Adaptive Mesh Simulations of Compressible Flows using Stabilized Formulations

Camilo Andrés Bayona Roa

Supervisors:

Ramon Codina
Joan Baiges

Escola Tècnica Superior d'Enginyers
de Camins, Canals i Ports de Barcelona

Universitat Politècnica de Catalunya

December 2017

Acknowledgements

Let me first thanks my supervisors, professors Ramon Codina and Joan Baiges, for all the clarifying and fruitful lessons that they have given to me. It is remarkable the number of upgrades that, undoubtedly, they have given to each topic of this thesis. Thank you for teaching me how to overcome the inherent difficulties of research.

Thanks to my family for being my constant support. To Lorena, my darling love, and to Emilia for both being part of me. Also to our extended family, for being a great support in the distance, and for all their patience and hope.

Let me also mention my friends and colleagues, which have been through each step of this work, thanks for contributing me so much in this process. In the same line, this work has also been the joint effort of so many other people that I may not mention explicitly; many thanks for all your commitment.

Finally, I would like to acknowledge the doctoral scholarship that I had received from the Colombian government – Colciencias, and the big support that the International Center for Numerical Methods in Engineering gave to this research.

My gratitude to you all.

Abstract

This thesis investigates numerical methods that approximate the solution of compressible flow equations.

The first part of the thesis is committed to studying the Variational Multi-Scale (VMS) finite element approximation of several compressible flow equations. In particular, the one-dimensional Burgers equation in the Fourier space, and the compressible Navier-Stokes equations written in both conservative and primitive variables are considered. The approximations made for the VMS formulation are extensively researched; the design of the matrix of stabilization parameters, the definition of the space where the subscales live, the inclusion of the temporal derivatives of the subscales, and the non-linear tracking of the subscales are formulated. Also, the addition of local artificial diffusion in the form of shock capturing techniques is included. The accuracy of the formulations is studied for several regimes of the compressible flow, from aeroacoustic flows at low Mach numbers to supersonic shocks.

The second part of the thesis is devoted to make the solution of the smallest fluctuating scales of the compressible flow affordable. To this end, a novel algorithm for $h-$refinement of computational physics meshes in a distributed parallel setting, together with the solution of some refinement test cases in supercomputers are presented. The definition of an explicit a-posteriori error estimator that can be used in the adaptive mesh refinement simulations of compressible flows is also developed; the proposed methodology employs the variational subscales as a local error estimate that drives the mesh refinement.

The numerical methods proposed in this thesis are capable to describe the high-frequency fluctuations of compressible flows, especially, the ones corresponding to complex aeroacoustic applications. Precisely, the direct simulation of the fricative [s] sound inside a realistic geometry of the human vocal tract is achieved at the end of the thesis.

Resumen

Esta tesis investiga métodos numéricos que aproximan la solución de las ecuaciones de flujo compresible.

La primera parte de la tesis está dedicada al estudio de la aproximación numérica del flujo compresible por medio del método multiescala variacional (VMS) en elementos finitos. En particular, se consideran la ecuación de Burgers unidimensional descrita en el espacio de Fourier y las ecuaciones de Navier-Stokes de flujo compresible escritas en variables conservativas y primitivas. Las aproximaciones hechas para plantear la formulación VMS son ampliamente investigadas; el diseño de la matriz de parámetros de estabilización, la definición del espacio donde viven las subescalas, la inclusión de las derivadas temporales de las subescalas y el seguimiento no lineal de las subescalas son particularidades de la formulación que se analizan para cada una de las ecuaciones consideradas. Además, se incluye la adición de difusión artificial local en forma de técnicas de captura de choque. La precisión de las formulaciones se estudia para varios regímenes del flujo compresible, desde flujos aeroacústicos a bajos números de Mach hasta choques supersónicos.

La segunda parte de la tesis está dedicada a hacer asequible la solución de las escalas fluctuantes más pequeñas del flujo compresible. Con este fin, se presenta un algoritmo novedoso para el refinamiento h de las mallas de física computacional usadas en computación distribuida en paralelo. Además, se demuestra la solución en superordenadores de algunos casos de prueba del refinamiento de mallas. También se desarrolla la definición de un estimador de error explícito a posteriori que se puede usar en las simulaciones adaptativas de refinamiento de malla de flujos compresibles; la metodología propuesta emplea las subescalas variacionales como una estimación de error local que induce el refinamiento de la malla.

Los métodos numéricos propuestos en esta tesis son capaces de describir las fluctuaciones de alta frecuencia de los flujos compresibles, especialmente los correspondientes a aplicaciones aeroacústicas complejas. Precisamente, la simulación directa del sonido consonántico fricativo [s] dentro de una geometría realista del tracto vocal humano se demuestra al final de la tesis.

Contents

List of Figures

Part I

Chapter 1

Introduction

1.1 Prologue

The increasing amount of computing resources available in scientific and industrial research has motivated the solution of realistic computational fluid dynamics applications. This is the case of aeroacoustics, described by the solution of the compressible Navier-Stokes equations, which spans a wide range of flow scales. But the main problem for achieving an accurate description of practical engineering applications is the complexity of the fluid mechanics involved: many authors working on compressible fluid mechanics avoid the major difficulty of the full compressible Navier-Stokes equations and work on lower complexity equations.

That is the case of analyzing the Burgers equation (described in [1], and references therein). This equation is a simplified model of flow, and in spite of its simplicity, it shares some features of the fluid dynamics governed by the Navier-Stokes equations: the one-dimensional Burgers equation retains a transient term, a diffusive term (containing second order derivatives), and a non-linear (convective) term. The proof of the existence of analytic solutions for this equation was analyzed since the end of the '50s and was achieved by Hopf [2] and Cole [3]: when the non-linear term is dominant it leads to the formation of shocks, and the solution exhibits an energy spectrum that is similar to that encountered in fluid flow turbulence (governed by the Navier-Stokes equations), containing also an inertial and a dissipation range.

On the other hand, the full compressible Navier-Stokes equations, namely the conservation of mass, momentum, and energy, together with constitutive and thermodynamical relations, constitute the physical model that completely describes the compressible fluid flow phenomena. When these equations are written in conservative variables (also known as the conservative formulation of the compressible Navier-Stokes equations), those are able to represent the widest range of flow regimes (characterized by the free stream Mach number). This ability arises from the conservative property of the convective and viscous fluxes that is defined for this system of equations (as reviewed in [4]). The conservation of fluxes allows, among other properties, the description of discontinuous solutions, being this one a crucial attribute in the physics of supersonic shocks. This property is also retained in the case of the hyperbolic Euler equations, so that, one particular question related to the Navier-Stokes equations is that of its dependence on some physical parameters like viscosity. In this sense, it has been demonstrated in [5, 6] that the solution of the Euler equations is indeed, the same solution as for the inviscid limit of the Navier-Stokes equations (only subject to the absence of boundaries).

Another particular question that arises is that of the incompressible limit. A substantial contribution to the fundamentals of this problem goes back to the second half of the twentieth century with the work of Klainerman and Majda [7, 8], and Da Veiga [9]. Other works like [10] followed that early achievement and ended up with the demonstration of the correspondence with the incompressible Navier-Stokes equations in the low Mach number limit of the compressible model. However, this result is subject to small entropy variations. But, the use of the compressible flow equations for certain real fluids that are slightly compressible, as vice versa (incompressible flow equations for slightly compressible fluids), is justified: a formulation of the compressible Navier-Stokes equations that can be suitable for the low Mach number limit (or both for compressible and incompressible flows) is also important for numerical reasons, as commented in [11].

Still, the main challenge among the scientific community has been to prove the existence of solutions of the full Navier-Stokes equations. Although some advances have been made, such as those presented in [12, 13], global weak solutions to the compressible Navier-Stokes equations remains open for most types of initial conditions (as reviewed in [14]). The alternative left is to numerically approximate the solution by means of numerical methods; the possibility to apply the numerical approximation of compressible flows onto practical engineering applications is an active research topic in the computational mechanics field.

1.2 Numerical methods

Nowadays, the numerical approximation of the fluid flow equations is the subject of the computational fluid dynamics field. Yet, the complexity of the enterprise involved in the numerical approximation of the fluid flow is increased when it is aimed to represent the flow up to the smallest scales. In that case, the effort has to be focused either on applying higher precision numerical schemes or in refining the mesh and time step sizes. The first approach is cheaper in contrast to the refinement approach, but for the majority of the numerical methods high order spatial schemes can only be applied to very simple domain configurations; in the case of complex geometries, higher order approximations restrain to Finite Volume methods and Finite Element Methods (FEM).

Among the advantages of the FEM are the capability of using unstructured meshes to discretize complex geometries, and the possibility of developing efficient high order methods in the complete domain. Nonetheless, when the Galerkin method is used to approximate the fluid equations, which possesses non-symmetric operators, an unstable behavior of the solution might appear, generated for instance, by unresolved boundary layers. These local instabilities may arise when convection is dominant, and also due to the restriction of the interpolation compatibility between the different variables of the problem (as reviewed in [15]). Stabilized numerical methods like the Petrov Galerkin Streamline Upwind (SUPG) in [16], and the methods that can be framed in the Variational Multi-Scale (VMS) concept, first introduced in [17], add a stabilization term to the Galerkin formulation and eliminate the restrictions for stability.

A different approach than FEM can be done for the integration of the transient term in the fluid flow equations. Explicit higher order schemes go in line with the accurate description of the propagation of the smallest scales in the flow, but those are closely restricted to the time step size: they are limited by a temporal stability condition. In particular, high propagation

speeds restrain the explicit time step size, and intermediate stages are needed by higher order schemes, which results in several computational efforts that can be wasted in cumbersome calculations. On the contrary, the time step size can be prescribed for implicit (higher order) schemes, but some particular method for dealing with the nonlinear character of the fluid flow equations needs to be included.

Refining the mesh and time step sizes may be another option when one aims to describe small spatial and temporal scales, yet the solution of problems in which those small scales coexist with a big computational domain leads to a discretized system of equations that contains a large number of unknowns, and therefore, attempting to resolve this linear system is computationally very expensive. Adaptive Mesh Refinement (AMR) methods deal with this issue by dynamically re-configuring an initial mesh (that may be conducted to optimize the computational effort), and consequently, this technique can be implemented to make affordable the simulation of large applications. The AMR involves two main steps: first, the decision of which elements to modify (mainly the ones contributing the most to the global solution error), and then, the adaptation of those selected elements. Among the wide variety of AMR methods those that adapt the mesh size are the so-called $h-$adaptive methods, the ones that modify the order of the interpolation given by the polynomial shape function are known as the $p-$adaptive methods, the so-called $hp-$methods in [18] combine both the aforementioned h and p methods, and the moving-grid methods are designed to distort the points of the mesh (as done in [19]).

The development of the $h-$adaptive methods may involve some difficulties, which are mainly associated with the applicability of the algorithm in high-performance computations. First, the algorithm may allow successively refinements and unrefinements for any type of element (typically used in the computational meshes for fluid dynamics applications), such as triangles, quadrilateral, hexahedra, and tetrahedra, including a combination of those (of the same dimension) in the same mesh. But more importantly: it may be applied in partitioned meshes, so that, the calculations can be achieved in distributed memory machines. It is also significant that the method admits the presence of different levels of refinement across adjacent elements. Most of the refinement algorithms, like the ones in [20–22], are subject to a restriction in the number of refinement levels across adjacent elements, to what is referred as the balancing restriction. One way to overcome the enforcement of a balancing restriction is the introduction of hanging nodes: nodes that do not share the same refinement level with the adjacent element. The hanging nodes can be used together with an algebraic constriction over its degrees of freedom so that, this allows for an arbitrarily large jump of the refinement between neighbor elements. But the hanging nodes represents a difficulty for Continuous Galerkin (CG) schemes since the discrete solution is required to be continuous across the domain, and thus, support nodes are required for the solution of hanging nodes. In the case of multiple levels of hanging nodes, this process has to be repeated enriching, even more, the algebraic constriction over the hanging nodes. Another relevant aspect of the refinement algorithms is related to the ordering of the topological information of the mesh. Hierarchical data structures containing the refinement levels (which are added or subtracted), have been the most reliable method to arrange the topological information so that, the search of neighboring elements at the inter-domain level is simplified (as in references [23–25]). The parallel algorithm of the mesh adaptivity method (working over the hierarchical data structures) is a challenging topic because refining neighboring elements of the partitioned domains must be done consistently, and the most important

criteria: work balance between subdomains must be preserved. The refinement data structures have to be designed to efficiently perform communications since it has to be possible to move the hierarchical information between the partitioned domains.

With respect to the decision of which elements to modify by the AMR, typically, this is accomplished by using a local estimate of the solution error. This estimate can be defined; *a priori*, using an interpolation error of the discrete approximation, or, *a posteriori*, using the obtained discrete solution of the problem. Even though a priori estimations exhibit some noteworthy features (like the easiness of implementation and the cheap numerical cost), these approaches can lead to rough indications of the solution error (as explained in [26]). Instead, the a posteriori error estimates have demonstrated high accuracy that, however, comes with the consequent increment of the implementation effort to make the estimate practical. A posteriori estimates comprise several methods: feature-based methods, output-based methods, and residual-based methods. Among the most used, the output-based methods have been the preferred ones, especially the adjoint-based indicators where the estimation of the discretization error is related to the calculation of two sets of problems, namely, the (primal) flow problem, and the duality of the typical discrete formulation of the flow equations. Although in references [18, 27, 28], this approach has been widely applied to hyperbolic problems in the framework of Discontinuous Galerkin (DG) methods, adjoint-based indicators have recently been applied in the context of the CG compressible flow formulations, such as in [29, 30]. The latter approach can be explained as CG methods require fewer degrees of freedom than DG for a comparable accuracy: especially in the case of three-dimensional problems, in which the amount of resources needed by the DG can be cumbersome (as demonstrated in [31] and references therein). However, since the local error indicator is constructed from the adjoint solution and the residual of the finite problem, and the exact solution of the compressible problem is unavailable, such methods try to estimate the value of the finite residual by approximating the exact solution. Even for coarse approximations of the element residual in larger subdomains, this approach results inconvenient in the case of time-dependent simulations because the numerical cost is high at each time step.

Explicit estimates overcome the difficulties associated with the previous approaches by estimating the error field without the need to solve a global problem. Some early explicit estimates relate the error to the recovery of derivatives inside each element, principally the second order derivatives of the solution, arguing that this term leads the error of the discrete solution. Likewise feature-based methods, these are able to estimate compressible flow features (like boundary layers and shocks), but it has been demonstrated in [32] that those may lead to an inaccurate description of the global error. More recently, explicit residual methods aim to measure the element-wise difference between the exact solution and the one obtained by the numerical approximation. The key point of such methods is that since the exact solution of the problem is not known, the local error is related to the internal residual of the finite solution, as demonstrated in [33] for compressible flow problems. In any case, the methods that try to estimate the error of the unknowns of the problem (like the explicit residual methods) may have different objectives than the feature-based and output-based methods.

1.3 Aeroacoustics applications

The solution of the compressible Navier-Stokes equations up to the scales of sound is referred as the Direct Numerical Simulation (DNS) of sound. In this sense, the full compressible Navier-Stokes equations have the advantage describing the propagation of waves (without the need for acoustic analogies), but, computations might be limited by computing resources. An important application of the compressible Navier-Stokes formulation is in the aeroacoustics field; this is, the sound generated by turbulent flows, and more importantly, the sound produced by turbulent flows within the human speech. How the sound is generated in the particular configuration of the human vocal tract is a question that still needs to be studied. Speech therapies, for example, require the detailed illustration of the voice production mechanisms.

Among the human speech sounds that are primarily related to the compressible flow of air passing through the vocal tract, the fricative sounds are those generated by turbulent jets that occur inside the mouth. The most important difficulty for the DNS of fricative sounds is the description of both the small fluctuating pressure scales that appear near the mouth, and the large propagation distances of the radiated sound: the acoustic sources of noise are located inside the vocal tract, whereas, the radiated sound is present up to a considerable distance. Another major difficulty for the aeroacoustic problem of human voice production is the low compressibility condition of the air flow inside the mouth. At low Mach numbers the acoustic speed is very high, and consequently, the number of operations required by the compressible solver to completely describe the sound propagation is very demanding. The stability restriction placed for explicit time marching solvers limits the maximum time step size of the method. In the case of implicit solvers, the number of operations that the linear solver has to complete is large: the discrete linear system is very ill-conditioned, and therefore, the solution convergence rate is low. And for some numerical solvers of the compressible Navier-Stokes equations (e.g. using conservative variables), there is even a total lack of accuracy of the method near the incompressible limit. Indeed, numerical solvers that actually manage to directly simulate the voice need to be accurate in the zero Mach limit. Several aeroacoustic applications have been traditionally solved by using the finite difference method (see for example [34, 35]) but precisely, the main difficulty for simulating the human voice production has been the complex three-dimensional geometry of the human morphology, which cannot be represented by those methods and which is, in part, responsible for the main characteristics of the generated sound.

