for a single wake case, extracted from the experimental Sexbierum testCleijne (1992). In Figure 6.18 we show the difference between Chimera-background mesh and one-domain mesh. In Figure 6.19 we can see how both cases, one-domain and

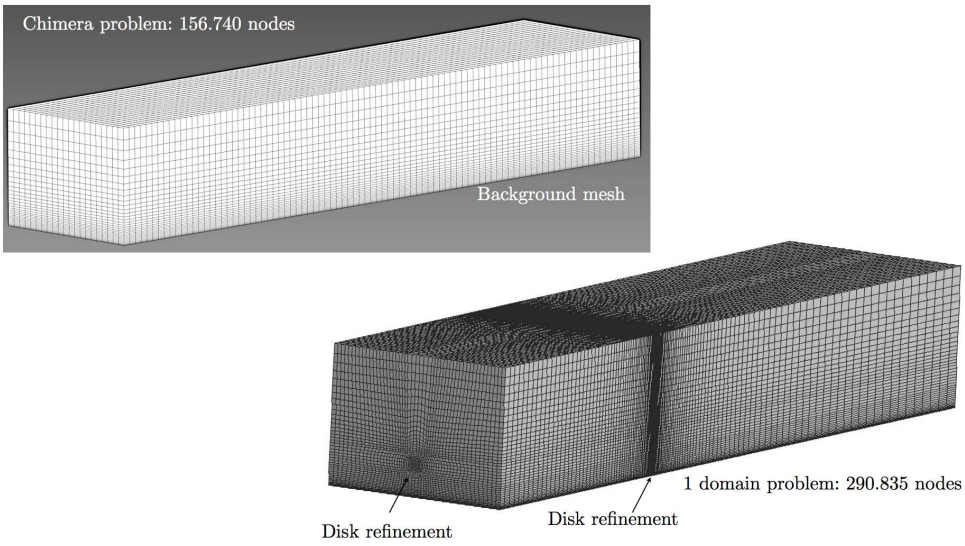


Figure 6.18: (Top) Chimera background mesh without refinement. (Bottom) One-domain mesh with refinement.

Chimera domain, reproduce very similar results. It is worth mentioning that the number of nodes of the one-domain mesh is 290,835 while 156,740 were sufficient in the case of the Chimera method. This difference is due to the fact that with one-domain mesh, the refinement around the disk is propagated during the whole domain, while in Chimera-type problem this refinement is only necessary in the patch mesh. The difference in the number of nodes can be significant when considering tens of wind turbines. In Figure 6.19 we observe that the difference in the velocity drop down to the turbines between one and two subdomains is of no significance.

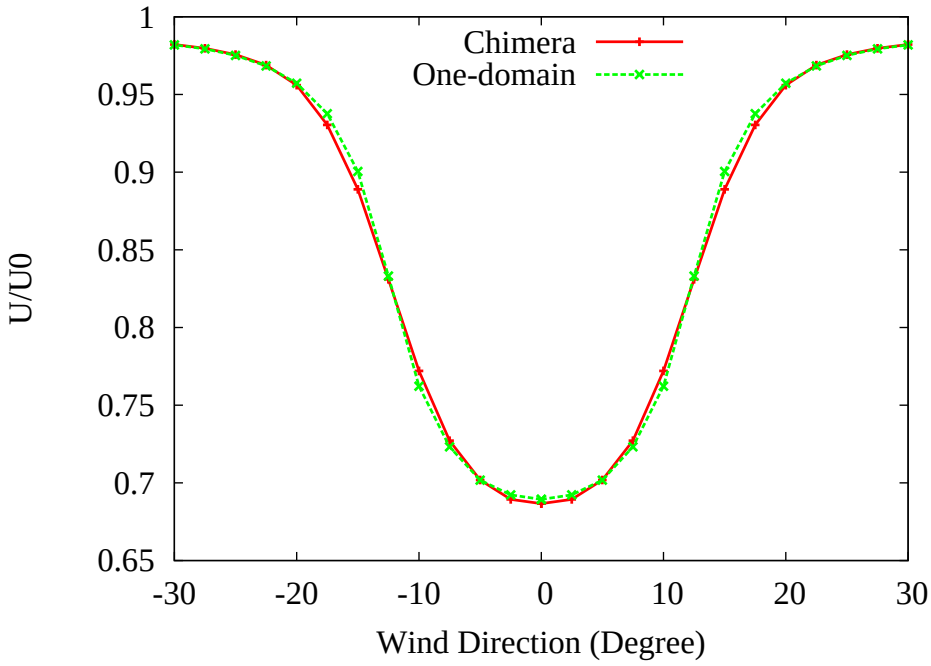


Figure 6.19: Wind farm. Wind speed deficit 2.5D downstream. Comparison between one-domain and Chimera method.

Figure 6.20 shows the results obtained for two wind turbines, using isotropic pyramids, on a flat topography. The second turbine is located 2.5 rotor diameters downstream from the first one. The left part of the figure shows the patch boundary meshes. The other figures on the right show the solution (velocity, pressure and turbulent viscosity) obtained on a vertical cut passing through the middle of the actuator disk.

The results point out that the HERMESH method is a proper strategy for this problem since it overcomes the drawbacks presented when one has to face simulations of this kind.

Acknowledgements. This work has been sponsored by an Fundación IBERDROLA grant.

6.4 Joining large and small airways

This application originates from a joint research between the Barcelona Supercomputing Center (BSC), Imperial College (IC) and Jackson State University (JSU). From tomography images, IC obtains a mesh for the large airways down

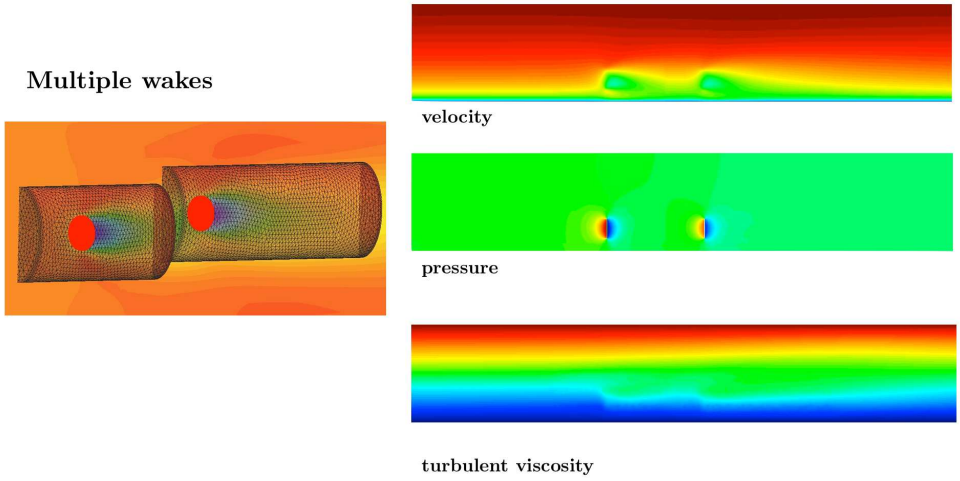


Figure 6.20: Double weak test. Patch meshes and solution.

to the trachea, see Doorly, Taylor, Gambaruto, Schroter, & Tolley (2008) for more details; permission to use the patient image data was obtained from the physician in charge with the patient's consent. On the other hand, JSU automatically generates an arbitrary number of generations of the idealized bronchopulmonary tree, see Soni & Aliabadi (2013). The task of BSC during the preprocess phase is to join the two meshes using the HERMESH method presented in the previous section, as illustrated in Figure 6.21. Then, the global mesh is ready to be treated to solve the transient flow. The utility of this joining procedure is to provide realistic inlet conditions for the study of the flow patterns and particle deposition in small airways.

The joining procedure of the preprocess step is not straightforward, as one major issue must be overcome before carrying out the simulation: the bronchopulmonary tree starts as a perfect cylinder, which is not the case of the large airways mesh, see Figure 6.22 (Top). This mesh comes in fact from tomography and is patient-dependent. Therefore, a correction procedure is necessary to adapt the crown of the inlet of the small airways to the outlet crown of the large airways. We project the nodes of the first crown onto the geometry of the second crown by prescribing their displacements, and then carrying out a volume mesh motion to adapt the geometry smoothly downwards. This is carried out by solving a weighted Laplacian, to conserve the boundary layer mesh, using the aforementioned displacement as the Dirichlet boundary condition.

Once the inlet and outlet are adapted, we apply the HERMESH procedure to create the extension elements from both interfaces, as is illustrated in Figure 6.22 (Bot.). In this case, both interfaces were of the *known* types: the interface

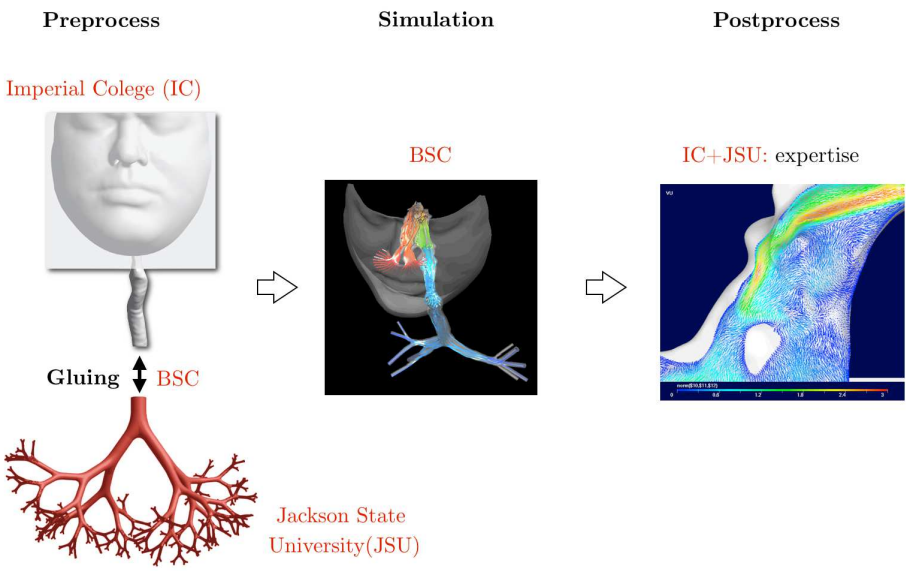


Figure 6.21: Joint research

of the large airways is their outlet section while that of the small airways corresponds to their inlet section. Figure 6.23 shows the extension elements in the joining region. We observe a large difference in mesh sizes from one side to the other. The mesh of the large airways includes a boundary layer mesh while the mesh of the small airways is almost isotropic. In spite of this, the extensions were created properly.

Figure 6.24 shows the mesh and the velocity field at one time step of the simulation, using real respiration conditions (in this case, a sniff). Continuous primary variables are obtained across the extension elements.

Acknowledgements. This research was motivated by an ongoing PRACE project involving BSC and IC, for simulating the transient flow in large airways. The research at JSU was carried out under the National Science Foundation (NSF) EPSCoR grant (EPS-0903787).

6.5 Flow passing through a bypass in a stenosed artery

Atherosclerosis is one of the major causes of death in the western world. The numerical simulation of the hemodynamics in the cardiovascular system has proven to be an important tool for the understanding of the problem. One of the aspects is the prediction of the post-operative flow and the related

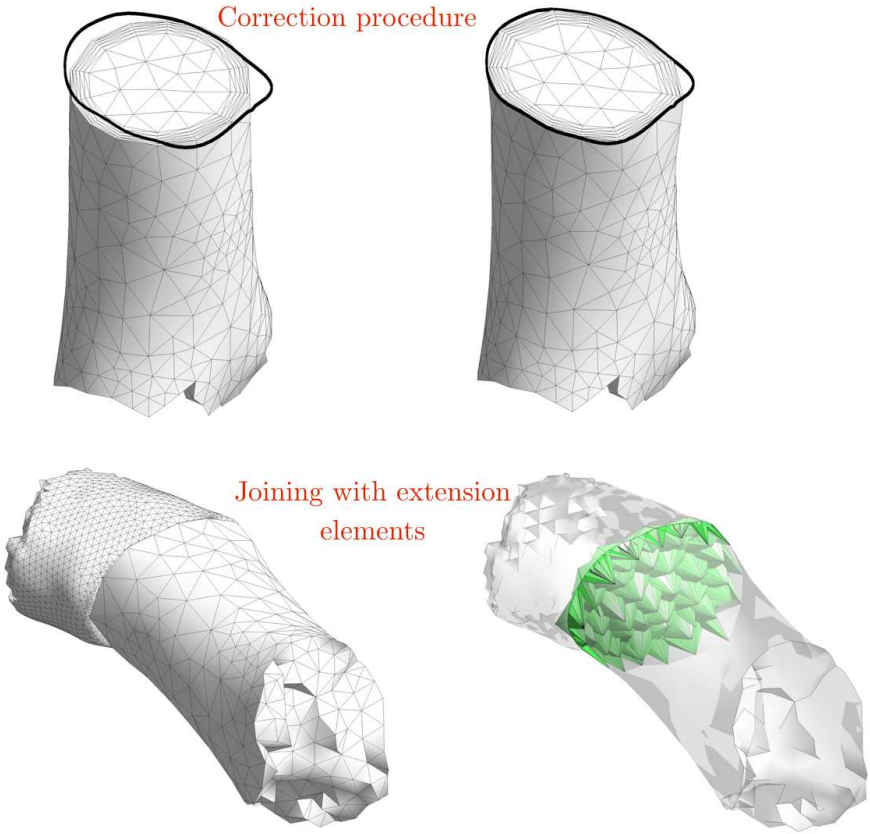


Figure 6.22: Top: Adapted crown. Bottom: Extension elements.

global parameters for helping doctors to determine the best option for the treatment of vascular disease. The geometrical aspects in vascular interventions affect the hemodynamic and permeability of the intervention. The study of the influence of the caliber of a bypass or insertion angle would be very useful for reducing the graft occlusion and determining the best configuration in a vascular intervention. We propose applying the HERMESH method to couple the bypass with the vessel. In this way, the geometry of the bypass can be easily changed in order to test various configurations (angles and diameters), without any need to remesh the complete computational domain.

We take into consideration a realistic highly stenosed segment with a length of 24 cm and a diameter of 1 cm in unaffected zones. This configuration could correspond to a femoral or iliac artery. The idea is to study the viability of the HERMESH method for studying the hemodynamic effects of the bypass diameter and incidence angle. For the sake of simplicity, we assume the following hypothesis: density $\rho = 994$ kg/m^3 is constant; the fluid is Newtonian with

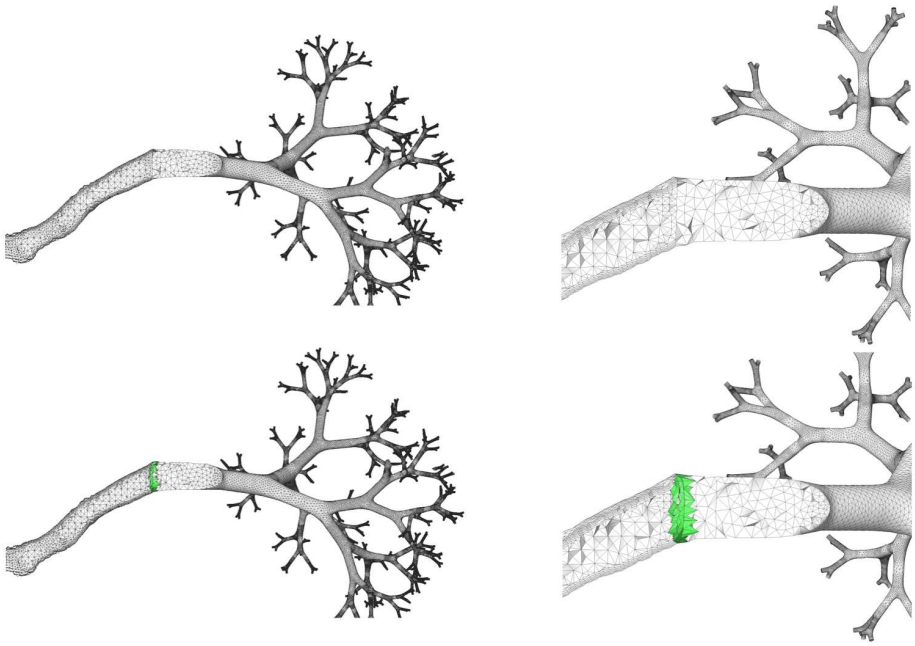


Figure 6.23: Extensions, large and small airways meshes.

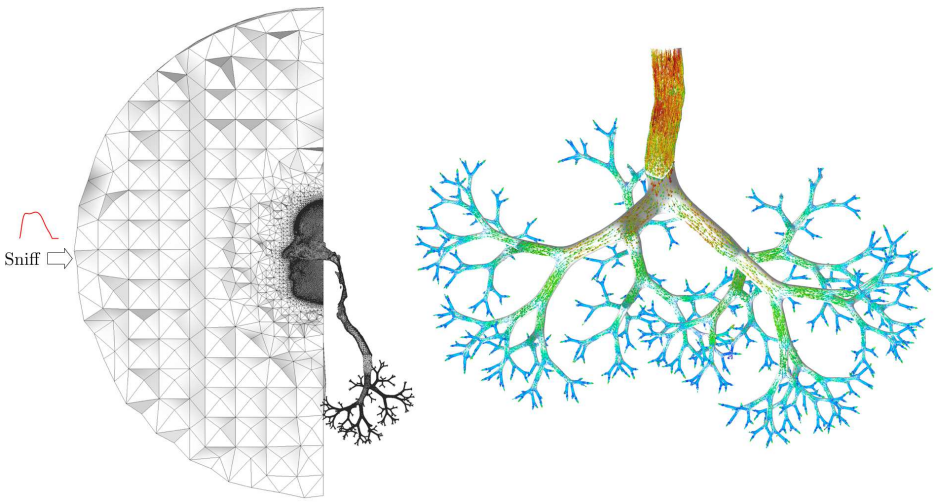


Figure 6.24: (Left) Resulting mesh. (Right) Velocity field.

viscosity $\mu = 0.0036$ Pa $\cdot$ s; the walls of the artery are rigid; velocity at inlet is parabolic and steady and pressure at outflow is constant. The computational domain consists of two independent meshes. One corresponds to the stenosed

segment of the artery and the other one to the bypass.