1.4 Objectives and outline

The work developed in the framework of this thesis can be enclosed in the main objective of investigating a stabilized finite element formulation of the compressible Navier-Stokes equations able to represent high-frequency fluctuations of the fluid flow. In particular, it has to be capable of reproducing complex aeroacoustics applications, namely, the ones corresponding to fricative sounds of speech. In order to avoid the instabilities that may appear when using the standard Galerkin method, a two-scale approximation based on the VMS concept is employed.

The specific content of this work is divided into several topics, which are developed progressively, and that will be presented in the document as follows.

Chapter 2 opens the first part of this thesis with the VMS approximation of the one-

dimensional Burgers equation. The study of this simple problem is done in the Fourier space, so that, special attention is drawn to the description of the VMS problem in that space.

Chapter 3 is devoted to the VMS formulation of the full compressible Navier-Stokes equations written in conservative variables. The design of the VMS stabilized formulation is the main objective of the first part of this thesis and specifically, it is extensively covered in that chapter. The solution of supersonic shocks with shock capturing methods is also investigated.

Chapter 4 extends the work of the previous chapters and focuses on the solution of aeroacoustic flows at the incompressible limit. This is achieved with a VMS approximation of the compressible Navier-Stokes equations written in primitive variables.

Chapter 5 closes the first part devoted to the stabilized formulations, and describes some ideas about the conservation of physical quantities in relation to the compressible Navier-Stokes equations written in primitive variables.

The second part of this thesis is dedicated to making affordable the solution to the smallest fluctuating scales of flow. To this end, the development of an $h-$refinement algorithm and the solution of some AMR examples in supercomputers are presented in Chapter 6.

Chapter 7 involves the definition of an explicit a-posteriori error estimator that can be used in AMR simulations of the compressible Navier-Stokes equations. The proposed methodology employs the VMS framework, and specifically, the idea is to use variational subscales to estimate the error.

Chapter 8 presents the application of the numerical developments that were introduced in the previous chapters to the direct solution of the fricative [s] sound.

Finally, Chapter 9 closes the thesis with some conclusions and the summary of further possible research lines.

1.5 Research dissemination

The research work contained in each chapter is quite self-contained even if this implies the need of repeating some information. The notation is gradually introduced as it is required and may vary (slightly) from one chapter to another. This is due to the fact that each chapter of this thesis has been disseminated in the form of oral presentations in scientific conferences and congresses, and in the format of articles in peer-reviewed scientific journals, as follows:

1. Chapter 2:
   C. Bayona, J. Baiges, and R. Codina, "Variational multi-scale approximation of the one-dimensional forced Burgers equation: the role of orthogonal sub-grid scales in turbulence modeling", *International Journal for Numerical Methods in Fluids*, DOI:10.1002/fld.4420.

2. Chapter 3:
   C. Bayona, J. Baiges, and R. Codina, "Variational multi-scale finite element approximation of the compressible Navier-Stokes equations", *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 26, no. 3/4, pp. 1240–1271, 2016.

3. Chapter 4:
   C. Bayona, J. Baiges, and R. Codina, "Solution of low Mach number aeroacoustic flows

using a Variational Multi-Scale Finite Element formulation of the compressible Navier-Stokes equations written in primitive variables", *Computer Methods in Applied Mechanics and Engineering*, Submitted.

4. Chapter 6:
   J. Baiges and C. Bayona, "Refficientlib: An Efficient Load-Rebalanced Adaptive Mesh Refinement Algorithm for High-Performance Computational Physics Meshes", *SIAM Journal on Scientific Computing*, vol. 39, no. 2, pp. C65–C95, 2017.

5. Chapter 7:
   C. Bayona, R. Codina, and J. Baiges, "Variational Multiscale error estimators for the adaptive mesh refinement of compressible flow simulations", *Computer Methods in Applied Mechanics and Engineering*, Submitted.

6. Chapter 8:
   C. Bayona, A. Pont, J. Baiges, and R. Codina, "Direct numerical simulation of the fricative [s] sound production", In preparation.

1.6 Applicability

A significant part of the present work has been developed within the framework of the Extensive UNIfied-domain SimulatiON of the human voice (EUNISON) research project supported by the International Center for Numerical Methods in Engineering (CIMNE).

Chapter 2

Variational multi-scale approximation of the one-dimensional forced Burgers equation: the role of orthogonal sub-grid scales in turbulence modeling

In this chapter, a numerical approximation for the one-dimensional Burgers equation is proposed by means of the Orthogonal Sub-Grid Scales - Variational Multi-Scale (OSGS-VMS) method. We evaluate the role of the variational subscales in describing the Burgers "turbulence" phenomena. Particularly, we seek to clarify the interaction between the subscales and the resolved scales when the former are defined to be orthogonal to the finite dimensional space. Direct Numerical Simulation (DNS) is used to evaluate the resulting OSGS-VMS energy spectra. The comparison against a Large Eddy Simulation (LES) model is presented for numerical discretizations in which the grid is not capable of solving the small scales.

2.1 Introduction

The Burgers equation is a simplified model of the equations that govern fluid flow, and in spite of its simplicity, it shares some features of the nonlinear dynamics present in the Navier-Stokes equations. Burgers [36] attempted unsuccessfully to arrive at a statistical theory of turbulent fluid motion, but he was able to clarify the interaction between transient, dissipative, and nonlinear terms in an extremely simplified equation of motion. His contribution still represents a significant achievement when one aims to describe turbulence: Burgers turbulence exhibits an energy cascade that is similar to that encountered in fluid flow turbulence governed by the Navier-Stokes equations, containing also an inertial and a dissipation range.

Analytic solutions are known for the forced Burgers equation: when the nonlinear term is dominant it leads to the formation of shocks, and the solution exhibits an energy spectrum that behaves as k^{-2} in the inertial range, k being the wavenumber. A complete review of the analytical solutions by Hopf [2] and Cole [3], including some of the mathematical and computational framework for the viscous form of the time-dependent Burgers equation, has been reported in [37]. Nevertheless, the important difference between the Navier-Stokes and

the Burgers dynamics is that the smallest scales that dissipate the energy of the flow in the Burgers equation do not refer to the smallest eddies, but to the shock thickness instead.

Numerically resolving every scale of the solution is what is referred as the Direct Numerical Simulation (DNS). Most of the numerical approaches that aim to correctly describe turbulent flows can be prohibitively expensive for nearly all systems with complex geometries since the discrete meshes of the approximation (or the number of basis functions in the approximation space) need to be refined up to a power of the Reynolds number. Hence, it is presently impossible to simulate flows at high Reynolds numbers using DNS, and in most of the practical fluid dynamics applications, turbulence phenomena are modeled somehow.

Several alternatives have raised in the turbulence modeling community, being Large Eddy Simulation (LES) one of the most robust and prevailing approaches. The basic idea of this method is to filter the fluid equations in space and time, simulating only the filtered large scales, while the smallest (and most expensive to compute) scales are incorporated into the overall solution by including a modeling term in the filtered equations. The filtered scales are called the resolved scales of the flow, the scales below the resolved scales are called the subscales. However, filtering the equations results in an unclosed term involving the subscales that cannot be determined from the resolved quantities, so that it needs to be calculated by assuming some *a priori* properties of the flow. In particular, eddy viscosity type models, such as the Smagorinsky model [38] and the Germano model [39], apply proportionality constants that must be set empirically *a priori*, and this makes them incomplete models. Some other authors, such as [40, 41], approximated the problem by decomposing the resolved and unresolved scales into deterministic and stochastic components, in which the subscales are calculated by using a stochastic estimation. And yet, as reviewed by [42], all these models fail to accurately represent inhomogeneous flows.

Another method that separates the solution into resolved and unresolved scales is the Variational Multi-Scale (VMS) method [17]. Alternatively to the LES approach, this method splits the solution in order to stabilize the numerical approximation of the problem. Fluid flow equations have been traditionally stabilized by using the family of VMS methods, including the Algebraic Sub-Grid Scales (ASGS) and the Orthogonal Sub-Grid Scales (OSGS) methods (like in [43]). The original idea of using the multiscale formulation with a local approximation to the fine scales to compute turbulent flows was introduced in [44], and later discussed in [45]. The illustration that the VMS formulation works as a turbulent model, that depends on the validity of the approximation made to derive the evolution equation for the unresolved scales, was elaborated in [46, 47]. Some other authors proposed to further split the resolved scales into coarse and fine scales, and to adopt the hypothesis that the subscales do not affect the coarse resolved scales. Among these methods, we can distinguish [48, 49], in which the effect of the unresolved scales is introduced only into the fine resolved scales. More recently, [50–54] have directly applied the VMS method in order to solve the Navier-Stokes turbulence, in what is called Implicit LES (ILES) techniques. This approach accurately solves turbulence relying on the addition of dissipative numerical terms solely, and without any modification of the continuous problem.

In this chapter, we aim to apply the OSGS-VMS method to the numerical approximation of the one-dimensional forced Burgers equation. In the spirit of ILES methods, we aim to clarify the mechanisms through which the resolved and unresolved scales interact with each other, and remark about the role of the orthogonal sub-grid scales in modeling Burgers turbu-

lence. The proposed numerical approximation exploits the Fourier transform of the Burgers equation, following the work in [55], in which the scale dependence on the numerical dissipation introduced by the subscales is clarified. Complementary to that approach, we include the orthogonal sub-grid scales into the resolved scales equation by means of the adjoint of the nonlinear operator applied to the test function, so that both terms associated with the Cross and Reynolds stresses are accounted for. Solving the subscales in a separated equation (subscales equation) leads to the inclusion of a Leonard stress term. In addition, we propose to calculate subscales in terms of the resolved scales by defining *a priori* the space where the subscales exist (as the orthogonal space to the finite-dimensional resolved space), and approximating the nonlinear operator associated to the Burgers problem. The scale dependence of the introduced numerical dissipation terms is studied by considering the triadic interactions among the resolved scales and the subscales. More precisely, terms associated with the Leonard, Cross, and Reynolds stresses (involving resolved and unresolved scales) allow forward and backward scattering. Contrary to [56], we are not interested in transient turbulent solutions. Instead, we test the OSGS-VMS method in contrast to the static Smagorinksy LES model and present some conclusions about the behavior of the orthogonal subscales in the numerical approximation of the Burgers turbulence phenomena.

The outline of this chapter is organized as follows. In Section 2.2 we define the Burgers equation and the VMS form of the problem. The numerical approximation used to solve the problem is presented in Section 2.3. Section 2.4 shows the numerical results for a test example. OSGS-VMS results are compared both to DNS and LES simulations at the end of that section. Finally, conclusions are stated in Section 2.5.

2.2 Problem definition

In this section, we define the VMS approximation of the Burgers problem. First, we recall the one-dimensional forced Burgers equation in the physical space, and derive the Galerkin form of the problem. Then, we apply the VMS framework and present the equations for both the resolved scales and the subscales.

2.2.1 Burgers equation

The one-dimensional forced Burgers equation [1] consists of finding a scalar function $u(x,t)$ of position $x \in \Omega = (0, 2\pi)$ and of time $t \geq 0$ such that, given a forcing term $f(x,t)$,

$$\partial_t u + u\partial_x u - \nu\partial_{xx} u = f, \qquad x \in (0, 2\pi), \;\; t > 0, \tag{2.1}$$

with an initial condition $u(x,0) = u^0(x)$ in Ω and periodic boundary conditions at $x = 0$ and $x = 2\pi$, $t \geq 0$. The diffusivity coefficient ν is considered as homogeneous and constant.

The Burgers equation shares some of the properties of the Navier-Stokes equations. Apart from the temporal derivative, the left-hand-side (LHS) contains both a nonlinear and a linear term, associated with convection and diffusion, respectively. The interaction between the dissipation given by the diffusive term (containing second order derivatives) and the convection given by the nonlinear inertial term also appears in the incompressible Navier-Stokes equations.

For convenience Eq. (2.1) can be written in the form

$$\partial_t u + \mathcal{L}(u;u) = f, \qquad x \in (0, 2\pi), \;\; t > 0, \tag{2.2}$$

by introducing the bilinear operator $\mathcal{L}(u;v) = u\partial_x v - \nu\partial_{xx} v$. For smooth solutions, $\mathcal{L}(u;u) = u\partial_x u - \nu\partial_{xx} u = \partial_x \left(\frac{1}{2}u^2 - \nu\partial_x u\right)$. The latter is the form that admits physically meaningful solutions when they develop discontinuities; it is known as the conservation form.

Let $\mathcal{W} = H^1_{\text{per}}(\Omega)$ be the subspace of periodic functions in $H^1(\Omega)$, and let us write $(f,g) = \int_\Omega fg$ for any two functions f and g. Introducing the form

$$A(u;w,v) := -\frac{1}{2}(u\partial_x w, v) + \nu(\partial_x w, \partial_x v), \tag{2.3}$$

the variational form of the problem can be written as: find $u : \mathbb{R}^+ \longrightarrow \mathcal{W}$ such that

$$(w, \partial_t u) + A(u;w,u) = (w,f), \quad t > 0, \tag{2.4}$$

$$(w,u) = (w,u^0), \;\; t = 0, \tag{2.5}$$

for all $w \in \mathcal{W}$.

2.2.2 Variational multi-scale method

Let us consider a finite-dimensional subspace $\mathcal{W}_N \subset \mathcal{W}$, of dimension N, which approximates $\mathcal{W}$ as $N \to \infty$. The Galerkin approximation to problem (2.4)-(2.5) can be stated as follows: find $u_N : \mathbb{R}^+ \longrightarrow \mathcal{W}_N$ such that

$$(w_N, \partial_t u_N) + A(u_N;w_N,u_N) = (w_N,f), \quad t > 0, \tag{2.6}$$

$$(w_N,u_N) = (w_N,u^0), \;\; t = 0, \tag{2.7}$$

for all $w_N \in \mathcal{W}_N$. This approximation suffers from instability problems, which vary according to the way to construct $\mathcal{W}_N$, but which are in any case due to the convective property of the nonlinear term.

The idea of the VMS method is to decompose the space of the unknown into a finite-dimensional space $\mathcal{W}_N$, and an infinite-dimensional one, $\tilde{\mathcal{W}}$, so that $\mathcal{W} = \mathcal{W}_N \oplus \tilde{\mathcal{W}}$. The unknown and the test functions are accordingly split as $u = u_N + \tilde{u}$ and $w = w_N + \tilde{w}$, respectively. We shall refer to functions in $\mathcal{W}_N$ as the *large* or *resolved* scales and to functions in $\tilde{\mathcal{W}}$ as the *subscales* or *unresolved* scales.

Equation (2.4) can be equivalently written as the system of equations

$$(w_N, \partial_t u) + A(u;w_N,u) = (w_N,f), \quad \text{for all } w_N \in \mathcal{W}_N, \;\; t > 0, \tag{2.8}$$

$$(\tilde{w}, \partial_t u) + A(u;\tilde{w},u) = (\tilde{w},f), \quad \text{for all } \tilde{w} \in \tilde{\mathcal{W}}, \;\; t > 0, \tag{2.9}$$

and likewise for the initial condition, Eq. (2.5). The second term in the LHS of Eq. (2.8) can be split as

$$A(u;w_N,u) = A(u_N;w_N,u_N) + A(u_N;w_N,\tilde{u}) + A(\tilde{u};w_N,u_N) + A(\tilde{u};w_N,\tilde{u}). \tag{2.10}$$

We may define:

$$A(u_N; w_N, u_N) \quad \text{Galerkin terms} \tag{2.11}$$
$$A(u_N; w_N, \tilde{u}) + A(\tilde{u}; w_N, u_N) \quad \text{Nonlinear Cross terms} \tag{2.12}$$
$$A(\tilde{u}; w_N, \tilde{u}) \quad \text{Nonlinear contribution of the unresolved scales} \tag{2.13}$$

If $\mathcal{W}_N$ is made of smooth functions, as it is the case considered below, it is readily seen that the equation for the unresolved scales (2.9) can be written as

$$(\tilde{w}, \partial_t \tilde{u}) + (\tilde{w}, \mathcal{L}(u; \tilde{u})) = (\tilde{w}, f - \partial_t u_N - \mathcal{L}(u; u_N)), \quad \text{for all } \tilde{w} \in \tilde{\mathcal{W}}, \ \ t > 0. \tag{2.14}$$

The objective of the VMS method is to approximate the subscales in terms of the resolved scales, in order to end up with a problem for the resolved scales alone. The key point is the approximation of $\mathcal{L}(u; \tilde{u})$ in order to make Eq. (2.14) easily solvable. We will not describe the motivation, which can be based on a Fourier analysis of the problem, but the approximation that we consider is (see [57])

$$\mathcal{L}(u; \tilde{u}) \approx \tau^{-1}(u)\tilde{u}, \tag{2.15}$$

where $\tau(u) > 0$ is a numerical parameter of the formulation, the expression of which is given later for a particular approximation. The objective of (2.15) is not to provide an accurate pointwise approximation to $\tilde{u}$, but to capture its effect on the resolved scales.