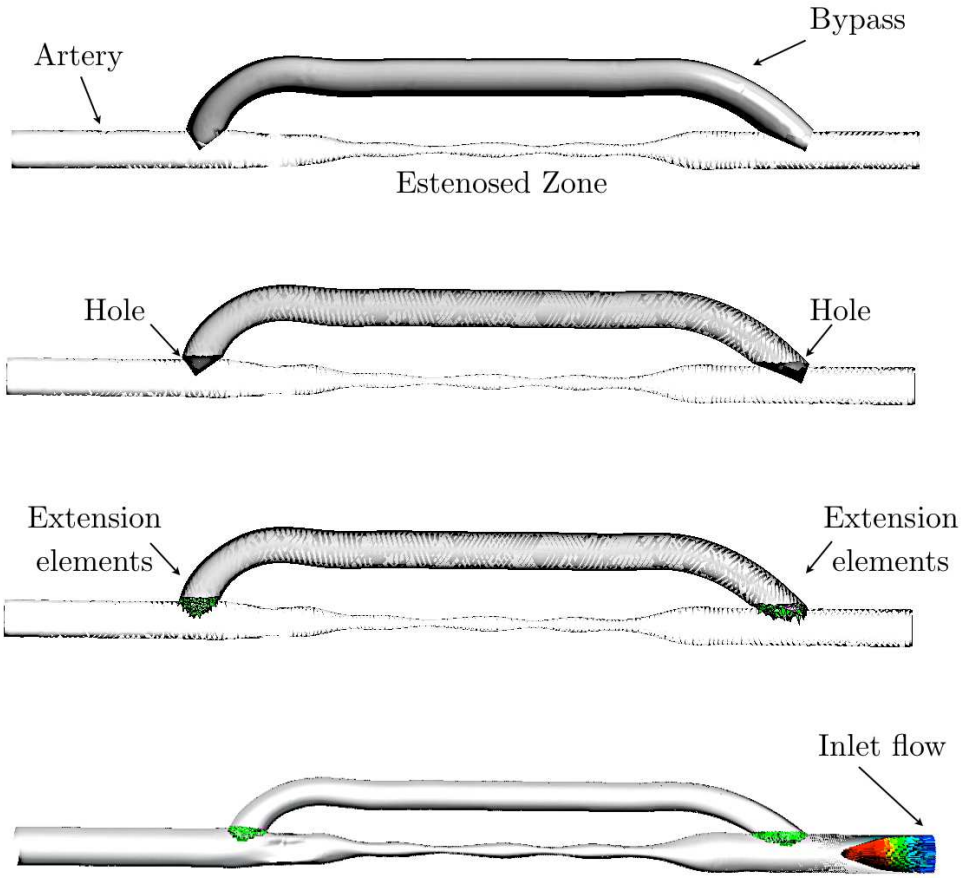


Figure 6.25: Geometry of artery and bypass

The geometry of the problem is shown in Figure 6.25. As we can see in the figure, the bypass mesh consists of two non-connected interfaces to be coupled to the artery mesh. So in this case, we cannot define the interfaces as *unknown/unknown*, following the nomenclature presented in Chapter 5, because the *shadow* of the bypass projected to the artery becomes half of this artery! Nor can we define the interfaces as *known/unknown* without overlapping, because the curvature of the vessel would entail excessively large extension elements. Our solution to this case is defining as *holed-unknown/unknown* by imposing some overlapping between the meshes, as in the example in Figure 5.1-d. Figure 6.26 shows a zoom of the extension elements.

Finally, in Figure 6.27 we show some results of the velocity and pressure fields. Note that in this case, the gluing procedure is carried out in a critical region of the flow, where a small fraction of the fluid goes straight to the artery

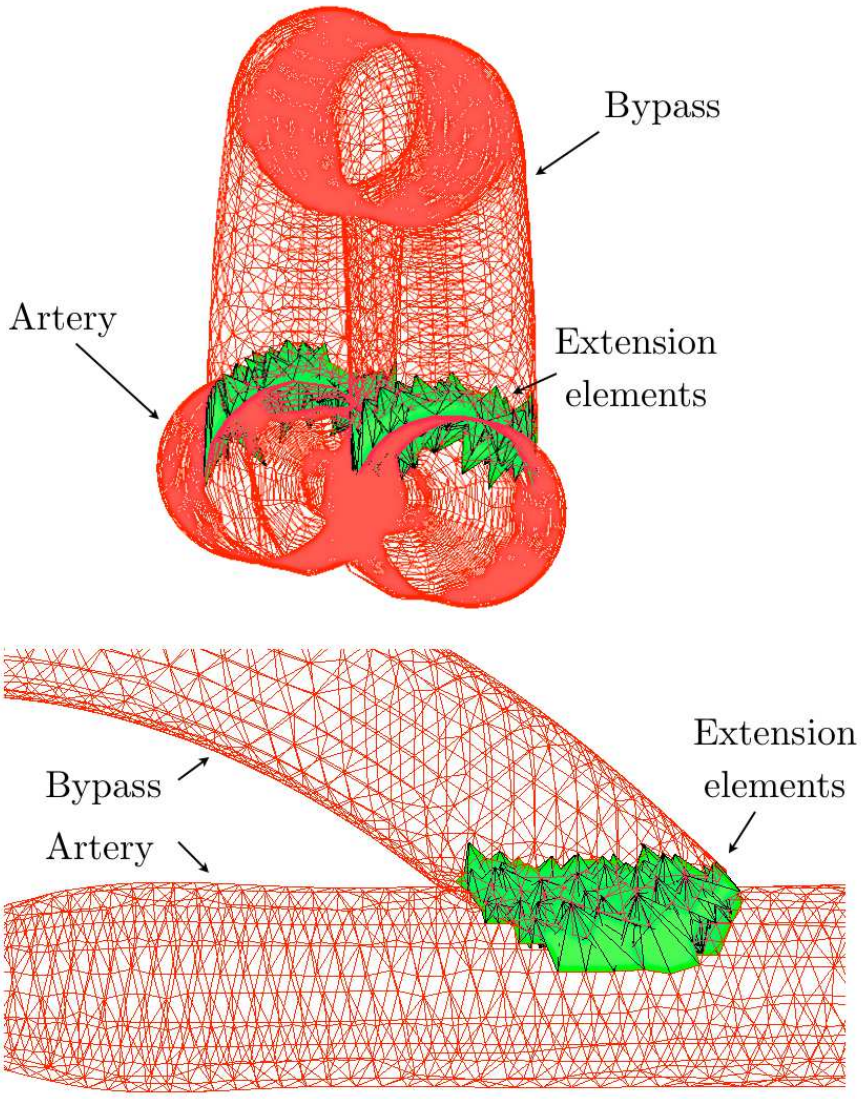


Figure 6.26: (Top) Extension elements in artery and bypass mesh. (Bot.) Zoom of the extension elements.

and the main fraction of it goes through the bypass. Despite this, we observe a good continuity of the main flow variables.

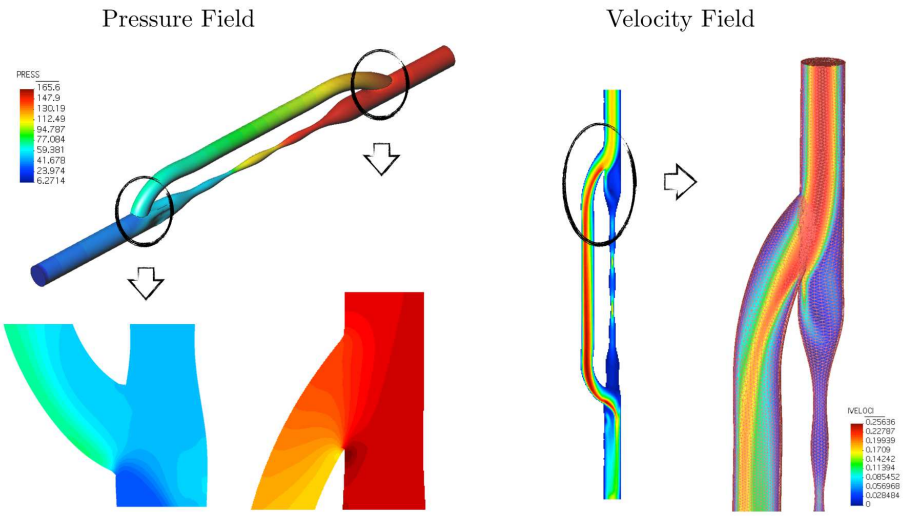


Figure 6.27: (Top) Pressure field. (Bot.) Velocity field.

Chapter 7

Conclusions and future work

During all this work we have focused on mesh composition techniques in the framework of the finite element method and applied to computational mechanics simulations. To be able to couple independent meshes in an efficient manner is a crucial point in the area. This allows to face problems like: to assemble components obtained from different sources, to simplify the meshing of complex geometries, to perform local refinement, to couple multi-physics problems, to couple subdomains in relative motion or to optimize the relative positions of some components, without having to remesh the whole computional domain. These approaches are quite common and powerful to solve complex problems. In this work we have developed a new method which addresses some of these approaches applied to any partial differential system of equations. Problems in fluid mechanics coupled with turbulence models and level set equation as well as solid mechanics and advection-diffusion-reaction (ADR) equation have been explored.

The first topic we have presented in this work is a multiscale variational method to solve the ADR equation. A methodology was developed to obtain a solution to the subgrid scale. It consists in solving the one-dimensional subgrid scale with constant coefficients by: firstly, requiring that the subgrid scale is zero on the nodes; secondly, assuming that the subgrid scale residual is constant inside the element; finally, averaging the result over the element. The method gives nodally exact result in the case of the advection-diffusion equation, and justifies the use of h/2 in the definition of the stabilization parameter for quadratic elements.