The residual of the resolved scales is defined as $R_N(u; u_N) = f - \partial_t u_N - \mathcal{L}(u; u_N)$. If approximation (2.15) is inserted in (2.14), one obtains a nonlinear problem for $\tilde{u}$. Even though it is possible to deal with this nonlinearity (see [57]), we will consider that the unresolved scales are smaller than the resolved ones, so that

$$\tau(u) \approx \tau(u_N), \quad R_N(u; u_N) \approx R_N(u_N; u_N). \tag{2.16}$$

These approximations and (2.15) allow us to write the approximate equation for the subscales as

$$\left(\tilde{w}, \partial_t \tilde{u} + \tau^{-1}(u_N)\tilde{u}\right) = (\tilde{w}, R_N(u_N; u_N)), \qquad \text{for all } \tilde{w} \in \tilde{\mathcal{W}}, \ \ t > 0. \tag{2.17}$$

Even though we have not distinguished them, the subscales solution of (2.17) are approximate, whereas those solution of (2.14) are exact.

The subscales space is generally defined in two different ways. The first, and the most common approach, is to define it as the space of residuals of the resolved scales, and therefore to consider that

$$\partial_t \tilde{u} + \tau^{-1}(u_N)\tilde{u} = R_N(u_N; u_N), \ \ t > 0.$$

This is what we call Algebraic Sub-Grid Scales (ASGS) method in finite elements. The second approach is the Orthogonal Sub-Grid Scales (OSGS) method, which defines the subscales space as the space orthogonal to the resolved scales space. If P_N is the L^2-projection onto $\mathcal{W}_N$ and $P_N^\perp = I - P_N$, I being the identity, then the equation for the unresolved scales in this case is

$$\partial_t \tilde{u} + \tau^{-1}(u_N)\tilde{u} = P_N^\perp[R_N(u_N; u_N)], \ \ t > 0.$$

In the spectral approximation described next, *we shall see that the OSGS method is in fact the natural approach*, since the basis functions are mutually L^2-orthogonal (and also H^1-orthogonal). In this case, we thus have that $\mathcal{W} = \mathcal{W}_N \oplus \mathcal{W}_N^\perp$, with $\mathcal{W}_N^\perp = \tilde{\mathcal{W}}$.

2.3 Solution method

In the previous section, we have defined the variational formulation for each scale of the Burgers problem. The two sub-problems (2.8) and (2.17) are now numerically approximated by transforming them into the Fourier space or, equivalently, by using a spectral Fourier method. Considering $\tilde{u} = 0$ would reduce the problem to be solved to (2.6)-(2.7), i.e., the Galerkin-Fourier method. It is well known that this approximation of the Burgers equation suffers from numerical instabilities. In this section we recall the importance of the numerical diffusion introduced by the VMS formulation, not only to prevent spurious oscillations in the numerical solution, but also to model the "turbulence" phenomena. Next, we derive the energy equation in Fourier space. Finally, the time integration scheme is described.

2.3.1 Discrete Burgers equation

Let us consider N even and define the discrete space where the solution is sought as

$$\mathcal{W}_N := \text{span}\{\mathrm{e}^{\mathrm{i}kx} : k \in [-N/2, N/2-1] \subset \mathbb{N}\},$$

i.e., the space spanned by N Fourier modes, k being the wavenumber. Let us write $(f, g) = \int_\Omega \overline{f} g$ for the L_2 inner product over complex-valued functions f and g, by denoting the complex conjugate with an overline. The approximate unknown can thus be expressed as

$$u_N(x,t) = \sum_{k=-N/2}^{N/2-1} \hat{u}_k(t)\mathrm{e}^{\mathrm{i}kx},$$

where $\hat{u}_k(t) := \left(u_N(x,t), \mathrm{e}^{\mathrm{i}kx}\right)$. We recall the orthogonality relation $\left(\mathrm{e}^{\mathrm{i}kx}, \mathrm{e}^{\mathrm{i}lx}\right) = 2\pi\delta_{kl}$, where δ_{kl} is the Kronecker delta. The notation $\sum_{k=-N/2}^{N/2-1} \hat{u}_k(t)\mathrm{e}^{\mathrm{i}kx} := \sum_k \hat{u}_k \mathrm{e}^{\mathrm{i}kx}$ will be used in the following.

To obtain the discrete weak form of the problem, let us test the differential equation (2.8) with the basis functions of $\mathcal{W}_N$. If $\hat{f}_l$, $l \in \mathbb{N}$, are the Fourier coefficients of the forcing term f, we have that $\left(\mathrm{e}^{\mathrm{i}kx}, \sum_l \hat{f}_l \mathrm{e}^{\mathrm{i}lx}\right) = 2\pi\hat{f}_k$. For the transient and diffusive terms we have

$$\left(\mathrm{e}^{\mathrm{i}kx}, \partial_t \sum_l \hat{u}_l \mathrm{e}^{\mathrm{i}lx}\right) = 2\pi\partial_t\hat{u}_k, \quad \left(\nu\partial_x \mathrm{e}^{\mathrm{i}kx}, \partial_x \sum_l \hat{u}_l \mathrm{e}^{\mathrm{i}lx}\right) = -2\pi\nu k^2 \hat{u}_k,$$

for all $k, l \in [-N/2, N/2-1]$ and $t > 0$. Moreover, the nonlinear Galerkin term results in

$$\left(\frac{1}{2}\sum_q \hat{u}_q \mathrm{e}^{\mathrm{i}qx}\partial_x \mathrm{e}^{\mathrm{i}kx}, \sum_l \hat{u}_l \mathrm{e}^{\mathrm{i}lx}\right) = -\pi\mathrm{i}k \sum_{q+l=k} \hat{u}_q\hat{u}_l, \tag{2.18}$$

for all $k, q, l \in [-N/2, N/2-1]$ and $t > 0$. The remaining terms of Eq. (2.10), which depend on the subscales, are developed next. Note first that the transient and diffusive terms belong to $\mathcal{W}_N$, and we may assume that f also belongs to this space without losing accuracy, since the order of the error made will be the same as that of approximating $\mathcal{W}$ by $\mathcal{W}_N$. Therefore, if

the subscales are defined to belong to the space orthogonal to the finite-dimensional resolved space, then the subscales equation can be expressed as

$$\left(\tilde{w}, \partial_t \tilde{u} + \tau^{-1}(u_N)\tilde{u}\right) = \left(\tilde{w}, -u_N \partial_x u_N + P_N(u_N \partial_x u_N)\right), \qquad \text{for all } \tilde{w} \in \tilde{\mathcal{W}}. \tag{2.19}$$

Now, the term $u_N \partial_x u_N$ can be developed as

$$u_N \partial_x u_N = \sum_q \mathrm{e}^{\mathrm{i}qx} \hat{u}_q \sum_l \mathrm{i} l \mathrm{e}^{\mathrm{i}lx} \hat{u}_l = \sum_{q,l} \mathrm{i} l \mathrm{e}^{\mathrm{i}(q+l)x} \hat{u}_q \hat{u}_l,$$

for all $q, l \in [-N/2, N/2-1]$. This term belongs to the space $\text{span}\{\mathrm{e}^{\mathrm{i}rx} : r \in [-N, N-2]\}$. The projection onto the space of resolvable scales, P_N, can be obtained by considering in the sum above only the wavenumbers q, l, that satisfy $q + l \in [-N/2, N/2-1]$. Consequently, the orthogonal subscales belong to the Fourier space $\tilde{\mathcal{W}}$, given by

$$\tilde{\mathcal{W}} := \text{span}\{\mathrm{e}^{\mathrm{i}rx} : r \in [-N, N-2] \setminus [-N/2, N/2-1]\}.$$

With this definition, taking $\tilde{w} = \mathrm{e}^{\mathrm{i}rx}$ in (2.19), we obtain the equation for the subscales

$$\partial_t \hat{\tilde{u}}_r + \tau^{-1}(u_N)\hat{\tilde{u}}_r = \sum_{q+l=r} \mathrm{i} l \hat{u}_q \hat{u}_l, \tag{2.20}$$

for all $q, l \in [-N/2, N/2-1], r \in [-N, N-2] \setminus [-N/2, N/2-1]$, and $t > 0$.

In order to close the subscales calculation, the approximation of the Burgers nonlinear operator (2.15) is needed. The key point in the design of the VMS formulation is the construction of τ. In this work we use the approximation of the nonlinear Burgers operator proposed in [55], which is defined as

$$\tau(u_N) = \left[3\pi\nu^2 \left(\frac{4}{h^2}\right)^2 + \frac{4}{h^2}\|u_N\|^2 \right]^{-\frac{1}{2}}, \tag{2.21}$$

where $h = \pi/N$ can be considered as an effective grid size and $\|u_N\|$ is the L^2-norm of u_N. This approximation basically aims to bound the effect of the nonlinear operator in Fourier space. Now we can compute the subscales in terms of the resolvable scales from (2.20) and use the resulting expression in the equation for the latter.

Remark 1. *The space of subscales is in principle of infinite dimension, and defined as the complimentary of the Fourier space of resolved scales. It can also be defined as the difference between the DNS solution and the resolved space, such as the one derived in [55]. However, with the approximations we have assumed, the space of subscales turns out to be of finite dimension, and defined in terms of the finite-dimensional resolved space. In particular, the Fourier space where the orthogonal subscales live has the same dimension as the resolved space, and a finer discretization generates a richer definition of the subscales.*

Remark 2. *We could actually solve the subscales taking into account the nonlinearity of the problem, that is to say, we could proceed without approximation (2.16). This simply amounts to replace u by $u_N + \tilde{u}$ and deal with the additional nonlinearity by using an appropriate linearization scheme.*

Remark 3. *Neglecting the time derivative in (2.20) leads to the so-called quasi-static subscales, which could be understood as the subscales given automatically by the residual. On the contrary, by defining the subscales equation as in (2.20), it must be solved as a separated* differential *equation of the problem. This corresponds to what was termed as* dynamic subscales *in [57]. Note that Eq. (2.20) is, in fact, an ordinary differential equation that needs to be integrated in time for each Fourier mode.*

Let us look now at the contribution of the subscales into the finite-dimensional equation (2.10). For conciseness, let us imply $q, k \in [-N/2, N/2-1], r \in [-N, N-2]\backslash[-N/2, N/2-1]$, and $t > 0$. Since the subscales are orthogonal to the finite-dimensional space, and the diffusive term in (2.13) belongs to the resolved space, it reduces to $\left(\nu\partial_{xx}\mathrm{e}^{\mathrm{i}kx}, \sum_r \hat{\tilde{u}}_r \mathrm{e}^{\mathrm{i}rx}\right) = 0$. The same occurs with the transient term for the subscales (2.13), $\left(\mathrm{e}^{\mathrm{i}kx}, \partial_t \sum_r \hat{\tilde{u}}_r \mathrm{e}^{\mathrm{i}rx}\right) = 0$, which is orthogonal to the resolved scales by definition. Regarding the nonlinear terms that involve the subscales in (2.10), each one is developed as follows:

$$\left(\frac{1}{2}\sum_q \hat{u}_q \mathrm{e}^{\mathrm{i}qx}\partial_x \mathrm{e}^{\mathrm{i}kx}, \sum_r \hat{\tilde{u}}_r \mathrm{e}^{\mathrm{i}rx}\right) = -\pi\mathrm{i}k \sum_{q+r=k} \hat{u}_q \hat{\tilde{u}}_r, \tag{2.22}$$

$$\left(\frac{1}{2}\sum_r \hat{\tilde{u}}_r \mathrm{e}^{\mathrm{i}rx}\partial_x \mathrm{e}^{\mathrm{i}kx}, \sum_q \hat{u}_q \mathrm{e}^{\mathrm{i}qx}\right) = -\pi\mathrm{i}k \sum_{r+q=k} \hat{\tilde{u}}_r \hat{u}_q, \tag{2.23}$$

$$\left(\frac{1}{2}\sum_r \hat{\tilde{u}}_r \mathrm{e}^{\mathrm{i}rx}\partial_x \mathrm{e}^{\mathrm{i}kx}, \sum_p \hat{\tilde{u}}_p \mathrm{e}^{\mathrm{i}px}\right) = -\pi\mathrm{i}k \sum_{r+p=k} \hat{\tilde{u}}_r \hat{\tilde{u}}_p. \tag{2.24}$$

As we can solve each Fourier wavenumber k separately, the final expression for the resolvable scales of the Burgers equation is

$$\partial_t \hat{u}_k + \frac{\mathrm{i}k}{2}\sum_{q+l=k} \hat{u}_q \hat{u}_l + \mathrm{i}k \sum_{q+r=k} \hat{u}_q \hat{\tilde{u}}_r + \frac{\mathrm{i}k}{2}\sum_{r+p=k} \hat{\tilde{u}}_r \hat{\tilde{u}}_p + \nu k^2 \hat{u}_k = \hat{f}_k, \tag{2.25}$$

for all $k, q, l \in [-N/2, N/2-1]$, $r, p \in [-N, N-2] \setminus [-N/2, N/2-1]$, and $t > 0$.

The formulation is now complete. It consists of solving (2.25) *together with* (2.20), for u_N and $\tilde{u}$.

Remark 4. *The problem has to be completed with initial conditions for (2.20). We assume that* $\hat{\tilde{u}}_r^0 = 0$*, for all* $r \in [-N, N-2]\setminus[-N/2, N/2-1]$*, which means that we assume for simplicity that the initial condition belongs to the space of resolvable scales.*

2.3.2 Discrete energy equation

We now develop the numerical approximation to the energy equation in Fourier space. The energy e_k of the k-th wavenumber is obtained by taking the product of $\hat{u}_k$ with its complex conjugate $\overline{\hat{u}_k}$ (and dividing by 2, if wished). Its time derivative is given by:

$$\partial_t e_k = \partial_t(\hat{u}_k \overline{\hat{u}_k}) = \hat{u}_k \partial_t \overline{\hat{u}_k} + \overline{\hat{u}_k}\partial_t \hat{u}_k. \tag{2.26}$$

After the right-hand-side terms of the previous equation are developed, the discrete energy equation can be written as:

$$\partial_t e_k = -\nu 2k^2 e_k - T(k) - C(k) - R(k) + \overline{\hat{u}_k}\hat{f}_k + \overline{\hat{f}_k}\hat{u}_k, \tag{2.27}$$

for all $k \in [-N/2, N/2-1]$, and $t > 0$, where

$$T(k) = \overline{\hat{u}_k}\left(\frac{\mathrm{i}k}{2}\sum_{q+l=k}\hat{u}_q\hat{u}_l\right) + \hat{u}_k\left(\overline{\frac{\mathrm{i}k}{2}\sum_{q+l=k}\hat{u}_q\hat{u}_l}\right) \tag{2.28}$$

stands for the nonlinear transfer term due to the resolved scales,

$$C(k) = \overline{\hat{u}_k}\mathrm{i}k\sum_{q+r=k}\hat{u}_q\hat{\tilde{u}}_r + \hat{u}_k\overline{\mathrm{i}k\sum_{q+r=k}\hat{u}_q\hat{\tilde{u}}_r} \tag{2.29}$$

is the transfer due to the Cross nonlinear term involving resolved scales and subscales, and

$$R(k) = \overline{\hat{u}_k}\left(\frac{\mathrm{i}k}{2}\sum_{r+p=k}\hat{\tilde{u}}_r\hat{\tilde{u}}_p\right) + \hat{u}_k\left(\overline{\frac{\mathrm{i}k}{2}\sum_{r+p=k}\hat{\tilde{u}}_r\hat{\tilde{u}}_p}\right) \tag{2.30}$$

is the transfer term due to the nonlinear relation among subscales. All previous relations are defined for $k, q, l \in [-N/2, N/2-1]$, $r, p \in [-N, N-2] \setminus [-N/2, N/2-1]$, and $t > 0$.