As a first effort to develop a mesh composition method, we have presented the implicit Dirichlet and Neuman boundary transmission conditions via virtual elements. This strategy avoids the iterative loop, typical in the literature to impose the transmission conditions. The Neumann condition has been implemented using schemes of first and second order of accuracy. This strategy was integrated in a computational mechanics code, Alya, at the expense of significant modifications in the basic structure of the code. This transmission conditions were equation dependent so each of the modules in the Alya code corresponding to each physic had to contain the corresponding terms. As far as we know, this implicit strategy to impose the transmission conditions is original. However, the Neuman condition did not gave satisfactory results in general. Therefore we proposed to introduce additional terms to this condition, coming from the variational multiscale context.

The next step was the unsuccessful stabilization of the Neumann boundary conditions. The idea was to analize if the boundary term of the subscale would contain the information of the neighboring elements belonging to the adjacent subdomains. A simple one-dimensional example in different regimes showed that the idea did not help us to obtain better solution of the coupled problem.

After this attempt, we took step forward and we developed a new coupling mesh method: HERMESH method, the main result of this thesis. The study is presented in one, two and three dimensions. The method consists of a *nonconforming overlapping finite element method*. The coupling is based on the construction of new elements which are the responsible to add new connectivities between the nodes of the interfaces with the ones of the neighboring subdomains. Although no theoretical analysis of the method has been developed, we have proved through different cases the following main properties:

1. Implicit: the method assembles the transmission conditions implicitly.
2. Equation independent: HERMESH does not depends on the equation to be solved.
3. Relative mesh position autonomy: independent meshes can be composed with or without overlap (partial or total) and even with a gap between them.
4. Flexible to the relative size and mesh simplification: the ratio between the sizes of the elements forming the independent meshes can be dissimilar. HERMESH is valid as a mesh simplification technique.
5. Parallel: Adapted to a high performance parallel code, Alya.
6. Accuracy: HERMESH is of order 2 for linear elements and the solution is exact.

7. HERMESH=1-domain: If the coupled meshes plus its extension elements form the same mesh as a one-domain mesh, what we have refer to as *mesh matching* in this thesis, the solution is exactly the same as the solution in one domain.

Implicit. The first item is one of the main requirements of this work. We want to avoid the iterative loop between subdomains. Thanks to the extension elements presented in chapter 3 and the way that we have implemented the method, we have achieved our objective.

Equation independent. This property has been reflected during all this thesis. The HERMESH method has been applied to different tests governed by different PDE's . The problem consisting in the simulation of ship hull presented in section 6.1 involves to solve the RANS equations, Spallart-Almaras turbulence model and level set equations. The problem of the optimization of the wind farm also involve RANS equations and another turbulence model: k-ε. The solid mechanics equation is also solved with the neuron test problem presented in section 6.2. The ADR equation also has been tested in sections 3.7 and 4.4.1. All of these problems prove the versatility of the technique.

Relative mesh position autonomist. The autonomy to the relative position of the meshes to be joined is shown also during all the problems presented in this thesis. We have presented the problems mentioned before as an example of Chimera-type problems. We present a detailed description of this kind of problems in chapter 4. The jargon used in this context as well as implementation issues are explained in this chapter. In order to simplify the construction of the extension elements in the hole created in this kind of problems, we shown the strategy to obtain a manifold boundary mesh. In addition, we have proved that we are able to *gluing meshes* in chapter 5. Two real cases reflected the potential of the strategy in chapter 6. One corresponds to the simulation of the small and large airways down to the trachea, presented in section 6.4. Both components of the computational domain have been generated in different centers. As the shape interface meshes differ significantly, a strategy was developed to fit the shape of the interfaces . The other example presented in section 6.5 corresponds to the simulation of a bypass in a stenosed artery. In this case the by-pass is designed with a certain overlap with the vessel so that a hole-cutting step is necessary.

Flexible to the relative size and mesh simplification. This is illustrated in the wind farm simulation in section 6.3, where we show that the HERMESH

method is useful as a mesh refinement technique. In fact, all of the real cases solved and presented in this thesis which present a complex geometry, also indicate that the method is a powerful strategy for mesh simplification purpose.

Parallel. It is reflected in all the real problems solved in the previous chapter. All of them have been solved in parallel and involve a large number of elements. The strategy has been designed in such a way that it does not imply significant modifications in the code where it has been programmed, Alya code. The method does not add more degrees of freedoms in the system of equations.

Accuracy. This property is related to the convergence of the method. We have shown in 5.2.1 that if the solution lives in the finite element space, the error is zero. As far as the mesh convergence property is concerned, order two with linear elements, is proved for different problems presented in sections 4.4.2 and 5.2.3. In the context of Navier-Stokes equations, we showed the consistency of mass conservation with the finite element accuracy.

HERMESH=1-domain. The last point we want to stress, called in this work *matching overlap* which shows when HERMESH is equal to one-domain problem, is proved in 5.2.2.

A lot of work remains to be done in order to make the presented method a more powerful tool in computational mechanics. As we have noted, non-repeated extension elements also remain a still-pending improvement, saving memory and time resources (Section 3.6.5). HERMESH has been programmed in a Serial way. The Parallel version of the method constitutes the next step to be taken. This is not an easy task in the context of distributed memory machines due to the strong dependence among the different meshes on the construction of the extension elements process. In fact, in a distributed memory context, candidates could be located in any subdomain of the partition or CPU. Construction the extension elements would therefore involve lots of communications between different CPU's. In addition, each partition should be able to reallocate all its arrays, as the extension elements imply new connectivity in the mesh.
We also have to consider the possibility of non-fixed components to be joined. This case implies to repeat the previous communications in each time step, so the parallel strategy has to be implemented in the most efficient manner. The relative motion of the components is not available for the time being as the method is integrated in Alya code as a preprocess step.

Bibliography

ABAQUS (2014). Unified FEA.
URL http://imechanica.org/files/l7-connections.pdf

Achdou, Y. (2002). The mortar element method with overlapping subdomains. *SIAM J. Numer. Anal.*, *40*(2), 601–628.

Achdou, Y., Tallec, P., Nataf, F., & Vidrascu, M. (2000). A domain decomposition preconditioner for an advection-diffusion problem. *Comp. Meth. Appl. Mech. Eng.*, *184*, 145–170.

Ahusborde, E., & Glockner, S. (2010). An implicit method for the Navier–Stokes equations on overlapping block-structured grids. *Int. J. Num. Meth. Fluids*, *62*(7), 784–801.

Altair (2014). Hyperworks.
URL http://www.altairhyperworks.com

Aminpour, M., Ransom, J., & McCleary, S. L. (1995). A coupled analysis method for structures with independently modelled finite element subdomains. *Int. J. Num. Meth. Eng.*, *38*(21), 3695–3718.

Attaway, S., Heinstein, M., & Swegle, J. (1994). Coupling of smooth particle hydrodynamics with the finite element method. *Nuclear engineering and design*, *150*(2), 199–205.

Axelsson, O. (2003). A survey of algebraic multilevel iteration (AMLI) methods. *BIT Numerical Mathematics*, *43*(5), 863–879.

Badia, S., & Baiges, J. (2013). Adaptive finite element simulation of incompressible flows by hybrid continuous-discontinuous galerkin formulations. *SIAM J. Sci. Comput.*, *35*(1), A491–A516.

Bank, R. E., & Dupont, T. F. (1980). Analysis of a two-level scheme for solving finite element equations. Tech. rep., Center for Numerical Analysis, University of Texas at Austin.

Barszcz, E., Weeratunga, S., & Meakin, R. (1993). Dynamic overset grid communication on distributed memory parallel processors. *AIAA Paper*, (p. 3311).

Baruzzi, G., Habashi, W., & Hafez, M. (1992). A second order accurate finite element method for the solutions of the Euler and Navier-Stokes equations. In *Proceedings of the 13*th *International Conference on Numerical Methods in Fluid Dynamics*, (pp. 509–513). Rome (Italy): Springer-Verlag.

Batchelor, G. (1970). *An Introduction to Fluid Dynamics*. Cambridge University Press.

Becker, R., Hansbo, P., & Stenberg, R. (2003). A finite element method for domain decomposition with non-matching grids. *ESAIM: Mathematical Modelling and Numerical Analysis*, *37*(02), 209–225.

Behr, M., & Tezduyar, T. (1999). The shear-slip mesh update method. *Comp. Meth. Appl. Mech. Eng.*, *174*, 261–274.

Belgacem, F. (1999). The mortar finite element method with lagrange multipliers. *Numerische Mathematik*, *84*(2), 173–197.

Belgacem, F., & Maday, Y. (1994). A spectral element methodology tuned to parallel implementations. *Comp. Meth. Appl. Mech. Eng.*, *116*(1), 59–67.

Belgacem, F., & Maday, Y. (1997). The mortar element method for three dimensional finite elements. *R.A.I.R.O. Modél. Math. Anal. Numér.*, *31*(2), 289–302.

Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., & Krysl, P. (1996). Meshless methods: an overview and recent developments. *Comp. Meth. Appl. Mech. Eng.*, *139*(1), 3–47.

Belytschko, T., Moran, B., & Liu, W. (1999). *Non–linear finite element analysis for continua and structures*, vol. 1. Wiley.

Belytschko, T., Organ, D., & Krongauz, Y. (1995). A coupled finite element-element-free galerkin method. *Computational Mechanics*, *17*(3), 186–195.