Remark 5. *The Galerkin nonlinear term (2.18) is responsible for the interaction among the resolved scales. The triadic interaction of this term expresses all the possible combinations of the pairs of resolved wavenumbers q and l such that $q + l = k$. In Fig. 2.1 a brief schematic depicts the triadic interactions between the resolved wavenumbers; the two groups of possible combinations $q + l = k$ include: both q and l smaller than k (at the LHS of the figure), and either one of q or l greater than k, added with the remainder conjugate less than k (at the RHS of the figure). The possible combination depicted at the LHS of the figure relates two big resolved scales with one smaller resolved scale. Instead, the combination at the RHS indicates the relation between a resolved scale with a bigger and a smaller resolved scale. It can be proved (like in [58]), that for a given time, $T(k) \in \mathbb{R}$, and that the total energy transfer due to the Galerkin nonlinear term is conserved, i.e., $\sum_k T(k) = 0$. Therefore, the nonlinear transfer term due to the resolved scales (2.28), and its respective sign, determines the way the energy is transferred between the resolved scales, in which not only large resolved wavenumbers can transfer energy forward to small resolved wavenumbers, but also the possibility of transferring backwards the energy from small to large resolved wavenumbers exists, as commented before.*

Having remarked the importance of the Galerkin nonlinear term, which rules the development of an energy cascade as it transfers energy between the resolved wavenumbers in the inertial range [58], we focus our attention on the terms arising from relations (2.20), (2.22), (2.23) and (2.24). Such terms represent the interaction between the resolved scales and the subscales, and correspond to the so-called *Leonard, Cross, and Reynolds stresses* in the standard large eddy simulation (LES) approach to solve turbulent flows. It is worthy to note that the presence of these terms is related only to the introduction of the VMS method, without the modification of the continuous problem that occurs with the LES method.

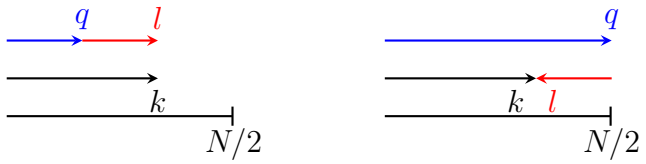


Figure 2.1: Triadic interactions among the resolved scales

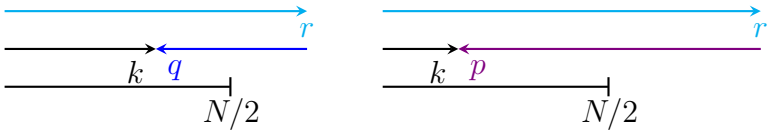


Figure 2.2: Triadic interactions among the resolved and subgrid scales: Cross stress (left), and Reynolds stress (right).

We can detail the effect of the Cross nonlinear term (2.22) and (2.23) by looking to Fig. 2.2. This figure shows how the subscales play a fundamental role in the resolved scales equation by means of this term. More precisely, the LHS of the figure depicts the possible combinations $q + r = k$ between the resolved and subgrid scales. These type of combinations increase with the resolved wavenumber k, and therefore, the term (2.29) can be seen as increasingly responsible for dissipating energy at high wavenumber resolved scales. This will be verified later in the numerical example. On the contrary, for the Reynolds stress term (2.24) possible combinations $p + r = k$ decrease with the resolved wavenumber. A depiction of that type of combinations involving the subscales is shown on the right side of Fig 2.2. Therefore, the transfer term due to the nonlinear relation among subscales (2.30) is essentially related to the dissipation of energy at small resolved wavenumbers.

As in (2.27), the discrete energy equation for the subscales can be obtained by taking the product of Eq. (2.20) with the complex conjugate of the subscale $\overline{\left(\hat{\tilde{u}}_r\right)}$. If e_r is the energy of the subscales, its time derivative is:

$$\partial_t e_r = -2\tau^{-1}(u_N)e_r + L(r), \tag{2.31}$$

for all $r \in [-N, N-2] \setminus [-N/2, N/2-1]$, and $t > 0$. Here we have arranged the nonlinear transfer term due to the residual as:

$$L(r) = \overline{\hat{\tilde{u}}_r} \left(\sum_{q+l=r} \mathrm{i} l \hat{u}_q \hat{u}_l \right) + \hat{\tilde{u}}_r \left(\overline{\sum_{q+l=r} \mathrm{i} l \hat{u}_q \hat{u}_l} \right), \tag{2.32}$$

for all $q, l \in [-N/2, N/2-1]$, $r \in [-N, N-2] \setminus [-N/2, N/2-1]$, and $t > 0$. This transfer term is related to the Leonard stress term and provides the energy that is dissipated by the subscales. An example of the possible combinations between the resolved wavenumbers in the Leonard stress term is presented in Fig. 2.3, where it can be observed that the possible combinations of the pairs of resolved wavenumbers q and l, such that $q + l$, decreases with the subscale wavenumber. As a consequence of this, the rate of energy transfer (and dissipation) of the subscales is in principle smaller for unresolved high wavenumbers.

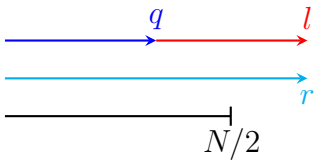


Figure 2.3: Triadic interactions among the resolved and subgrid scales for the Leonard stress.

Remark 6. *The energy transfer terms (2.29), (2.30), and (2.32) allow for the backward transfer of energy from the subscales to the resolved scales, provided the sign of those terms is negative for certain resolved and subscale wavenumbers.*

2.3.3 Time integration method

Let $0 = t^0 < t^1 < ... < t^N$ be a partition of the time interval of analysis, with $0 < \delta t = t^{n+1} - t^n$ for $n = 0, 1, 2, ...$ Here and below the superscript denotes the time step counter. The transient term integration of the evolution equation for the resolvable scales and the subscales, as well as the energy equations, can be described as follows. Let us write these equations in the form:

$$\partial_t v_k = \mathcal{F}\left(v_k\right), \tag{2.33}$$

for each k wavenumber. We use a fully explicit Runge-Kutta time-integration scheme of S stages. If v_k^n is known, the solution at t^{n+1} is given by

$$v_k^{n+1} = v_k^n + \delta t \sum_{i=1}^{S} \mathsf{b}_i K_i^n, \tag{2.34}$$

for each k wavenumber, where K_i^n (defined for $1 \leq i \leq S$) are the stage increments, obtained as

$$K_i^n = \mathcal{F}\left(v_k^n + \delta t \sum_{j=1}^{i-1} \mathsf{a}_{ij} K_j^n\right). \tag{2.35}$$

Expresions (2.34) and (2.35) depend on the definition of the coefficients a_{ij} (for $1 \leq j \leq i \leq S$) and b_i (for $1 \leq i \leq S$) for a specific explicit method. In particular, the explicit Euler scheme corresponds to $S = 1$ and $\mathsf{b}_1 = 1$. A stability condition of Courant-Friedrichs-Lewy type must be imposed in order to guarantee stability of the time integration method. The previous time marching technique applied to Eqs. (2.25), (2.20), and (2.27), defines the spatial and temporal OSGS-VMS discretization of the Burgers problem. In the following section, we apply this discretized formulation in order to simulate a numerical example.

2.4 Numerical example

In this section, we solve a numerical example that is selected to test the OSGS-VMS formulation. We first define the continuous problem, which is a steady one-dimensional viscid case.

The fact that the "turbulent" energy spectrum of the one-dimensional Burgers problem is only due to the resolution of strong gradients in the solution (shocks), and not to the unsteady solution of the problem (as in the case of the dissipative eddies in the Navier-Stokes turbulence), has prompted us to select the problem as a steady case. Then we present the DNS of the problem, which accounts for all the wavenumbers in the solution. The DNS solution is used to evaluate the accuracy of the OSGS-VMS method. Specifically, we discuss the resulting energy spectra for discretizations in which the number of modes N is not enough to resolve all of the wavenumbers present in the solution. As in (2.21), we can consider an effective grid size $h = \pi/N$ and refer to coarse and fine grids. In the spirit of ILES methods, coarse grid solutions are also compared against LES results in the last part of this section.

Under certain conditions (detailed in [59]), the periodic solution of the one-dimensional forced Burgers problem tends toward a stationary state. This is achieved in particular by selecting $\hat{u}_0^0 = 0$ and $\hat{u}_k^0 = k^{-1}$ as initial conditions, and by fixing $\partial_t \hat{u}_1 = 0$ for $t > 0$. Given these conditions, the diffusivity coefficient of the example is fixed to $\nu = 0.025$, so that the nonlinear term is dominant in the Burgers equation. The periodic solution describes a single shock that arises from the large-scale restriction over $\hat{u}_1$ and the predominance of the nonlinear term. This restriction also provides the amount of energy that is propagated through the simulation and dissipated by the center of the shock. In this sense, the smallest scales corresponding to the "Kolmogorov" scale belong to the viscous shock thickness $l \approx \nu/|u|$, and the maximum resolved wavenumber of the DNS solution is calculated such that the associated grid spacing is below this size. The DNS solution of the numerical example is carried out by completely solving all the scales, which is achieved with $N = 400$ (so that $h = 7.85 \times 10^{-3}$ is below $l \approx 10^{-2}$). We also perform simulations in which the grid is not fine enough to resolve all the wavenumbers of the solution. The comparison of the coarse grid simulations (solved with a maximum resolved wavenumber N) and the DNS solution is explained below. In order to guarantee numerical stability, the time integration stability criterion is set for all the resolved cases to a CFL number smaller than 0.01 for the linearized problem, converging to the steady state at approximately $t = 2$.

Figure 2.4 shows the energy spectrum at the stationary converged state for the DNS and coarse grid simulations. The energy spectrum at the initial simulation time has a regular distribution and behaves as k^{-2}, with the large scales containing the majority of the energy in the system. As it evolves in time, the energy spectrum develops the cascade shape with the typical slope -2 (in a log scale) behavior in the inertial range (that differs from the well-known $k^{-5/3}$ behavior of the Navier-Stokes turbulence), and a drop-off corresponding to the dissipation provided by the shock. The decay in time of the energy spectra solved by the OSGS-VMS method is accurate for all the coarse grids, giving the k^{-2} behavior at the inertial range, with only small deviations occurring near the maximum resolved wavenumber. In this sense, an energy pile up is observed for a few wavenumbers below the maximum resolved wavenumber. It is also observed that refining the grid, and consequently enriching the space of the subscales, improves the accuracy of the energy spectrum near the maximum resolved wavenumber as the dissipation of energy increases and the pileup vanishes. These results indicate that the turbulent energy spectrum is correctly approximated by the OSGS-VMS numerical method. Even for coarse discretizations that are not sufficient to solve the smallest scales of the solution, the inertial range of the energy spectrum is accurately captured.

Let us discuss now the solution to the Burgers problem in the physical space. Figure 2.5

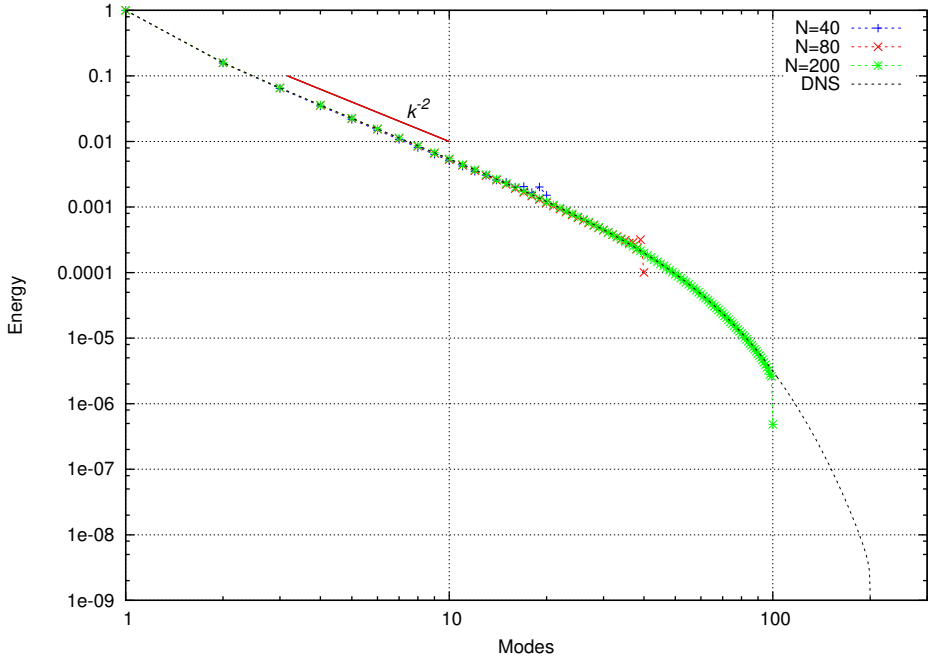


Figure 2.4: OSGS-VMS results: energy spectrum.

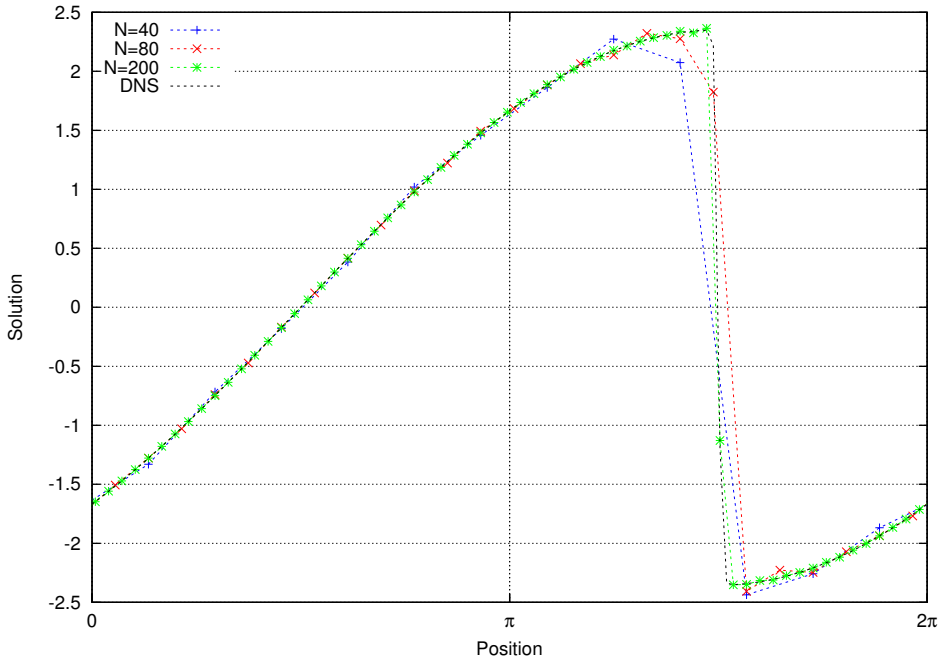


Figure 2.5: OSGS-VMS results: physical space solution.

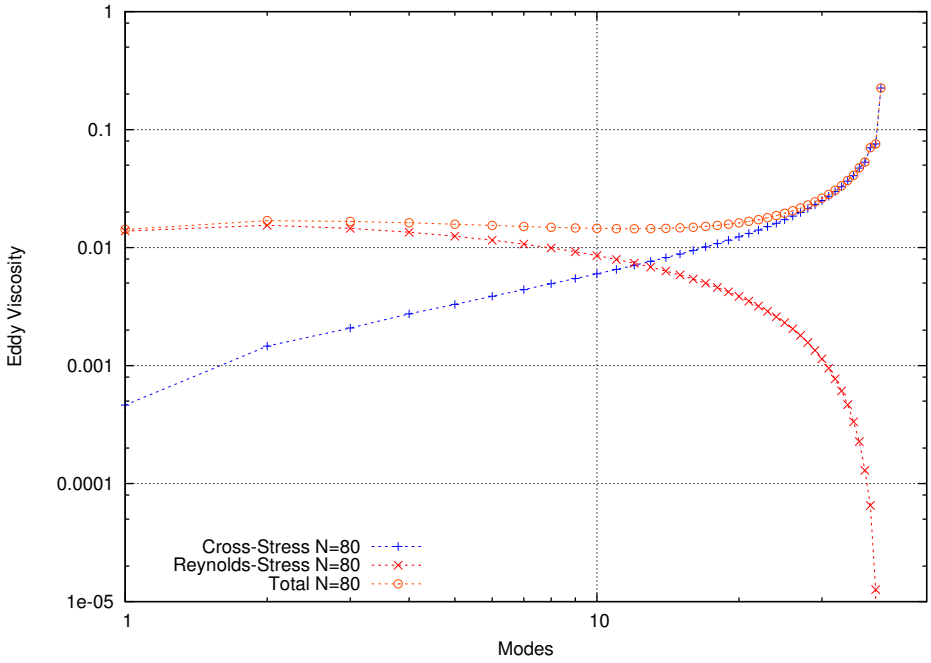


Figure 2.6: OSGS-VMS results: Eddy viscosity.

presents the physical space solution, including the coarse grid solutions and the DNS solution. The physical space solution $u_N \in \mathbb{R}$ is plotted in the range $[0, 2\pi]$ since the solution is periodic. In general, the OSGS-VMS method gives an accurate physical solution, including the periodic wave behavior and the formation of the one-dimensional shock as a step in the solution. For the coarse grid solutions, the Gibbs phenomenon is present. This can be observed in the figure as local oscillations near the shock, which are commonly eliminated through shock capturing techniques. A slight inaccuracy different from the Gibbs phenomenon is observed for the coarsest grid solutions, which corresponds to the previously described energy pile up at the highest resolved wavenumbers.

Now we center the discussion about the implicit modeling of turbulence done by the OSGS-VMS method. As in the case of the Navier-Stokes turbulence, the forced Burgers problem exhibits the energy dissipation cascade, in which the energy of the large scales is transferred to the small scales where the energy is dissipated. The triadic interaction mechanism commented in Section 2.3, corresponding to the nonlinear term involving the resolved scales, is responsible for the energy transfer from the preponderant resolved large scales to the much smaller resolved scales. The mechanisms through which the energy contained in the resolved and unresolved scales interact with each other can be clarified by analyzing the scale dependence and the dissipation driven by the nonlinear terms. For doing so, we plot the spectral eddy viscosity arising from the Cross and Reynolds stress terms, which is a scaled version of the energy dissipated by each nonlinear term, and it is calculated as $\nu_C(k) = C(k)/k^2e(k)$, and $\nu_R(k) = R(k)/k^2e(k)$, where $e(k)$ is the energy of the wavenumber. We plot the scale dependence of the nonlinear terms for a $N = 80$ case in Fig 2.6. It has been previously explained that the Cross stress contribution (involving the resolved and unresolved scales) dissipates energy

increasingly with the resolved wavenumber. On the contrary, the energy dissipation related to the Reynolds stress is greater for the largest scales of the solution. In fact, the largest scales of turbulence are only correctly solved if the nonlinear term accounting for the unresolved scales is included. Contrary to [55], where the stabilizing term was defined by a linearized asymptotic expansion over the residual, and hence only Cross stress terms appeared in the discretized equation, we find that including the Reynolds stress tensor (resulting from nonlinear terms involving the subscales) is a requirement for the proper stabilization of the problem, but also for the correct approximation of the turbulence phenomena. In this sense, we also find that including the transient term in the subscales equation (2.14), and therefore solving the subscales equation as a separate equation, makes the Leonard stress term to be explicitly defined and improves the numerical convergence to the steady state of the solution. Adding the dissipation provided by the Cross and Reynolds terms results in a plateau distribution of the spectral eddy viscosity regardless of the resolved wavenumber. The dissipation increases only at higher resolved wavenumbers by the effect of the Cross stress term. It is remarkable that the numerical dissipation given by the OSGS-VMS method removes energy at a correct rate, mainly due to the relation between the variational subscales and the finite resolved scales through the nonlinear terms, which improves by enriching the discretization.

We aim to compare now the previous results with some Burgers turbulence results obtained with a LES model. First, let us briefly comment about the modifications in the continuous Burgers equation when the LES method is applied. The LES method filters the Burgers equation in space and time, leading to the problem

$$\partial_t \underline{u} + \partial_x(\frac{1}{2}\underline{uu}) = \nu \partial_{xx}\underline{u} + \underline{f}, \quad x \in (0, 2\pi),\ t > 0. \tag{2.36}$$

We have denoted the filtered quantities by using an underline. The filtered nonlinear term $\underline{uu}$ is the main difficulty in LES: it requires knowledge of the unfiltered velocity field *a priori*, which is unknown, so it must be modeled somehow. In this sense, the filtered nonlinear term can be split up in the form $\underline{uu} = \mathcal{T} + \underline{u}\,\underline{u}$, where the quantity $\underline{u}\,\underline{u}$ is resolved. The introduced residual subfilter-scale stress tensor $\mathcal{T}$ must be taken into account in the solution of Eq. (2.36). This tensor has to be modeled in terms of the filtered unknown in order to close the filtered equation. The static Smagorinsky model $\mathcal{T} \approx -2\nu_t \mathcal{S}$ introduced in [38] is a common closure model for this problem, where $\mathcal{S} = \frac{1}{2}\partial_x \underline{u}$ is the one-dimensional strain tensor and ν_t an additional eddy viscosity. Other closures, such as the Dynamic Smagorinsky, Scale-similarity, Mixed, and Linear unified RANS-LES models, are discussed in [56]. The closure model can be replaced directly in the filtered equation, so that the modified Burgers equation is obtained:

$$\partial_t \underline{u} + \underline{u}\partial_x \underline{u} = \nu_{\text{eff}} \partial_{xx}\underline{u} + \underline{f}, \quad x \in (0, 2\pi),\ t > 0. \tag{2.37}$$

In practice, the main change with respect to the original equation is to modify ν by $\nu_{\text{eff}} = \nu + \nu_t$. In order to numerically approximate the previous equation, the Fourier-Galerkin method can applied to the previous equation. Using the same notation as before, the discretized problem is then:

$$\partial_t \hat{\underline{u}}_k + \frac{\mathrm{i}k}{2} \sum_{k=q+l} \hat{\underline{u}}_q \hat{\underline{u}}_l + \nu_{\text{eff}} k^2 \hat{\underline{u}}_k = \hat{f}_k, \tag{2.38}$$

for all $k, q, l \in [-N/2, N/2-1]$, and $t > 0$. We apply also the time marching scheme described in Section 2.3. The eddy viscosity defined in the Fourier space is modeled using a spectral eddy viscosity proposed in [60], in our case taking the cutoff wavenumber as the number of modes, N. It is given for each wavenumber k by

$$\nu_t(k) = \nu_t^{+\infty}\Big(\frac{E_N}{N}\Big)^{1/2}\nu_t^*\Big(\frac{k}{N}\Big),$$

with

$$\nu_t^{+\infty} = 0.31\frac{5-m}{m+1}\sqrt{3-m}\,C_k^{-3/2},$$

where m is the negative slope of the energy spectrum, E_N is the total energy spectrum at the cutoff wavenumber, C_k is the Kolmogorov constant, and ν_t^* is a non-dimensional eddy-viscosity, given by

$$\nu_t^*(k) = 1 + 34.5\,\mathrm{e}^{-3.03(N/k)}.$$

Since we may consider that the density is $\rho = 1$, the total energy spectrum is such that $E = \int_\Omega |\underline{\hat{u}}|^2 d\boldsymbol{k}$, and therefore, its value at the cutoff wavenumber is $E_N = e_N/N$. Numerical coefficients must be set in order to close the viscosity model, which must be known *a priori*. In particular, we need to set the Burgers energy spectrum negative slope, which has been previously shown to be $m = 2$, and the Kolmogorov constant C_k, which we set experimentally to 0.1 in order to obtain the most accurate results.

As the filtering procedure selects only a certain number of wavenumbers to be solved by the spectral method, the resolved scales are the filtered scales below the selected cutoff wavenumber N. Figure 2.7 shows several solutions in the physical space when applying the static Smagorinsky model with different grid resolutions. The figure shows a correct description of the physical solution for all simulations. Gibbs phenomenon close to the shock is also observed. In general, the method is accurate and correctly portrays the physical solution of the Burgers problem with the given numerical coefficients. Figure 2.8 presents the energy spectrum for the simulations using the static Smagorinsky model. Even though the LES model is accurate for most of the resolved scales of turbulence, the energy spectra of the DNS and the LES solutions differ substantially for some resolved wavenumbers belonging to the inertial range. In this sense, the LES spectrum presents some energy pile up near the cutoff wavenumber, which is stronger for finer grids. This is an important difference between the LES and the OSGS-VMS turbulence modeling; while applications with the former have to care for the range of scales that are correctly solved by the grid, applying the OSGS-VMS method correctly describes the turbulence phenomena no matter the grid resolution. Remarkably, refining the grid improves the accuracy of the OSGS-VMS energy spectrum near the maximum resolved wavenumber, contrary to the decrease in the energy dissipation observed in LES. Numerical experiments indicate that increasing the Kolmogorov constant value improves the accuracy of the energy spectrum in the inertial range, but produces a more pronounced energy pile up at the cutoff wavenumber. These under dissipative solutions behave similarly to the Fourier-Galerkin method presented in Fig. 2.9 for a $N = 40$ simulation, which gives globally unstable solutions. In that figure, we show the energy spectrum results for Galerkin, OSGS-VMS and LES methods. On the contrary, the LES energy spectrum results are not accurate when the Kolmogorov constant is decreased. In this sense, the LES spectrum falls faster than the DNS one,

and even some energy pile up near the cutoff wavenumber occurs. The possibility of obtaining an over-dissipative character is well known for the LES models: even though the large scales of turbulence may be correctly solved, the energy spectra of the LES solution can differ substantially from DNS due to the enforcement of the energy dissipation for the smaller resolved scales belonging to the inertial range. Another substantial difference between the OSGS-VMS and LES methods is that, while for the OSGS-VMS the calculation of the energy in (2.27) and (2.31) is performed only as a post-process (for illustrating purposes), the discrete energy equation for the resolved scales needs to be solved at each time step for the calculation of the LES spectral eddy viscosity, with the consequent increment in the computational cost.

Finally, we address the scale dependence of the LES and the OSGS-VMS nonlinear terms spectral eddy viscosity by using a grid resolution and cutoff wavenumber of $N = 80$. The plot of the eddy viscosity with respect to the resolved wavenumbers is presented in Fig 2.10. It can be seen that the eddy viscosity given by the static Smagorinsky model is almost flat, only increasing at high resolved wavenumbers. The amount of dissipation given by the LES model is tightly related to the parameters of the spectral model. Decreasing the Kolmogorov constant intensifies viscosity, but also makes the solution over-dissipative (as demonstrated in Fig. 2.9). In the case of the OSGS-VMS method, even if the obtained eddy viscosity is higher than the one obtained with the LES results, the energy spectrum is more accurate than the one obtained with the $C_k = 0.05$ LES model. The OSGS-VMS viscosity exhibits a sharp eddy increment near the cutoff wavenumber that is not observed for the LES results. Numerical experiments with finer grids confirm this sharp behavior, which is defined solely by the numerical terms involving the variational subscales, and without the need of tuning numerical parameters.

We can effectively conclude that the contribution from the Galerkin form of the problem is not sufficient to correctly reproduce the turbulent energy spectrum. Moreover, it is clear that for coarse discretizations the present OSGS-VMS formulation works as a turbulence model, only depending on the approximations made to derive the subscales (OSGS method and τ definition). The numerical terms involving the dynamic evolution of the subscales seem to accurately dissipate the energy of the system and to account for the subscale "turbulent effects" onto the resolved scales even better than former LES methods. In the spirit of the discussion in [56], the results of the present work encourage the application of ILES numerical methods with built-in numerical dissipation instead of LES methods. Nevertheless, this conclusion is drawn from the particular case of the steady turbulent solution obtained with the OSGS-VMS method, and the static Smagorinsky model with spectral eddy viscosity.

2.5 Conclusions

In this chapter, an Orthogonal Sub-Grid Scales - Variational Multiscale (OSGS-VMS) method for numerically approximating the Burgers problem has been presented. The analysis of the Burgers equation has been done in the Fourier space, which allows us to clarify the scale dependence of the numerical diffusion introduced by the variational multiscale method. In particular, the numerical dissipation introduced with the orthogonal sub-grid scales has been explained in detail. For this, the space for the orthogonal subscales has been defined in terms of the finite-dimensional resolved space, so that the numerical approximation of the unresolved scales improves as the grid is refined. Results have been contrasted with DNS simulations,

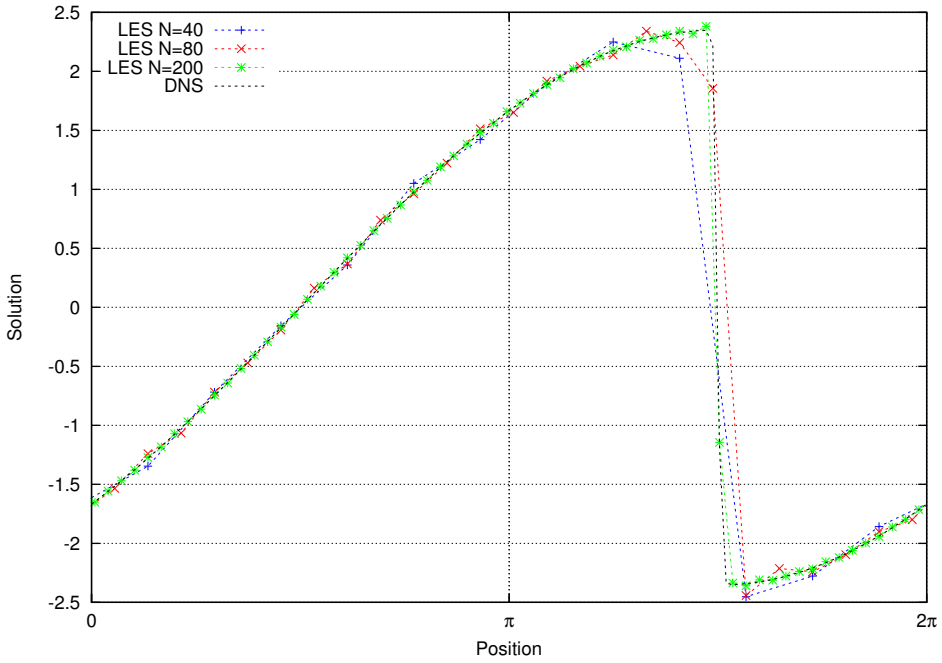


Figure 2.7: LES results: physical space solution.

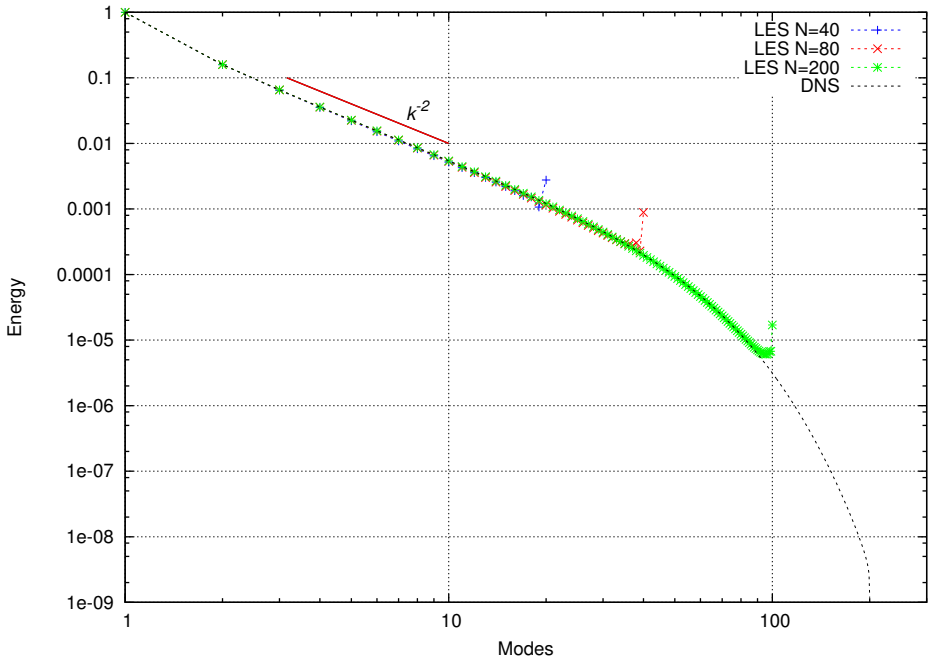


Figure 2.8: LES results: energy spectrum.