Benek, J., Steger, J., Dougherty, F., & Buning, P. (1986). Chimera. A Grid-Embedding Technique. Tech. rep., DTIC Document.

Benzi, M. (2002). Preconditioning techniques for large linear systems: a survey. *J. Comput. Phys.*, *182*(2), 418–477.

Berger, M., & Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, *82*(1), 64–84.

Bernardi, C., & Maday, Y. (1990). Coupling finite element and spectral methods: first results. *Mathematics of computation*, *54*(189), 21–39.

Bernardi, C., & Maday, Y. (2000). Mesh adaptivity in finite elements using the mortar method. *Revue européenne des éléments finis*, *9*(4), 451–465.

Bernardi, C., Maday, Y., & Rapetti, F. (2005). Basics and some applications of the mortar element method. *GAMM-Mitt.*, *28*(2), 97–123.

Berselli, L. C., & Saleri, F. (2000). New substructuring domain decomposition methods for advection–diffusion equations. *J. Comput. Appl. Math.*, *116*(2), 201–220.

Bjorstad, P., & Widlund, O. (1986). Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. Numer. Anal.*, (pp. 1097–1120).

Blacker, T., Bohnhoff, W., & Edwards, T. (1994). Cubit mesh generation environment. volume 1: Users manual. Tech. rep., Sandia National Labs., Albuquerque, NM (United States).

Blades, E. L., & Marcum, D. (2007). A sliding interface method for unsteady unstructured flow simulations. *Int. J. Num. Meth. Fluids*, *53*(3), 507–529.

Boer, A. D., Zuijlen, A. V., & Bijl, H. (2007). Review of coupling methods for non-matching meshes. *Comp. Meth. Appl. Mech. Eng.*, *196*(8), 1515–1525.

Boillat, E. (2003). Finite element methods on non-conforming grids by penalizing the matching constraint. *ESAIM: Mathematical Modelling and Numerical Analysis*, *37*(02), 357–372.

Bourgat, J., Glowinski, R., Tallec, P. L., Vidrascu, M., et al. (1988). Variational formulation and algorithm for trace operation in domain decomposition calculations. Tech. Rep. 804, Inria–Rocquencourt.

Bramble, J., Pasciak, J. E., & Schatz, A. (1986). The construction of preconditioners for elliptic problems by substructuring. I. *Math. Comput.*, *47*(175), 103–134.

Brezzi, F., Franca, L., Hughes, T., & Russo, A. (1997a). $b = \int g$. *Comp. Meth. Appl. Mech. Eng.*, *145*, 329–339.

Brezzi, F., Franca, L. P., Marini, D., & Russo, A. (1997b). *Stabilization techniques for domain decomposition methods with non-matching grids*. University of Colorado at Denver, Center for Computational Mathematics.

Brezzi, F., Lions, J., & Pironneau, O. (2001). Analysis of a chimera method. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, *332*(7), 655–660.

Brown, D., Henshaw, W., & Quinlan, D. (1999). Overture: Object-oriented tools for overset grid applications. *AIAA paper*, (p. 9130).

Cai, J., Tsai, H., & Liu, F. (2006). A parallel viscous flow solver on multi-block overset grids. *Computers & fluids*, *35*(10), 1290–1301.

Cai, X. (1993). An optimal two-level overlapping domain decomposition method for elliptic problems in two and three dimensions. *J. Sci. Comput.*, *14*, 239–247.

Cai, X. (1994). Multiplicative schwarz methods for parabolic problems. *SIAM J. Sci. Comput.*, *15*(3), 587–603.

Cai, X., Dryja, M., & Sarkis, M. (1999). Overlapping nonmatching grid mortar element methods for elliptic problems. *SIAM J. Numer. Anal.*, *36*(2), 581–606.

Cai, X.-C., & Sarkis, M. (1999). A restricted additive schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, *21*(2), 792–797.

Carlenzoli, C., & Quarteroni, A. (1995). Adaptive domain decomposition methods for advection-diffusion problems. *IMA Volumes in Mathematics and its Applications*, *75*.

Cebral, J., Löhner, R., Choyke, P., & Yim, P. (2001). Merging of intersecting triangulations for finite element modeling. *Journal of biomechanics*, *34*(6), 815–819.

Chan, T. (1987). Analysis of preconditioners for domain decomposition. *SIAM J. Numer. Anal.*, *24*(2), 382–390.

Chan, W. (2005). Advances in software tools for pre-processing and post-processing of overset grid computations. In *Proceedings of the 9th International Conference on Numerical Grid Generation and Field Simulation*.

Chan, W. (2009). Overset grid technology development at NASA Ames Research Center. *Computers & Fluids*, *38*(3), 496–503.

Chan, W., & Buning, P. G. (1996). *User's Manual for FOMOCO Utilities-Force and Moment Computation Tools for Overset Grids*. National Aeronautics and Space Administration, Ames Research Center.

Chen, Z., Tristano, J., & Kwok, W. (2004). Construction of an objective function for optimization-based smoothing. *Engineering with Computers*, *20*(3), 184–192.

Chesshire, G., & Henshaw, W. (1990). Composite overlapping meshes for the solution of partial differential equations. *J. Comput. Phys.*, *90*(1), 1–64.

Cho, Y., & Im, S. (2006). MLS–based variable–node elements compatible with quadratic interpolation. Part I: formulation and application for non-matching meshes. *Int. J. Num. Meth. Eng.*, *65*(4), 494–516.

Cho, Y., Jun, S., Im, S., & Kim, H. (2005). An improved interface element with variable nodes for non-matching finite element meshes. *Comp. Meth. Appl. Mech. Eng.*, *194*(27), 3022–3046.

Clark, B., Hanks, B., & Ernst, C. (2008). Conformal assembly meshing with tolerant imprinting. In *Proceedings of the 17th International Meshing Roundtable*, (pp. 267–280). Springer.

Cleijne, J. (1992). Results of sexbierum wind far. *Report 92-388*, *1*.

Codina, R. (1998). Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Comp. Meth. Appl. Mech. Eng.*, *156*, 185–210.

Codina, R. (2000). Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods. *Comp. Meth. Appl. Mech. Eng.*, *190*, 1579–1599.

Codina, R. (2001). A stabilized finite element method for generalized stationary incompressible flows. *Comp. Meth. Appl. Mech. Eng.*, *190*, 2681–2706.

Dey, S., Shephard, M., & Georges, M. (1997). Elimination of the adverse effects of small model features by the local modification of automatically generated meshes. *Engineering with Computers*, *13*(3), 134–152.

Dhia, H., & Rateau, G. (2005). The Arlequin method as a flexible engineering design tool. *Int. J. Num. Meth. Eng.*, *62*(11), 1442–1462.

Dihn, Q., Glowinski, R., & Périaux, J. (1984). Solving elliptic problems by domain decomposition methods with applications. *Elliptic Problem Solvers II*, (pp. 395–426).

Djomehri, M., & Biswas, R. (2003). Performance enhancement strategies for multi-block overset grid CFD applications. *Parallel Computing*, *29*(11), 1791–1810.

Dohrmann, C., Key, S., & Heinstein, M. (2000). Methods for connecting dissimilar three-dimensional finite element meshes. *Int. J. Num. Meth. Eng.*, *47*(5), 1057–1080.

Dolbow, J., & Belytschko, T. (1998). An introduction to programming the meshless Element Free Galerkin method. *Archives of Computational Methods in Engineering*, *5*(3), 207–241.

Dompierre, J., Labbé, P., Guibault, F., & Camarero, R. (1998). Proposal of benchmarks for 3D unstructured Tetrahedral Mesh Optimization. Tech. rep., Centre de Recherche en Calcul Appliqué.

Doorly, D., Taylor, D., Gambaruto, A., Schroter, R., & Tolley, N. (2008). Nasal architecture: form and flow. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1879), 3225–3246.

Dorr, M. R. (1989). On the discretization of interdomain coupling in elliptic boundary-value problems. *Domain decomposition methods*, (pp. 17–37).

Douglas, J., & Wang, J. (1989). An Absolutely Stabilized Finite Element Method. *Mathematics of Computation*, *52*, 495–508.

Dryja, M., Sarkis, M. V., & Widlund, O. (1996). Multilevel schwarz methods for elliptic problems with discontinuous coefficients in three dimensions. *Numerische Mathematik*, *72*(3), 313–348.

Dryja, M., & Widlund, O. (1994). Domain decomposition algorithms with small overlap. *SIAM J. Sci. Comput.*, *15*, 604–620.

Duarte, C., Liszka, T., & Tworzydlo, W. (2007). Clustered generalized finite element methods for mesh unrefinement, non-matching and invalid meshes. *Int. J. Num. Meth. Eng.*, *69*(11), 2409–2440.

Dureisseix, D. (2008). Two examples of partitioning approaches for multiscale and multiphysics coupled problems. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, *17*(5-7), 807–818.

Dureisseix, D., & Bavestrello, H. (2006). Information transfer between incompatible finite element meshes: application to coupled thermo-viscoelasticity. *Comp. Meth. Appl. Mech. Eng.*, *195*(44), 6523–6541.

Eguzkitza, B., Houzeaux, G., Aubry, R., & Vázquez, M. (2013a). A parallel coupling strategy for the Chimera and Domain Decomposition methods in Computational Mechanics. *Computers & Fluids*, *80*, 128–141.