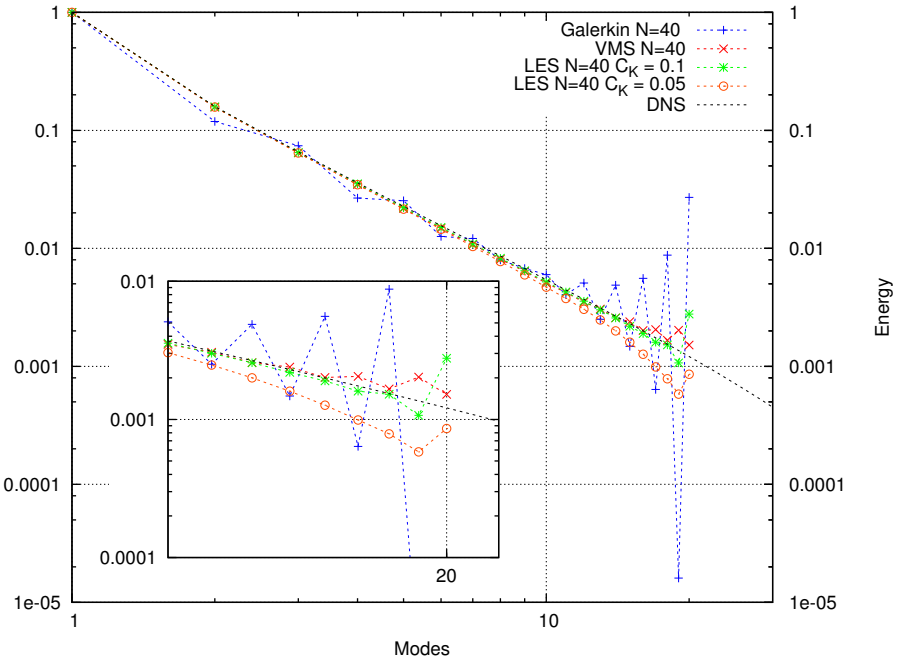


Figure 2.9: Energy spectrum results for Galerkin, OSGS-VMS, and LES simulations.

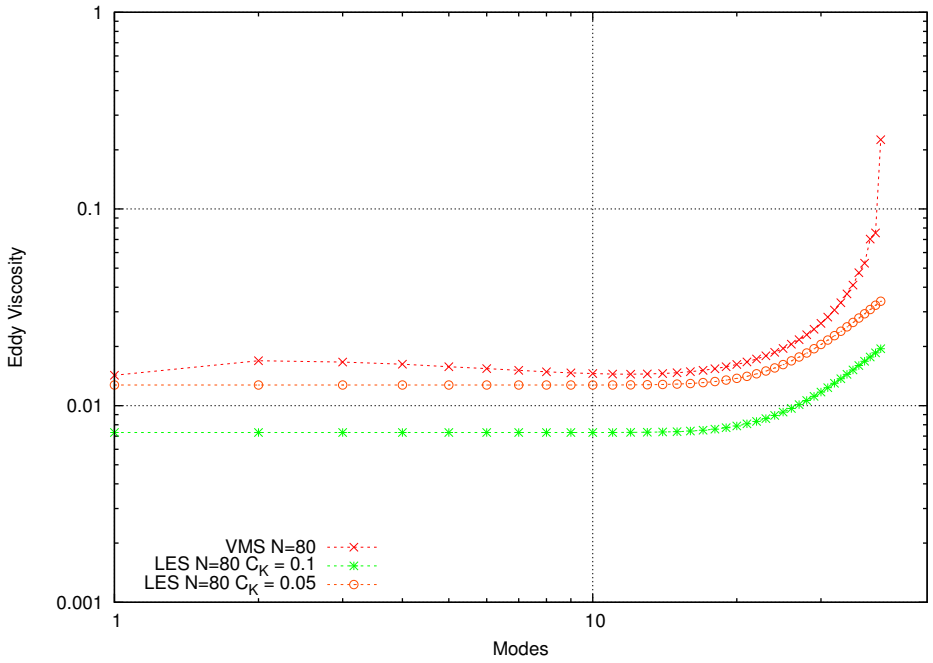


Figure 2.10: Eddy viscosity results for OSGS-VMS and LES simulations.

validating the ability of the OSGS-VMS model to describe the turbulent behavior of the Burgers equation. The scale dependence of the introduced numerical dissipation terms has been also compared with a spectral LES method. An accurate dissipative structure for the OSGS-VMS method has been found, which arises from the numerical approximation exclusively, and without the need of modifying the continuous equation.

Chapter 3

Variational multi-scale finite element approximation of the compressible Navier-Stokes equations written in conservative variables

In this chapter, the variational multi-scale framework is applied to the finite element approximation of the compressible Navier-Stokes equations written in conservation form. Even though this formulation is relatively well known, some particular features that have been applied with great success in other flow problems are incorporated. The orthogonal subgrid scales, the non-linear tracking of these subscales, and their time evolution are applied. Moreover, a systematic way to design the matrix of algorithmic parameters from the perspective of a Fourier analysis is given, and the adjoint of the non-linear operator including the volumetric part of the convective term is defined. Because the subgrid stabilization method works in the streamline direction, an anisotropic shock capturing method that keeps the diffusion unaltered in the direction of the streamlines, but modifies the crosswind diffusion is implemented. The artificial shock capturing diffusivity is calculated by using the orthogonal projection onto the finite element space of the gradient of the solution, instead of the common residual definition.

3.1 Introduction

The compressible Navier-Stokes equations are commonly used to model flow problems when compressibility effects become relevant. Some areas that require a compressible flow description are the aerodynamics and aeroacoustics fields, in which applications range from turbomachinery design to speech therapy. The compressible Navier-Stokes equations consist of the conservation of mass, momentum and energy equations. Thermodynamical properties and constitutive relations of the fluid close the mathematical description. We restrict our problem to perfect fluids in the gas state, which we model using Newton's law for fluids together with the caloric equation, the ideal gas law, and Fourier's heat law. However, other compressible fluids can be modeled by the compressible Navier-Stokes equations using the appropriate constitutive definitions.

In this work, we are interested in the finite element method approximation of compressible flow problems. In particular, we solve this set of equations using the conservative variables formulation, that is, defining density, momentum, and total energy as the problem unknowns. When these equations are approximated by the classical Galerkin approach, numerical instabilities may appear due to the hyperbolic nature of the equations. The first attempt to deal with this instability was the introduction of a stabilizing term into the Galerkin finite element formulation. Within this concept, the Streamline Upwind Petrov Galerkin (SUPG) method in [61] was the first method adopted for solving the compressible Navier-Stokes equations. In that formulation, the authors applied the stabilization methods previously developed for the convection-diffusion equation. The main idea was to optimally introduce numerical diffusion along the streamlines using a stabilization term that contained a matrix of algorithmic parameters, a certain operator applied to the test function, and the residual of the differential equation (e.g. the one in [62]). An important contribution of this pioneering work was the inclusion of the largest eigenvalue of the hyperbolic system into the matrix of algorithmic parameters. More recently, in references [63–65], the SUPG stabilization has been applied into compressible flow formulations. Some later stabilizations were formulated based on the Galerkin Least Squares (GLS) method. For example, in the articles [66–68], the authors transformed the conservative variables into entropy variables using Jacobian transformation matrices.

The multi-scale concept was first included in the compressible Navier-Stokes formulations to account for the effect of the unresolved scales as a turbulence model instead of as a stabilization method. Within this branch, a mixed formulation was proposed in [69] that approximated the solution separating a priori the resolved and unresolved turbulent scales. They included the effect of unresolved scales into the solution using a Reynolds stress tensor. The unresolved scales were calculated with a finite volume cell-agglomeration projector. That initial work was contrasted in [70], specifically in the definition of the unresolved scales by means of the cell-agglomeration projector; they proposed instead a Fourier-Spectral projector, which could be implemented with a discontinuous Galerkin method. Moreover, in references [71, 72] turbulence modeling concept was implemented and presented turbulence energy spectra results for turbulent compressible flows.

The stabilization methods proposed in this work are based on the Variational Multi-Scale (VMS) framework introduced in [17] for the scalar convection-diffusion-reaction problem. The method splits the unknowns of the problem into a coarse-scale that belongs to the finite element space and a subgrid scale or subscale, which is the remainder. The original problem is subdivided into two separated problems, one for the finite element scale, and another for the subscales. To avoid increasing the number of degrees of freedom of the problem, an approximation over the subscales is done in terms of the resolved scales. In order to properly do this approximation, we propose to study the behavior of the compressible problem from the perspective of a Fourier analysis, instead of the previous approximations in [73, 74]. Hence, we aim to give a systematic way to design the approximation over the subscales, that is, to define the matrix of algorithmic parameters for the compressible case.

The main idea of the VMS formulation is to include the effect of the unresolved scales in order to stabilize the finite element equations. The VMS stabilized finite element formulation was introduced to solve the compressible Navier-Stokes equations in [73]. They extended satisfactorily the formulation previously done for the incompressible Navier-Stokes equations in [75]. In that work, they demonstrated the matrix of algorithmic parameters definition for a one-

dimensional advective-diffusive-reactive problem on quadratic elements. The contribution of the subscales into the resolved scales was done by integrating by parts the finite element scale equation. More recently, in [74] the VMS method was applied in order to stabilize the Euler equations, and the authors demonstrated the convergence of the numerical method in a wide range of stratified flows. They restricted the explicit formulation to a linear Euler time integration scheme. Even though the VMS formulation of the compressible problem has already been worked by the last mentioned authors, we incorporate some particular features that we have applied in other flow problems (see [15] and references therein). Since there are different ways to define the subscales, three different approaches are solved in this work. The first possibility is the definition of the space of the subscales, which can be either the space of finite element residuals or the space orthogonal to the finite element space [43, 76]. The second possibility is the inclusion of the transient term of the subscales equation [46]. The third possibility is the inclusion of the subscales in all the non-linear terms of the problem [57]. In addition to these definitions, we take into account the volumetric part of the convective term in the compressible nonlinear operator adjoint, which has not been proposed before in the stabilized formulations.

Furthermore, at the supersonic regime, some localized instabilities may arise from sharp gradients in the solution, which are inherent to the physics of the problem. Hence, the stabilized formulation requires being complemented with a local shock capturing term, in order to yield stability and convergence in the entire domain. The first approach was the residual-based shock capturing techniques that control oscillations in a non-linear fashion depending on the solution. This type of shock capturing operator was introduced in [77] and later in [78] for a SUPG compressible flow formulation. Some other formulations, like the ones in [79–82] evaluated several types of shock capturing techniques. In contrast with the previous formulations in [18, 65, 83], we aim to introduce the numerical diffusion in a "physical manner" for the compressible problem. That is, we modify the diffusion of the momentum and energy equations, but avoid introducing artificial diffusion into the mass equation. In this topic, we also propose to use the orthogonal projection onto the finite element space of the gradient of the unknown, instead of the common residual definition. Because the sub-grid stabilization method works in the streamline direction, we implement an anisotropic shock capturing method that keeps the diffusion unaltered in the direction of the streamlines, but modifies the crosswind diffusion. The scope of this proposed shock-capturing technique is evaluated with some numerical examples.

In this chapter we use an explicit time integration scheme in order to integrate the temporal derivatives, more precisely, we implement the family of explicit Runge-Kutta methods. Moreover, we calculate the time integration method for the temporal derivative of the dynamic subscales exploiting the explicit scheme adopted.

We perform numerical tests of the formulations presented. The first section of tests corresponds to the study of the formulations in the subsonic range, where there is no need for local stabilization and it is possible to compare the global stabilized formulations. We implement the three-dimensional lid-driven cavity and the flow past a cylinder problems, both in the subsonic range, and benchmark our solutions to those published by other authors. We use the cylinder results obtained with linear triangular elements to compare the convergence of the proposed methods. As another type of test, we make a comparison between the shock capturing methods (proposed in this thesis) using the supersonic reflected shock problem. Finally, we present the solution for the compressible viscid supersonic flow past a cylinder including the global and

local stabilization methods. This example closes the evaluation of the formulation.

This chapter is organized as follows. In Section 3.2 we present the compressible Navier-Stokes problem. Section 3.3 contains the numerical approximation using the VMS finite element formulation of the compressible Navier-Stokes equations. We also present the demonstration of the matrix of stabilization parameters, and the distinct approaches to solving the subscales equation. The shock capturing methods and the explicit time integration scheme are also discussed in Section 3.3. In Section 3.4 the numerical examples are presented, and the proposed stabilization is discussed. Finally, in Section 3.5 conclusions are stated.

3.2 The compressible Navier-Stokes problem

3.2.1 Initial and boundary value problem

The problem we consider consists in the Navier-Stokes equations posed in a time interval $(0, t_f)$ and in a domain $\Omega \subset \mathbb{R}^d$, being d the number of space dimensions ($d = 2$ or 3). Let $t \in (0, t_f)$ be a given time instant in the temporal domain, and $\boldsymbol{x} \in \Omega$ a given point in the spatial domain. Let Γ be the boundary of the domain Ω, and $\boldsymbol{n}$ the geometric unit outward normal vector on Γ. We split Γ into two sets: the *Dirichlet* boundary denoted as Γ_G, and the *Neumman* boundary denoted as Γ_N. Considering a compressible, Newtonian and viscous fluid, the governing equations are the conservation of mass, momentum, and energy written in conservation form:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}\left(\rho u_i\right) = 0, \tag{3.1}$$

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}\left(\rho u_j u_i + p\delta_{ij} - \tau_{ji}\right) = \rho f_i, \tag{3.2}$$

$$\frac{\partial}{\partial t}\left(\rho\left(e + \frac{1}{2}u_i u_i\right)\right) + \frac{\partial}{\partial x_j}\left(\rho u_j\left(\mathrm{h} + \frac{1}{2}u_i u_i\right) - u_i\tau_{ij} + q_j\right) = \rho f_i u_i + \rho r, \tag{3.3}$$

together with appropriate boundary and initial conditions. The usual summation convention is implied in the equations presented before, with indices running from 1 to d. In these equations ρ is the density, p is the pressure, $\boldsymbol{u}$ is the velocity, $\boldsymbol{\tau}$ is the viscous stress tensor, $\boldsymbol{f}$ is a body force vector, e is the internal energy, h is the enthalpy, $\boldsymbol{q}$ is the heat flux vector, r is a heat source/sink term and $\boldsymbol{I} = [\delta_{ij}]$ is the identity or *Kronecker* tensor. Supplementary constitutive relations are considered in order to close the problem. For the viscous part of the stress tensor we use the relation

$$\tau_{ij}\left(\boldsymbol{u}\right) = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2\mu}{3}\left(\frac{\partial u_l}{\partial x_l}\right)\delta_{ij}, \tag{3.4}$$

where μ is the viscosity. For the heat flux vector we use Fourier's law,

$$q_i\left(\theta\right) = -\lambda\frac{\partial \theta}{\partial x_i}, \tag{3.5}$$

where λ is the thermal conductivity and θ is the temperature of the fluid. The caloric equation $e = c_v\left(\theta\right)\theta$ and the perfect gas state equation $p = \rho R\theta$ are used to calculate the pressure

and the acoustic speed c. In these relations the specific heat at constant volume $c_v(\theta)$ and the specific heat at constant pressure $c_p(\theta)$ are thermodynamic properties of the fluid. We also define $\gamma = c_p/c_v$ for the ratio between the specific heats, and $R = c_p - c_v$ for the specific gas constant.

Hereafter, let us denote the transpose operation by the superscript $\top$. We write the conservation equations (3.1)-(3.3) in system form introducing the vector of conservative variables $\boldsymbol{U} = \left(\rho, \quad \boldsymbol{m}, \quad e_{\text{tot}}\right)^\top$, with density, momentum $\boldsymbol{m} = \rho\boldsymbol{u}$ and total energy $e_{\text{tot}} = \rho\left(e + \boldsymbol{u}\cdot\boldsymbol{u}/2\right)$ as its components. The system form of the compressible Navier-Stokes equations written in conservation variables is

$$\partial_t \boldsymbol{U} + \partial_j \boldsymbol{E}_j(\boldsymbol{U}) + \partial_j \boldsymbol{G}_j(\boldsymbol{U}) - \boldsymbol{S}(\boldsymbol{U}) = \boldsymbol{0} \qquad \text{in} \quad \Omega \subset \mathbb{R}^d, t \in (0, t_f), \tag{3.6}$$

together with appropriate boundary and initial conditions. Here ∂_t and ∂_j are short notations that indicate the Eulerian time derivative and $\partial/\partial x_j$, respectively. The *convective* flux in the jth-direction $\boldsymbol{E}_j$ and the *diffusive* flux in the jth-direction $\boldsymbol{G}_j$ are defined as:

$$\boldsymbol{E}_j(\boldsymbol{U}) = \begin{bmatrix} \rho u_j \\ \rho u_j u_1 + p\delta_{1j} \\ \rho u_j u_2 + p\delta_{2j} \\ \rho u_j u_3 + p\delta_{3j} \\ u_j\left(e_{\text{tot}} + p\right) \end{bmatrix}, \qquad \boldsymbol{G}_j(\boldsymbol{U}) = \begin{bmatrix} 0 \\ -\tau_{j1} \\ -\tau_{j2} \\ -\tau_{j3} \\ -u_i\tau_{ij} + q_j \end{bmatrix}. \tag{3.7}$$

The divergence of these fluxes is written in a more convenient manner as $\partial_j \boldsymbol{E}_j(\boldsymbol{U}) = \boldsymbol{A}_j(\boldsymbol{U})\,\partial_j \boldsymbol{U}$ and $\partial_j \boldsymbol{G}_j(\boldsymbol{U}) = -\partial_k\left(\boldsymbol{K}_{kj}(\boldsymbol{U})\,\partial_j \boldsymbol{U}\right)$, with the definition of the Euler Jacobian matrix $\boldsymbol{A}_j(\boldsymbol{U}) = \partial \boldsymbol{E}_j(\boldsymbol{U})/\partial \boldsymbol{U}$, and the diffusivity matrix $\boldsymbol{K}(\boldsymbol{U}) = [\boldsymbol{K}_{kj}(\boldsymbol{U})]$. As a remark, both the Euler Jacobian and the diffusivity matrices depend on the variables of unknowns through the constitutive relations of the fluid. The last term, the vector of *sources* $\boldsymbol{S}(\boldsymbol{U}) = \left[0, \quad \rho\boldsymbol{f}, \quad \rho\boldsymbol{f}\cdot\boldsymbol{u} + \rho r\right]^\top$, contains the right hand side terms of equations (3.1)-(3.3). We can linearize this term by introducing a reactive matrix

$$\boldsymbol{S} = \begin{bmatrix} 0 & 0 & 0 \\ \boldsymbol{f} & 0 & 0 \\ r & \boldsymbol{f}^\top & 0 \end{bmatrix}, \tag{3.8}$$

that multiplies the vector of unknowns $\boldsymbol{S}(\boldsymbol{U}) = \boldsymbol{S}\boldsymbol{U}$. The nonlinear operator

$$\boldsymbol{\mathcal{L}}\left(\breve{\boldsymbol{U}}; \boldsymbol{U}\right) = \boldsymbol{A}_j\left(\breve{\boldsymbol{U}}\right)\partial_j \boldsymbol{U} - \partial_k\left(\boldsymbol{K}_{kj}\left(\breve{\boldsymbol{U}}\right)\partial_j \boldsymbol{U}\right) - \boldsymbol{S}\boldsymbol{U}, \tag{3.9}$$

includes the previous definitions for the convective, diffusive, and reactive terms. The nonlinearity in the first argument of the operator arises from the dependency of the Euler Jacobian and diffusivity matrices on the unknowns. However, the operator is linear in the second argument.

Equation (3.6) may be now written using the nonlinear operator as

$$\partial_t \boldsymbol{U} + \boldsymbol{\mathcal{L}}(\boldsymbol{U}; \boldsymbol{U}) = \boldsymbol{0} \qquad \text{in} \quad \Omega \subset \mathbb{R}^d, t \in (0, t_f). \tag{3.10}$$

The compressible Navier-Stokes problem is a nonlinear initial and boundary value problem of hyperbolic type, subject to appropriate definitions for the boundary and initial conditions,

which can be written in vector form as

$$\boldsymbol{B}(\boldsymbol{U}) = \boldsymbol{H} \qquad \text{on} \quad \Gamma_N, \quad t \in (0, t_f), \tag{3.11}$$

$$\boldsymbol{D}(\boldsymbol{U}) = \boldsymbol{D}(\boldsymbol{U}_g) \qquad \text{on} \quad \Gamma_G, t \in (0, t_f), \tag{3.12}$$

$$\boldsymbol{U} = \boldsymbol{U}_0(\boldsymbol{x}) \qquad \text{in} \quad \Omega, t = 0. \tag{3.13}$$

The Dirichlet boundary operator $\boldsymbol{D}(\cdot)$ is used to impose the components of $\boldsymbol{U}$ on different parts of Γ. The Neumann boundary conditions $\boldsymbol{H}$ are given by the operator $\boldsymbol{B}(\cdot)$. Due to the hyperbolic and compressible nature of the problem, Dirichlet boundary conditions must be imposed in the inlet part of the boundary $\Gamma_{\text{in}} = \{\boldsymbol{x} \in \Gamma | (\boldsymbol{u} \cdot \boldsymbol{n})(\boldsymbol{x}) < 0\}$. For the sake of simplicity we have grouped Neumann conditions on Γ_N and Dirichlet conditions on Γ_G, but mixed types of conditions could be applied to different variables (momentum and total energy) on the same part of the boundary.

The nondimensional Mach number $\mathrm{M} = |\boldsymbol{u}|/c$ is used to calculate the compressibility regime. It can range from subsonic $(M < 0.8)$, transonic $(0.8 < M < 1.2)$, supersonic $(M > 1.2)$, and hypersonic flow $(M >> 1)$. Here we focus our attention to the subsonic, transonic, and supersonic regimes. All conservative variables are imposed at the inflow part of the Dirichlet boundary, regardless the compressibility regime. For the supersonic case, no Dirichlet conditions need to be imposed at the outflow $\Gamma_{\text{out}} = \{\boldsymbol{x} \in \Gamma | (\boldsymbol{u} \cdot \boldsymbol{n})(\boldsymbol{x}) > 0\}$. In the case of subsonic flow, only density is imposed at the outflow boundary Γ_{out}. Solid boundaries can be represented as a slip condition $\boldsymbol{u} \cdot \boldsymbol{n} = 0$, as a no slip condition $\boldsymbol{u} = \boldsymbol{0}$, or as an isothermal wall $e_{\text{tot}} = \rho c_v \theta$. In addition, initial conditions for the conservative variables must be defined at $t = 0$, which are of the form $\boldsymbol{U} = \boldsymbol{U}_0(\boldsymbol{x})$, with $\boldsymbol{U}_0(\boldsymbol{x})$ functions defined on the whole domain Ω. We will explicitly indicate in our examples how the initial and boundary conditions are prescribed.

3.2.2 The variational problem

Let us introduce some notation in order to write the variational form of the problem. Let $L^2(\Omega)$ be the space of square integrable functions in the domain Ω. We use the symbol $\langle ., . \rangle$ to denote the integral of the product of two functions, including the dual pairing, assuming it is well defined. The L^2 inner product in Ω is denoted by $(\cdot, \cdot)$.

Let $\boldsymbol{\mathcal{W}}$ be an appropriate test functions space. The weak form of the problem is obtained by testing (3.10) against an arbitrary test function $\boldsymbol{V}$. The weak form can be written as: find $\boldsymbol{U}$ belonging to the space of unknowns, such that

$$(\boldsymbol{V}, \partial_t \boldsymbol{U}) + \langle \boldsymbol{V}, \boldsymbol{\mathcal{L}}(\boldsymbol{U}; \boldsymbol{U}) \rangle = 0 \quad \forall \boldsymbol{V} \in \boldsymbol{\mathcal{W}}, \tag{3.14}$$

together with appropriate sets of boundary and initial conditions.

3.3 Numerical approximation

3.3.1 The Galerkin finite element discretization

The Finite Element Method (FEM) approximation of the continuous variational problem (3.14) can be done with the standard Galerkin method. Let us consider the finite element partition

$\mathcal{T}_h = \{K\}$ of the domain Ω. The diameter of the element partition is denoted by h. We define the test functions space $\boldsymbol{\mathcal{W}}_h \subset \boldsymbol{\mathcal{W}}$ as continuous picewise polynomial in space. The Galerkin FEM problem consists of finding a finite element solution $\boldsymbol{U}_h$ belonging to the trial space, such that

$$(\boldsymbol{V}_h, \partial_t \boldsymbol{U}_h) + \langle \boldsymbol{V}_h, \boldsymbol{\mathcal{L}}(\boldsymbol{U}_h; \boldsymbol{U}_h)\rangle = 0 \quad \forall \boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h, \tag{3.15}$$

together with the appropriate initial and boundary conditions of the problem. When the Galerkin method is used to solve this hyperbolic problem, which possesses non-symmetric operators, an unstable behaviour of the solution might appear when the convection is dominant, and due to the incompatibility of the interpolation of the different variables.

3.3.2 The space discrete variational multi-scale stabilized finite element formulation

We present a stabilized formulation for the compressible Navier-Stokes equations based on the VMS approach introduced in [17]. The basic idea is to approximate the effect of the components of the solution of the continuous problem that cannot be solved by the finite element mesh. It consists on the decomposition of the unknown $\boldsymbol{U} = \boldsymbol{U}_h + \widetilde{\boldsymbol{U}}$, into a coarse-scale $\boldsymbol{U}_h \in \boldsymbol{\mathcal{W}}_h$ that belongs to the finite element space and a subgrid scale or subscale $\widetilde{\boldsymbol{U}} \in \widetilde{\boldsymbol{\mathcal{W}}}$, which is the remainder. The spaces $\boldsymbol{\mathcal{W}}_h$ and $\widetilde{\boldsymbol{\mathcal{W}}}$ are such that $\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{W}}_h \oplus \widetilde{\boldsymbol{\mathcal{W}}}$. Hence, consistently $\boldsymbol{V} = \boldsymbol{V}_h + \widetilde{\boldsymbol{V}}$, where $\boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h$ and $\widetilde{\boldsymbol{V}} \in \widetilde{\boldsymbol{\mathcal{W}}}$. The variational formulation (3.14) can now be split into two equivalent subproblems:

$$(\boldsymbol{V}_h, \partial_t \boldsymbol{U}) + \langle \boldsymbol{V}_h, \boldsymbol{\mathcal{L}}(\boldsymbol{U}; \boldsymbol{U})\rangle = 0 \quad \forall \boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h, \tag{3.16}$$

$$\left(\widetilde{\boldsymbol{V}}, \partial_t \boldsymbol{U}\right) + \langle \widetilde{\boldsymbol{V}}, \boldsymbol{\mathcal{L}}(\boldsymbol{U}; \boldsymbol{U})\rangle = 0 \quad \forall \widetilde{\boldsymbol{V}} \in \widetilde{\boldsymbol{\mathcal{W}}}. \tag{3.17}$$

The objective is to approximate the subscales in order to end up with a problem for the finite element scale alone. On the one hand, we integrate by parts Eq. (3.16) and obtain

$$(\boldsymbol{V}_h, \partial_t \boldsymbol{U}) + \langle \boldsymbol{V}_h, \boldsymbol{\mathcal{L}}(\boldsymbol{U}; \boldsymbol{U}_h)\rangle + \langle \boldsymbol{\mathcal{L}}^*(\boldsymbol{U}; \boldsymbol{V}_h), \widetilde{\boldsymbol{U}}\rangle = 0 \quad \forall \boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h. \tag{3.18}$$

Here we have introduced the formal adjoint $\boldsymbol{\mathcal{L}}^*(\boldsymbol{U}, \cdot)$ of the operator $\boldsymbol{\mathcal{L}}(\boldsymbol{U}, \cdot)$. The adjoint operator is defined as $\langle \boldsymbol{V}, \boldsymbol{\mathcal{L}}(\boldsymbol{U}; \boldsymbol{W})\rangle = \langle \boldsymbol{\mathcal{L}}^*(\boldsymbol{U}; \boldsymbol{V}), \boldsymbol{W}\rangle$, for all $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W} \in \boldsymbol{\mathcal{W}}$. The duality might involve inter-element jump terms when finite element functions are considered. However, these inter-element terms are neglected by supposing that the subscales vanish at the element boundaries. With the above approximation, the outcome for the adjoint of the nonlinear operator (3.9) applied to the test functions vector is

$$\boldsymbol{\mathcal{L}}^*(\boldsymbol{U}; \boldsymbol{V}_h) = -\partial_j\left(\boldsymbol{A}_j^\top(\boldsymbol{U})\,\boldsymbol{V}_h\right) - \partial_j\left(\boldsymbol{K}_{kj}^\top(\boldsymbol{U})\,\partial_k \boldsymbol{V}_h\right) - \boldsymbol{S}^\top \boldsymbol{V}_h. \tag{3.19}$$

It is important to remark the contribution of the derivative of the Euler Jacobian matrix in the first term on the right hand side. In this work we linearize the derivatives of the first and second terms on the right hand side of the previous expression, respectively as

$$\frac{\partial}{\partial x_j}\left(\boldsymbol{A}_j^\top(\boldsymbol{U})\,\boldsymbol{V}_h\right) \approx \boldsymbol{A}_j^\top(\boldsymbol{U})\,\frac{\partial \boldsymbol{V}_h}{\partial x_j} + \frac{\partial \boldsymbol{A}_j^\top(\boldsymbol{U})}{\partial \boldsymbol{U}}\frac{\partial \boldsymbol{U}_h}{\partial x_j}\boldsymbol{V}_h, \tag{3.20}$$

$$\frac{\partial}{\partial x_j}\left(\boldsymbol{K}_{kj}^\top(\boldsymbol{U})\,\frac{\partial \boldsymbol{V}_h}{\partial x_k}\right) \approx \boldsymbol{K}_{kj}^\top(\boldsymbol{U})\,\frac{\partial^2 \boldsymbol{V}_h}{\partial x_j \partial x_k} + \frac{\partial \boldsymbol{K}_{kj}^\top(\boldsymbol{U})}{\partial \boldsymbol{U}}\frac{\partial \boldsymbol{U}_h}{\partial x_j}\frac{\partial \boldsymbol{V}_h}{\partial x_k}. \tag{3.21}$$

We complete the formulation for the finite element scale equation by integrating by parts the diffusive term of the nonlinear operator applied to the finite element unknowns. Since the normal component of the diffusive flux must be continuous across inter-element boundaries,

$$\sum_K \left(\boldsymbol{V}_h, n_k \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \boldsymbol{U}\right)_{\partial K} = 0 \quad \forall \boldsymbol{V}_h \in \mathcal{W}_h, \tag{3.22}$$

the equation for the finite element scale results in:

$$\begin{aligned}\left(\boldsymbol{V}_h, \partial_t \boldsymbol{U}_h\right) + \left(\boldsymbol{V}_h, \partial_t \widetilde{\boldsymbol{U}}\right) + \left(\boldsymbol{V}_h, \boldsymbol{A}_j\left(\boldsymbol{U}\right)\partial_j \boldsymbol{U}_h\right) + \left(\partial_k \boldsymbol{V}_h, \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \boldsymbol{U}_h\right) \\ - \left(\boldsymbol{V}_h, \boldsymbol{S}\boldsymbol{U}_h\right) + \sum_K \langle \boldsymbol{\mathcal{L}}^*\left(\boldsymbol{U}; \boldsymbol{V}_h\right), \widetilde{\boldsymbol{U}}\rangle_K = 0 \quad \forall \boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h.\end{aligned} \tag{3.23}$$

In the previous equation, $\langle \cdot, \cdot \rangle_K$ denotes the L^2 inner product over the element K.

On the other hand, if $\widetilde{\boldsymbol{P}}$ denotes the L^2 projection onto the space of subscales, the equation for the subgrid scale can be formally written as

$$\widetilde{\boldsymbol{P}}\left[\partial_t \widetilde{\boldsymbol{U}} + \boldsymbol{\mathcal{L}}\left(\boldsymbol{U}; \widetilde{\boldsymbol{U}}\right)\right] = \widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{U}; \boldsymbol{U}_h\right)\right], \tag{3.24}$$

where $\boldsymbol{R}\left(\cdot, \cdot\right) = \left(R_\rho\left(\cdot, \cdot\right), \boldsymbol{R}_{\boldsymbol{m}}\left(\cdot, \cdot\right), R_{e_{\text{tot}}}\left(\cdot, \cdot\right)\right)^\top$ stands for the residual vector, which is composed by the equations residuals, and is formally defined as

$$\boldsymbol{R}\left(\breve{\boldsymbol{U}}; \boldsymbol{U}\right) = -\partial_t \boldsymbol{U} - \boldsymbol{\mathcal{L}}\left(\breve{\boldsymbol{U}}; \boldsymbol{U}\right). \tag{3.25}$$

Since the subscales cannot be represented by the finite element mesh, the effect of the nonlinear operator applied to the subscales needs to be approximated. For this, we adopt a matrix of stabilization parameters that depends on the unknowns $\boldsymbol{\tau}\left(\boldsymbol{U}\right)$, such that an approximation of the nonlinear operator applied to the subscales is made in each element:

$$\boldsymbol{\mathcal{L}}\left(\boldsymbol{U}; \widetilde{\boldsymbol{U}}\right) \approx \boldsymbol{\tau}^{-1}\left(\boldsymbol{U}\right)\widetilde{\boldsymbol{U}}. \tag{3.26}$$

The way to construct this approximation is explained further below. Hence, for an adequate definition of the projection onto the subscales space, the subscales equation is

$$\widetilde{\boldsymbol{P}}\left[\partial_t \widetilde{\boldsymbol{U}} + \boldsymbol{\tau}^{-1}\left(\boldsymbol{U}\right)\widetilde{\boldsymbol{U}}\right] = \widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{U}; \boldsymbol{U}_h\right)\right]. \tag{3.27}$$

The previous equation is a nonlinear ordinary differential equation, which must be solved at the integration points. There are different approximations in order to calculate $\widetilde{\boldsymbol{U}}$ from this equation.