Eguzkitza, B., Houzeaux, G., Calmet, H., Vázquez, M., Soni, B., Aliabadi, S., Bates, A., & Doorly, D. (2013b). A Gluing Method for Non–Matching Meshes. *Computers & Fluids*. Submitted.

Fan, J., & Chao, Y. (2010). Enhancement and application of overset grid assembly. *Chinese Journal of Aeronautics*, *23*(6), 631–638.

Fard, A., Hulsen, M., Meijer, H., Famili, N., & Anderson, P. (2012). Adaptive non-conformal mesh refinement and extended finite element method for viscous flow inside complex moving geometries. *Int. J. Num. Meth. Fluids*, *68*(8), 1031–1052.

Farhat, C., Lesoinne, M., & Tallec, P. L. (1998). Load and motion transfer algorithms for fluid/structure interaction problems with non–matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity. *Comp. Meth. Appl. Mech. Eng.*, *157*(1), 95–114.

Farhat, C., & Roux, F. (1991). A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Num. Meth. Eng.*, *32*(6), 1205–1227.

Felippa, C. A., Park, K., & Farhat, C. (2001). Partitioned analysis of coupled mechanical systems. *Comp. Meth. Appl. Mech. Eng.*, *190*(24), 3247–3270.

Flemisch, B., Melenk, J., & Wohlmuth, B. (2005). Mortar methods with curved interfaces. *Applied numerical mathematics*, *54*(3), 339–361.

Fortin, M., & Brezzi, F. (1991). Mixed and hybrid finite element methods. *Springer Series in Computational Mathematics*, *15*.

Franca, L., & Farhat, C. (1995). Bubble functions prompt unusual stabilized finite element methods. *Comp. Meth. Appl. Mech. Eng.*, *123*, 299–308.

Franca, L., Frey, S., & Hughes, T. (1992). Stabilized Finite Element Methods: I. Application to the Advective–Diffusive Model. *Comp. Meth. Appl. Mech. Eng.*, *95*, 253–276.

Freitag, L. (1997). On combining laplacian and optimization-based mesh smoothing techniques. In *Trends in unstructured mesh generation*.

Freitag, L., & Knupp, P. (1999). Tetrahedral Element Shape Optimization via the Jacobian Determinant and Condition Number. *In Proceedings of IMR'1999*, (pp. 247–258).

Freitag, L., & Knupp, P. (2002). Tetrahedral mesh improvement via optimization of the element condition number. *Int. J. Num. Meth. Eng.*, *53*, 1377–1391.

Freitag, L., Plassmann, P., et al. (2000). Local optimization-based simplicial mesh untangling and improvement. *Int. J. Num. Meth. Eng.*, *49*(1), 109–125.

Funaro, D., Quarteroni, A., & Zanolli, P. (1988). An iterative procedure with interface relaxation for domain decomposition methods. *SIAM J. Numer. Anal.*, *25*(6), 1213–1236.

Gander, M. J. (2006). Optimized schwarz methods. *SIAM J. Numer. Anal.*, *44*(2), 699–731.

Gastaldi, F., Gastaldi, L., & Quarteroni, A. (1996). Adaptive domain decomposition methods for advection dominated equations. *EAST WEST JOURNAL OF NUMERICAL MATHEMATICS*, *4*, 165–206.

George, P. (1997). Improvements on delaunay-based three-dimensional automatic mesh generator. *Finite Elements in Analysis and Design*, *25*, 297–317.

Gerardo-Giorda, L., Nobile, F., & Vergara, C. (2010). Analysis and optimization of robin-robin partitioned procedures in fluid-structure interaction problems. *SIAM J. Numer. Anal.*, *48*(6), 2091–2116.

Ghia, U., Ghia, K., & Shin, C. (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comput. Phys.*, *48*, 387–411.

Glowinski, R., He, J., Lozinski, A., Rappaz, J., & Wagner, J. (2005). Finite element approximation of multi–scale elliptic problems using patches of elements. *Numerische Mathematik*, *101*(4), 663–687.

Golub, G. (1983). The use of pre-conditioning over irregular regions. *Proc. 6e Int. Confer. Comp. Meth. Sci. Eng.*, (pp. 3–14).

Golub, G., & Loan, C. V. (1996). *Matrix Computations*. The Johns Hopkins University Press.

Greenbaum, A. (1997). *Iterative Methods for Solving Linear Systems*. SIAM series in frontiers in Applied Mathematics.

Gropp, W., & Smith, B. (1994). Experiences with domain decomposition in three dimensions: Overlapping schwarz methods. *Contemporary Mathematics*, *157*, 323–323.

Guerrero, J. (2007). Efficient treatment of complex geometries and moving bodies using overlapping grids. In *10th ISGG conference on numerical grid generation*.

Hansbo, A., Hansbo, P., & Larson, M. (2003). A finite element method on composite grids based on nitsche's method. *ESAIM: Mathematical Modelling and Numerical Analysis*, *37*(03), 495–514.

Hansbo, P. (2005). Nitsche's method for interface problems in computational mechanics. *GAMM-Mitteilungen*, *28*(2), 183–206.

Hauke, G., & García-Olivares, A. (2001). Variational subgrid scale formulations for the advection-diffusion-reaction equation. *Comp. Meth. Appl. Mech. Eng.*, *190*, 6847–6865.

Houzeaux, G. (2003). *A geometrical Domain Decomposition method in Computational Fluid Dynamics*. Ph.D. thesis, Universitat Politècnica de Catalunya.

Houzeaux, G., Aubry, R., & Vázquez, M. (2011). Extension of fractional step techniques for incompressible flows: The preconditioned Orthomin(1) for the pressure Schur complement. *Computers & Fluids*, *44*, 297–313.

Houzeaux, G., & Codina, R. (2002). A Geometrical Domain Decomposition Method in Computational Fluid Dynamics. Tech. Rep. Monography N.70, CIMNE.

Houzeaux, G., & Codina, R. (2003a). A Chimera method based on a Dirichlet/Neumann(Robin) coupling for the Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, *192*, 3343–3377.

Houzeaux, G., & Codina, R. (2003b). An iteration-by-subdomain overlapping Dirichlet/Robin domain decomposition method for advection-diffusion problems. *J. Comput. Appl. Math.*, *158*(2), 243 – 276.

Houzeaux, G., Eguzkitza, B., Aubry, R., Owen, H., & Vázquez, M. (2013). A Chimera method for the Navier-Stokes equations. *Int. J. Num. Meth. Fluids*.

Houzeaux, G., Vázquez, M., Aubry, R., & Cela, J. (2009). A Massively Parallel Fractional Step Solver for Incompressible Flows. *J. Comput. Phys.*, *228*(17), 6316–6332.

Huerta, A., & Mendez, S. F. (2000). Enrichment and coupling of the finite element and meshless methods. *Int. J. Num. Meth. Eng.*, *50*, 507–524.

Hughes, T. (1995). Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Comp. Meth. Appl. Mech. Eng.*, *127*, 387–401.

Hughes, T. (2012). *The finite element method: linear static and dynamic finite element analysis*. DoverPublications. com.

Hughes, T., & Brooks, A. (1979). A multidimensional upwind scheme with no crosswind diffusion. In T. Hughes (Ed.) *Finite Elements Methods for Convection Dominated Flows*, (pp. 19–35). New-York (USA): ASME.

Hughes, T., Feijóo, G., Mazzei, L., & Quincy, J. (1998). The variational multiscale method – A paradigm for computational mechanics. *Comp. Meth. Appl. Mech. Eng.*, *166*, 3–24.

Hughes, T., Franca, L., & Balestra, M. (1986). A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzicondition: a stable Petrov-Galerkin formulation of the Stokes problem accomodating equal-order interpolation. *Comp. Meth. Appl. Mech. Eng.*, *59*, 85–99.

Hughes, T., Franca, L., & Hulbert, G. (1989). A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/Least-Squares method for advective-diffusive equations. *Comp. Meth. Appl. Mech. Eng.*, *73*, 173–189.

Hussein, A., & El-Shishiny, H. (2012). Modeling and simulation of micro-scale wind farms using high performance computing. *9*(02).

Jespersen, D., Pulliam, T., & P.G.Buning (1997). Recent enhancements to OVERFLOW. *AIAA paper*, (p. 644).

Johnson, C. (1987). *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press.

Kao, K., Liou, M., & Chow, C. (1994). Grid adaptation using Chimera composite overlapping meshes. *AIAA journal*, *32*(5), 942–949.

Katz, A., & Jameson, A. (2008). Edge–based meshless methods for compressible viscous flow with applications to overset grids. *AIAA paper*, *3989*.

Keeling, S. L. (1997). A Theoretical Framework for Chimera Domain Decomposition. *Advances in Flow Simulation Techniques–a Conference Dedicated to the Memory of Professor Joseph L. Steger*.

Khamayseh, A., & Kuprat, A. (2008). Deterministic point inclusion methods for computational applications with complex geometry. *Computational Science & Discovery*, *1*.

Kim, H. (2002). Interface element method (IEM) for a partitioned system with non-matching interfaces. *Comp. Meth. Appl. Mech. Eng.*, *191*(29), 3165–3194.

Kim, H. (2008). Development of three-dimensional interface elements for coupling of non-matching hexahedral meshes. *Comp. Meth. Appl. Mech. Eng.*, *197*(45), 3870–3882.

Kim, H., & Widlund, O. (2006). Two-level Schwarz algorithms with overlapping subregions for mortar finite elements. *SIAM J. Numer. Anal.*, *44*(4), 1514–1534.

Knupp, P. (2000a). Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. Part II – A framework for volume mesh optimization and the condition number of the Jacobian matrix. *Int. J. Num. Meth. Eng.*, *48*, 1165–1185.

Knupp, P. (2000b). Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. PartI – A framework for surface mesh optimization. *Int. J. Num. Meth. Eng.*, *48*, 401–420.

Krebs, G., Clénet, S., & Tsukerman, I. (2010). Overlapping Finite Elements for Arbitrary Surfaces in 3–D. *Magnetics, IEEE Transactions on*, *46*(8), 3473–3476.

Landmann, B., & Montagnac, M. (2011). A highly automated parallel Chimera method for overset grids based on the implicit hole cutting technique. *Int. J. Num. Meth. Fluids*, *66*(6), 778–804.

Lee, S., Vouvakis, M., & Lee, J. (2005). A non-overlapping domain decomposition method with non-matching grids for modeling large finite antenna arrays. *J. Comput. Phys.*, *203*(1), 1–21.

Lee, Y., & Baeder, J. (2003). Implicit hole cutting–a new approach to overset grid connectivity. *AIAA paper*, (p. 4128).

Li, S., & Liu, W. (2002). Meshfree and particle methods and their applications. *Applied Mechanics Reviews*, *55*(1), 1–34.

Lim, J., Im, S., & Cho, Y. (2007). MLS (moving least square)-based finite elements for three-dimensional nonmatching meshes and adaptive mesh refinement. *Comp. Meth. Appl. Mech. Eng.*, *196*(17), 2216–2228.

Lions, P. (1988). On the Schwarz alternating method I. *First International Symposium on Domain Decomposition Methods for Partial Differential Equations.SIAM, Philadelphia (USA)*, (pp. 1–42).

Lions, P. (1989). On the Schwarz alternating method.III: A variant for non–overlapping subdomains. *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*.

Liou, M., & Kao, K. (1994). *Progress in grid generation: From Chimera to DRAGON grids*. Citeseer.

Liu, J., & Shyy, W. (1996). Assessment of grid interface treatments for multi-block incompressible viscous flow computation. *Computers & fluids*, *25*(8), 719–740.

Liu, W., Uras, R., & Chen, Y. (1997). Enrichment of the finite element method with the reproducing kernel particle method. *Transactions of the ASME-E-Journal of Applied Mechanics*, *64*(4), 861–870.

Lo, S. (1985). A new mesh generation scheme for arbitrary planar domain. *Int. J. Num. Meth. Eng.*, *21*, 1403–1426.

Lo, S., & Wang, W. (2004). A fast robust algorithm for the intersection of triangulated surfaces. *Engineering with Computers*, *20*(1), 11–21.

Lube, G., Otto, F., & Muller, H. (1998). A non-overlapping domain decomposition method for parabolic initial-boundary value problems. *Applied Numer. Math.*, *28*(2-4), 359–369.

Mandel, J. (1993). Balancing domain decomposition. *Commun. Numer. Meth. Engng*, *9*(3), 233–241.

Maple, R., & Belk, D. (1994). Automated set up of blocked, patched, and embedded grids in the beggar flow solver. *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, (pp. 305–314).

Marini, F., & Brezzi, L. (1994). A three-field domain decomposition method. In *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition, June 15-19, 1992, Como, Italy*, vol. 157, (p. 27). American Mathematical Soc.

Marini, L., & Quarteroni, A. (1988). An iterative procedure for domain decomposition methods: A finite element approach. *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*.

Marini, L., & Quarteroni, A. (1989). A relaxation procedure for domain decomposition methods using finite elements. *Numerische Mathematik*, *55*(5), 575–598.

Massing, A., Larson, M., & Logg, A. (2013). Efficient implementation of finite element methods on non-matching and overlapping meshes in 3D. *SIAM J. Sci. Comput.*, *35*(1), C23–C47.

Meakin, R. (1991). A new method for establishing intergrid communication among systems of overset grids. *AIAA paper*, (p. 1586).

Meakin, R. (1993). Moving body overset grid methods for complete aircraft tiltrotor simulations. *AIAA paper*, (p. 3350).

Meakin, R. (1995). An efficient means of adaptive refinement within systems of overset grids. *AIAA paper*, (p. 1722).

Meakin, R., & Wissink, A. (1999). Unsteady aerodynamic simulation of static and moving bodies using scalable computers. In *Proc. 14th AIAA Computational Fluid Dynamics Conference*.

METIS (2014). Family of multilevel partitioning algorithms.
URL `http://glaros.dtc.umn.edu/gkhome/views/metis`

Nataf, F. (1995). Factorization of the Convection–Diffusion Operator and the Schwarz algorithm. *Math. Mod. Meth. Appl.Sci.*, *5*(1).

Nielsen, E., & Diskin, B. (2013). Discrete adjoint-based design for unsteady turbulent flows on dynamic overset unstructured grids. *AIAA Journal*, *51*(6), 1355–1373.

Noack, R. (2005). SUGGAR: a general capability for moving body overset grid assembly. *AIAA paper*, (p. 5117).

Noack, R. (2006). DiRTlib: A Library to Add an Overst Capability to Your Flow Solver. *AIAA paper*, (p. 5116).

Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization*. Springer.

öhner, R. L., Mut, F., Cebral, J., Aubry, R., & Houzeaux, G. (2011). Deflated preconditioned conjugate gradient solvers for the pressure-poisson equation: Extensions and improvements. *Int. J. Numer. Meth. Engn.*, *87*, 2–14.

Owen, H., Houzeaux, G., Lesage, A., Samaniego, C., & Vázquez, M. (2013). Recent ship hydrodynamics developments in the parallel two–fluid flow solver Alya. *Computers & Fluids*, *80*, 168–177.

PARAVER (2014). The flexible analysis tool.
URL `http://www.bsc.es/computer-sciences/performance-tools/paraver/general-overview`

Park, S., Jeong, B., Lee, J. G., & Shin, H. (2013). Hybrid grid generation for viscous flow analysis. *Int. J. Num. Meth. Fluids*, *71*(7), 891–909.

Parks, M., Romero, L., & Bochev, P. (2007). A novel Lagrange–multiplier based method for consistent mesh tying. *Comp. Meth. Appl. Mech. Eng.*, *196*(35), 3335–3347.

Pärt-Enander, E. (1995). *Overlapping grids and applications in gas dynamics*. Ph.D. thesis, Uppsala University.

Peers, E., Zhang, X., & Kim, J. (2010). Patched characteristic interface condition for high-order multiblock aeroacoustic computation. *AIAA Journal*, *48*(11), 2512–2522.

Piperno, S., Farhat, C., & Larrouturou, B. (1995). Partitioned procedures for the transient solution of coupled aroelastic problems. Part I: Model problem, theory and two-dimensional application. *Comp. Meth. Appl. Mech. Eng.*, *124*(1), 79–112.

Prewitt, N., Belk, D., & Shyy, W. (2000). Parallel computing of overset grids for aerodynamic problems with moving objects. *Progress in Aerospace Sciences*, *36*(2), 117–172.

Puso, M. (2004). A 3D mortar method for solid mechanics. *Int. J. Num. Meth. Eng.*, *59*(3), 315–336.

Quarteroni, A., & Valli, A. (1994). *Numerical Approximation of Partial Differential Equations*. Springer-Verlag.

Quarteroni, A., & Valli, A. (1999). *Domain decomposition methods for partial differential equations*. Oxford University Press, USA.

Rabczuk, T., Xiao, S., & Sauer, M. (2006). Coupling of mesh-free methods with finite elements: basic concepts and test results. *Commun. Numer. Meth. Engng*, *22*(10), 1031–1065.

Rapin, G., & Lube, G. (2004). A stabilized three-field formulation for advection-diffusion equations. *Computing*, *73*(2), 155–178.

Rivera, C. A., Heniche, M., Glowinski, R., & Tanguy, P. A. (2010). Parallel finite element simulations of incompressible viscous fluid flow by domain decomposition with lagrange multipliers. *J. Comput. Phys.*, *229*(13), 5123–5143.

Roache, P. (1998). *Verification and validation in computational science and engineering*. Hermosapublishers. Albuquerque, N.M.

Rogers, S., Suhs, N., & Dietz, W. (2003). Pegasus 5: an automated preprocessor for overset-grid computational fluid dynamics. *AIAA journal*, *41*(6), 1037–1045.

Sanders, J., & Puso, M. (2012). An embedded mesh method for treating overlapping finite element meshes. *Int. J. Num. Meth. Eng.*, *91*(3), 289–305.

Sanderse, B., der Pijl, S., & Koren, B. (2011). Review of computational fluid dynamics for wind turbine wake aerodynamics. *Wind Energy*, *14*(7), 799–819.

Sarkis, M., & Koobus, B. (2000). A scaled and minimum overlap restricted additive schwarz method with application to aerodynamics. *Comp. Meth. Appl. Mech. Eng.*, *184*(2), 391–400.

Schiermeier, J., Housner, J., Ransom, J., Aminpour, M., & Stroud, W. (1996). The application of interface elements to dissimilar meshes in global/local analysis. In *Proceedings of the 1996 MSC World Users' Conference*.

Shakib, F., Hughes, T., & Johan, Z. (1991). A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations. *Comp. Meth. Appl. Mech. Eng.*, *89*, 141–219.

Shapiro, L. W. (1976). A catalan triangle. *Discrete Mathematics*, *14*(1), 83 – 90.

Shewchuk, J. R. (2002). What is a good linear element? interpolation, conditioning, and quality measures. *Proceedings of the 11th International Meshing Roundtable Ithaca*, (p. 115–126).

Shih, T. (1985). A procedure to debug computer programs. *Int. J. Num. Meth. Eng.*, *21*(6), 1027–1037.

Smith, M., & Pallins, J. (1993). MEDUSA–An overset grid flow solverfor network–based parallel computer system. *AIAA paper*, (p. 3312).

Soni, B., & Aliabadi, S. (2013). Large–scale CFD simulations of airflow and particle deposition in lung airway. *Computers & Fluids*, *88*, 804–812.

Soto, O., Löhner, R., & Camelli, F. (2003). A linelet preconditioner for incompressible flow solvers. *Int. J. Num. Meth. Heat Fluid Flow*, *13*(1), 133–147.

Spalart, P. R., & Allmaras, S. (1992). A one-equation turbulence model for aerodynamic flows. *AIAA J.*, *92-0439*.

Staten, M. L., Shepherd, J. F., Ledoux, F., & Shimada, K. (2010). Hexahedral mesh matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces. *Int. J. Num. Meth. Eng.*, *82*(12), 1475–1509.

Stefanica, D., & Klawonn, A. (1999). The FETI method for mortar finite elements. In *Proceedings of 11th International Conference on Domain Decomposition Methods*, (pp. 121–129). Citeseer.

Steger, J. (1991). The Chimera method of flow simulation. *Workshop on applied CFD Univ. of Tennesse Space Institute*.

Steger, J., & Benek, F. D. J. (1983). A Chimera grid scheme. *Advances in Grid Generation*, *5*, 59–69.

Steger, J., & Benek, J. (1987). On the use of composite grid schemes in computational aerodynamics. *Comp. Meth. Appl. Mech. Eng.*, *64*, 301–320.

Stüben, K. (2001). A review of algebraic multigrid. *J. Comput. Appl. Mech.*, *128*(1), 281–309.

Tallec, P. L. (1993). Neumann–Neumann domain decomposition algorithms for solving 2D elliptic problems with nonmatching grids. *East-West J. Numer. Math*, *1*(2), 129–146.

Tang, H., Casey, J., & Sotiropoulos, F. (2003). An overset–grid method for 3D unsteady incompressible flows. *J. Comput. Phys.*, *191*(2), 567–600.

Taubin, G. (1995a). Curve and surface smoothing without shrinkage. *Proceedings of the Fifth International Conference on Computer Vision*.

Taubin, G. (1995b). A Signal Processing Approach To Fair Surface Design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, (pp. 351–358). ACM.

Tian, R., & Yagawa, G. (2007). Non-matching mesh gluing by meshless interpolation—an alternative to Lagrange multipliers. *Int. J. Num. Meth. Eng.*, *71*(4), 473–503.

Toselli, A. (2001). FETI domain decomposition methods for scalar advection–diffusion problems. *Comp. Meth. Appl. Mech. Eng.*, *190*(43), 5759–5776.

Toselli, A., & Widlund, O. (2005). *Domain decomposition methods–algorithms and theory*, vol. 34. Springer Verlag.

Trangenstein, J., & Kim, C. (2004). Operator splitting and adaptive mesh refinement for the Luo–Rudy I model. *J. Comput. Phys.*, *196*(2), 645–679.

Tsukerman, A. (1992). Overlapping finite elements for problems with movement. *Magnetics, IEEE Transactions on*, *28*(5), 2247–2249.

Wang, Z., Parthasarathy, V., & Hariharan, N. (2000). A fully automated chimera methodology for multiple moving body problems. *Int. J. Num. Meth. Fluids*, *33*(7), 919–938.

Wesseling, P., & Oosterlee, C. (2001). Geometric multigrid with applications to computational fluid dynamics. *J. Comput. Appl. Math.*, *128*(1), 311–334.

White, D., Saigal, S., & Owen, S. (2004). An overset-grid method for 3D unsteady incompressible flows. *Int. J. Num. Meth. Eng.*, *59*(14), 1839–1860.

Widlund, O. (1988). Iterative substructuring methods: Algorithms and theory for elliptic problems in the plane. In *First International Symposium on Domain Decomposition Methods for Partial Differential Equations: proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Ecole Nationale des Ponts et Chaussées, Paris, France, January 7-9, 1987*, (p. 113). Society for Industrial & Applied.

Wolf, C. (2004). *A Chimera Simulation Method and Detached Eddy Simulation for Vortex-Airfoil Interactions*. Ph.D. thesis, Georg-August-Universität Göttingen.

Zahle, F., Sørensen, N., & Johansen, J. (2009). Wind turbine rotor-tower interaction using an incompressible overset grid method. *Wind Energy*, *12*(6), 594–619.

Zheng, Y., & Liou, M. (2003). A novel approach of three-dimensional hybrid grid methodology: Part 1. Grid generation. *Comp. Meth. Appl. Mech. Eng.*, *192*, 4147–4171.

Zienkiewicz, O., & Taylor, R. (1977). *The finite element method*, vol. 3. McGraw-hill London.

Appendix A

Publications

This is a list of the works carried out

A.1 Journals

A.1.1 Articles submitted

Eguzkitza, B., Houzeaux, G., Calmet, H., Vázquez, M., Soni, B., Aliabadi, S., Bates, A. & Doorly, D. A gluing method for non-matching meshes. *Computers & Fluids* 2013. Submitted.

Casoni, E., Jerusalem, A., Samaniego, C., Eguzkitza, B., Lafortune, P., Tjahjanto, D.D., Sáez, X., Houzeaux,G. & Vázquez, M. Alya: computational solid mechanics for supercomputers. *Archives for Computational methods in Engineering*. 20014. In review.

A.1.2 Articles accepted

Houzeaux, G., Eguzkitza, B. & Vázquez, M. A variational multiscale model for the advection-diffusion-reaction equation. *Communications in Numerical Methods in Engineering*. Volumen 25. Number 7. Pages:787-809. 2009.

Eguzkitza, B., Houzeaux, G., Aubry, R., Owen, H. & Vázquez, M. A Parallel coupling strategy for the Chimera and Domain Decomposition methods in Computational Mechanics. *Computers & Fluids*. 2013.**80**:128-141.

Avila, M., Folch, A., Houzeaux, G., Eguzkitza, B., Prieto, L. & Cabezón, D. A Parallel CFD Model for Wind Farms. *Procedia Computer Science*.**18**:2157-2166. Elseveir. 2013

Houzeaux, G., Eguzkitza, B., Aubry, R., Owen, H. & Vázquez M. A Chimera method for the Navier-Stokes equations. *Intet. J. Num. Meth. Fluids*. 2014.

A.2 Articles in Conference

Eguzkitza, B., Houzeaux, G. & Vázquez, M. A variational Multiscale Model Based on the Analytical Solution of the Aproxímate Subgrid Scale Equation. *USNCCM*. San Francisco-USA. July 2007.

Eguzkitza, B., Houzeaux, G., Aubry, R. & Vázquez, M. An implicit and parallel Chimera type DD Method. *Domain Decomposition-20*. San Diego (La Jolla) - USA. February 2011.

Eguzkitza, B., Houzeaux, G., Aubry, R., Peredo, O. & Vázquez, M. An Implicit and Parallel Chimera Method. *PARCFD* . Barcelona-Spain. May 2011.

Eguzkitza, B., Houzeaux, G., Vázquez, M., Jerusalem, A., & Dharmawan Tjahjanto, D. Chimera Method for solid problems. *Domain Decomposition-21*. INRIA Rennes-Bretagne-Atlantique. June 2012.

Eguzkitza, B., Houzeaux, G. Chimera Method for computacional Mechanics.*GAMM 2013*. Novi Sad-Serbia. March 2013.

Eguzkitza, B., Houzeaux, G. A Chimera Method for the Navier-Stokes equations *Congreso de Métodos Numéricos en Ingeniería-CMN 2013*. Bilbao - Spain. June 2013.

Eguzkitza, B., Houzeaux, G. A Gluing Method for Non-matching Meshes based on HERMESH coupling. *Domain Decomposition -22*. Lugano, Switzerland. Setember 2013.

A.3 Book chapters

Houzeaux, G., Owen, H., Eguzkitza, B., Samaniego, C., de la Cruz, R., Calmet H., Vázquez, M., Ávila M. Developments in Parallel, Distributed, Grid and Cloud Computing for Engineering, volume 31 of Computational Science, Engineering and Technology Series, chapter Chapter 8: A Parallel

Incompressible Navier-Stokes Solver: Implementation Issues, pages 171-201. Saxe-Coburg Publications, B.H.V. Topping and P. Iványi (Editor) edition, 2013.

A.4 Participation in research projects

OPTIDIS: Diseño óptimo de la climatización en edificios mediante métodos de dinámica computacional de fluidos BSC-CNS, Universidad Politécnica de Catalunya, CIMNE, Universidad de Girona.Guillaume Houzeaux (BSC-CNS, ES)Financial Entity:Ministerio de Ciencia y Tecnología.Spain,2006-2010.

Optimización del diseño de parques eólicos Finalcial Entity: Iberdrola and BCS-CNS. Arnau Folch,Guillaume Houzeaux,Mariano Vázquez(BSC-CNS, ES). 2010-present.