Here we use two possibilities to construct the space where the subscales belong. The first and the most common choice is to take it equal to the space of the finite element residuals. That is, to define the projection onto the subscales space as the identity $\widetilde{\boldsymbol{P}} = \boldsymbol{I}$ onto the space of finite element residuals. The second possibility is the so-called orthogonal subscales method, which defines the subscales orthogonal to the finite element space $\widetilde{\boldsymbol{W}} = \boldsymbol{W}_h^\perp$. In this case, the projection is defined to be the orthogonal projection onto the finite element space $\widetilde{\boldsymbol{P}} = \boldsymbol{P}_h^\perp = \boldsymbol{I} - \boldsymbol{P}_h$, being $\boldsymbol{P}_h$ the L^2-projection onto the finite element space.

Apart from the construction of the spaces where the subscales belong, we call the subscales *dynamic* if the temporal derivative of subscales is taken into account. Instead, if the temporal derivative of the subscales is neglected we call them *quasi-static* subscales. Another possibility is to neglect the subscales effect in all the non-linear terms, whereas, if we take it into account we call them *non-linear* subscales.

3.3.3 The matrix $\boldsymbol{\tau}$ of stabilization parameters

The key point in the design of the stabilized formulation is the construction of the matrix $\boldsymbol{\tau}$. In order to do this, we study the behavior of the problem from the perspective of a Fourier analysis. This strategy was introduced in some other formulations, like in [84, 85], and we apply it now to the compressible Navier-Stokes problem. We basically want to bound the effect of the nonlinear operator and approximate it with matrix $\boldsymbol{\tau}^{-1}$. First, let us consider the following Fourier transform denoted by $\widehat{\cdot}$, and defined on each element K,

$$\widehat{g}\left(\boldsymbol{k}\right) := \int_K \mathrm{e}^{-\mathrm{i}\frac{\boldsymbol{k}\cdot\boldsymbol{x}}{h}} g\left(\boldsymbol{x}\right) d\Omega_x, \tag{3.28}$$

where $\mathrm{i} = \sqrt{-1}$. The wave number $\boldsymbol{k}$ is defined as $\boldsymbol{k} = (k_1, ..., k_d)$, and h as the diameter of K. If n_j is the j-th component of the normal exterior to K, it can be checked that

$$\widehat{\frac{\partial g}{\partial x_j}}\left(\boldsymbol{k}\right) = \int_{\partial K} n_j \mathrm{e}^{-\mathrm{i}\frac{\boldsymbol{k}\cdot\boldsymbol{x}}{h}} g\left(\boldsymbol{x}\right) d\Gamma_x + \mathrm{i}\frac{k_j}{h}\widehat{g}\left(\boldsymbol{k}\right). \tag{3.29}$$

The basic heuristic assumption is that $\widetilde{\boldsymbol{U}}$ is highly fluctuating, and therefore only contains *high wave numbers*

$$\widehat{\frac{\partial \widetilde{\boldsymbol{U}}}{\partial x_j}}\left(\boldsymbol{k}\right) \approx \mathrm{i}\frac{k_j}{h}\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right). \tag{3.30}$$

As a consequence, we may assume that values of $\widetilde{\boldsymbol{U}}$ on ∂K can be neglected to approximate $\widetilde{\boldsymbol{U}}$ in the interior of K. In addition, we can linearize $\boldsymbol{\mathcal{L}}\left(\boldsymbol{U}, \widetilde{\boldsymbol{U}}\right) \approx \boldsymbol{\mathcal{L}}\widetilde{\boldsymbol{U}}$ by considering $\boldsymbol{U}$ as given and constant in the non-linear convective and diffusive matrices of (3.9). Taking into account the above considerations, we calculate the Fourier transform of $\boldsymbol{\mathcal{L}}\widetilde{\boldsymbol{U}}$ as

$$\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}\right)\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right) = \mathrm{i}\frac{k_j}{h}\boldsymbol{A}_j\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right) + \frac{k_k k_j}{h^2}\boldsymbol{K}_{kj}\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right) - \boldsymbol{S}\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right). \tag{3.31}$$

The proper scaling of the problem is crucial to discuss the approximation of matrix $\boldsymbol{\tau}^{-1}$. Let $\boldsymbol{U}, \boldsymbol{V}$ be elements in the domain of $\boldsymbol{\mathcal{L}}\boldsymbol{U}$ and $\boldsymbol{F}, \boldsymbol{G}$ elements in its range. Suppose that $\boldsymbol{\mathcal{L}}\boldsymbol{U} = \boldsymbol{F}$ is written in such a way that $\boldsymbol{F}^\top\boldsymbol{U}$ is dimensionally well defined. Let also the $\boldsymbol{M}$-norm of $\boldsymbol{F}$ be defined as $|\boldsymbol{F}|^2_{\boldsymbol{M}} := \boldsymbol{F}^\top\boldsymbol{M}\boldsymbol{F}$, the $\boldsymbol{M}^{-1}$-norm of $\boldsymbol{U}$ as $|\boldsymbol{U}|^2_{\boldsymbol{M}^{-1}} := \boldsymbol{U}^\top\boldsymbol{M}^{-1}\boldsymbol{U}$, and $\|\boldsymbol{F}\|^2_{L^2_{\boldsymbol{M}}(K)} := \int_K |\boldsymbol{F}|^2_{\boldsymbol{M}}$. In order to compare in a consistent manner the previously defined norms, we introduce the scaling matrix $\boldsymbol{M}$ that makes dimensionally correct the products

$$\boldsymbol{F}^\top\boldsymbol{M}\boldsymbol{G} \qquad \text{and} \qquad \boldsymbol{U}^\top\boldsymbol{M}^{-1}\boldsymbol{V}. \tag{3.32}$$

We propose that the stabilization matrix $\boldsymbol{\tau}^{-1}$ *must be such that* $\|\boldsymbol{\mathcal{L}}\|_{L^2_{\boldsymbol{M}}(K)} \leq \|\boldsymbol{\tau}^{-1}\|_{L^2_{\boldsymbol{M}}(K)}$. In order to devise a way to satisfy it, let us note that

$$\begin{aligned}\left\|\boldsymbol{\mathcal{L}}\widetilde{\boldsymbol{U}}\right\|_{L^2_{\boldsymbol{M}}(K)} &= \int_K \left|\boldsymbol{\mathcal{L}}\widetilde{\boldsymbol{U}}\right|^2_{\boldsymbol{M}} d\boldsymbol{x} \approx \int_{\mathbb{R}^d} \left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}\right)\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right)\right|^2_{\boldsymbol{M}} d\boldsymbol{k} \\ &\leq \int_{\mathbb{R}^d} \left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}\right)\right|^2_{\boldsymbol{M}} \left|\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right)\right|^2_{\boldsymbol{M}^{-1}} d\boldsymbol{k} \\ &= \left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} \int_{\mathbb{R}^d} \left|\widehat{\widetilde{\boldsymbol{U}}}\left(\boldsymbol{k}\right)\right|^2_{\boldsymbol{M}^{-1}} d\boldsymbol{k} \approx \left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} \left\|\widetilde{\boldsymbol{U}}\right\|^2_{L^2_{\boldsymbol{M}^{-1}}(K)}, \end{aligned} \tag{3.33}$$

where the first approximation comes from the fact that boundary values of $\widetilde{\boldsymbol{U}}$ have been discarded, and $\boldsymbol{k}^0$ is a wave number whose existence follows from the mean value theorem. From the previous development, we have that

$$\|\boldsymbol{\mathcal{L}}\|_{L^2_{\boldsymbol{M}}(K)} \leq \left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}}. \tag{3.34}$$

Our proposal is to choose $\boldsymbol{\tau}$ *diagonal and such that* $\left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} = |\boldsymbol{\tau}^{-1}|^2_{\boldsymbol{M}}$, with the components of $\boldsymbol{k}^0$ understood as algorithmic constants. From (3.33) we can figure out the following way to achieve this approximation

$$\left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} = \sup_{\widehat{\widetilde{\boldsymbol{U}}}} \frac{\left(\widehat{\widetilde{\boldsymbol{U}}}, \widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)^* \boldsymbol{M}\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\widehat{\widetilde{\boldsymbol{U}}}\right)}{\left(\widehat{\widetilde{\boldsymbol{U}}}, \boldsymbol{M}^{-1}\widehat{\widetilde{\boldsymbol{U}}}\right)}, \tag{3.35}$$

with $\widehat{\widetilde{\boldsymbol{U}}}^* \widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)^* \boldsymbol{M}\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\widehat{\widetilde{\boldsymbol{U}}} \in \mathbb{R}^+$ but $\widehat{\widetilde{\boldsymbol{U}}}^*, \widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)^*, \widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)$, and $\widehat{\widetilde{\boldsymbol{U}}}$, being complex.

Let us denote by $\text{spec}_{\boldsymbol{M}^{-1}}(\boldsymbol{B})$ the spectrum of the generalized eigenvalue problem $\boldsymbol{B}\boldsymbol{x} = \lambda\boldsymbol{M}^{-1}\boldsymbol{x}$, being λ an eigenvalue, and its spectral radius by $\breve{\rho}(\boldsymbol{B})$. Equation (3.35) is equivalent to the following eigenvalue problem: if there exists $\widehat{\widetilde{\boldsymbol{U}}}$ such that $\left(\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)^* \boldsymbol{M}\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right)\widehat{\widetilde{\boldsymbol{U}}} = \lambda\boldsymbol{M}^{-1}\widehat{\widetilde{\boldsymbol{U}}}$, then

$$\left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} = \max \text{spec}_{\boldsymbol{M}^{-1}}\left(\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)^* \boldsymbol{M}\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right). \tag{3.36}$$

Furthermore, in this work we propose to estimate separately each contribution of the convective, diffusive and reactive terms of (3.31), since

$$\left|\widehat{\boldsymbol{\mathcal{L}}}\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} \leq \left|\widehat{\boldsymbol{\mathcal{L}}}_C\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} + \left|\widehat{\boldsymbol{\mathcal{L}}}_D\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}} + \left|\widehat{\boldsymbol{\mathcal{L}}}_R\left(\boldsymbol{k}^0\right)\right|^2_{\boldsymbol{M}}, \tag{3.37}$$

where subindices C, D and R denote the convective, diffusive, and reactive terms, respectively. We aim to analyze the one-dimensional case in the x_1 direction, and thereafter generalize to multiple d dimensions.

3.3.3.1 Convective term

The solution of the one-dimensional hyperbolic problem

$$\boldsymbol{\mathcal{L}}_C(\boldsymbol{U}) = \boldsymbol{A}\frac{\partial \boldsymbol{U}}{\partial x_1}, \tag{3.38}$$

is bounded by the eigenvalues of the Jacobian matrix. The one-dimensional linearized Euler Jacobian matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ (\gamma-3)\frac{u_1^2}{2} & (3-\gamma)u_1 & (\gamma-1) \\ (\gamma-1)u_1^3 - \gamma\frac{u_1 e_{\text{tot}}}{\rho} & \gamma\frac{e_{\text{tot}}}{\rho} - (\gamma-1)\frac{3u_1^2}{2} & \gamma u_1 \end{bmatrix}, \tag{3.39}$$

can be decomposed as $\boldsymbol{A} = \boldsymbol{T}\boldsymbol{D}\boldsymbol{T}^{-1}$, with the right and left eigenvectors matrices defined as

$$\boldsymbol{T} = \begin{bmatrix} 1 & 1 & 1 \\ u_1 - c & u_1 + c & u_1 \\ -\frac{bu_1^2+2cbu_1-2}{2b} & \frac{bu^2+2cbu_1+2}{2b} & \frac{u_1^2}{2} \end{bmatrix}, \quad \boldsymbol{T}^{-1} = \begin{bmatrix} \frac{u_1(cbu_1+2)}{4c} & -\frac{cbu_1+1}{2c} & \frac{b}{2} \\ \frac{u_1(cbu_1-2)}{4c} & -\frac{cbu_1-1}{2c} & \frac{b}{2} \\ -\frac{bu_1^2-2}{2} & bu_1 & -b \end{bmatrix}, \tag{3.40}$$

and $b = \frac{\gamma-1}{c^2}$. For convenience, let us denote as $\text{diag}(\boldsymbol{e})$ the diagonal matrix with vector $\boldsymbol{e}$ on the diagonal. The diagonal matrix of eigenvalues results from the previous decomposition as

$$\boldsymbol{D} = \text{diag}\left(u_1 - c, u_1 + c, u_1\right). \tag{3.41}$$

The spectrum of $\left(\widehat{\boldsymbol{\mathcal{L}}}_C\left(\boldsymbol{k}^0\right)^* \boldsymbol{M}\widehat{\boldsymbol{\mathcal{L}}}_C\left(\boldsymbol{k}^0\right)\right)$ can be evaluated without difficulty by transforming (3.38) into a problem which posses $\boldsymbol{D}$ as convective matrix. This can be achieved with a projector defined as the matrix of left-eigenvectors $\boldsymbol{T}^{-1}$. Thus, the unknowns and force vectors can be transformed into:

$$\boldsymbol{W} = \boldsymbol{T}^{-1}\boldsymbol{U} = \left(\frac{\rho}{2\gamma} \quad , \frac{\rho}{2\gamma} \quad , \frac{(\gamma-1)\rho}{\gamma}\right)^\top, \tag{3.42}$$

$$\boldsymbol{H} = \boldsymbol{T}^{-1}\boldsymbol{F} = \left(\frac{\rho u_1}{2\gamma h} \quad , \frac{\rho u_1}{2\gamma h} \quad , \frac{(\gamma-1)\rho u_1}{\gamma h}\right)^\top. \tag{3.43}$$

Because the derivatives of $\boldsymbol{T}^{-1}$ are null as a result of the linearized Euler Jacobian matrix, we can transform the convective problem

$$\boldsymbol{\mathcal{L}}'_C(\boldsymbol{U}) = \boldsymbol{T}^{-1}\boldsymbol{\mathcal{L}}_C(\boldsymbol{U}), \tag{3.44}$$

into the following problem

$$\boldsymbol{\mathcal{L}}'_C(\boldsymbol{W}) = \boldsymbol{D}\frac{\partial \boldsymbol{W}}{\partial x}. \tag{3.45}$$

Problem (3.45) is the so-called *Riemman* problem, used for physical considerations. At this point $\boldsymbol{\mathcal{L}}'_C(\boldsymbol{W}) = \boldsymbol{H}$ is such that $\boldsymbol{H}^\top\boldsymbol{W} = \sum_{i=1}^3 H_i W_i$ is dimensionally well defined. Let $\boldsymbol{W}, \boldsymbol{Y}$ be elements in the domain of $\boldsymbol{\mathcal{L}}'_C$ and $\boldsymbol{H}, \boldsymbol{J}$ elements in its range. The scaling matrix defined as the identity matrix $\boldsymbol{M} = \boldsymbol{I}$ makes the product $\boldsymbol{H}^\top\boldsymbol{M}\boldsymbol{J}$ correct, because

$[\rho^2 u_1^2 h^{-2}] = [\rho^2 u_1^2 h^{-2}] = [\rho^2 u_1^2 h^{-2}] = M^2 L^{-6} T^{-2}$, where $[\cdot]$ stands for dimensional group, M is mass, L is length, and T is time. The product $\boldsymbol{W}^\top \boldsymbol{M}^{-1} \boldsymbol{Y}$ is also consistent, indeed $[\rho^2] = [\rho^2] = [\rho^2] = M^2 L^{-6}$.

The previous development makes it possible to analytically calculate the spectrum of $\left(\widehat{\boldsymbol{\mathcal{L}}}_C^{\prime} (\boldsymbol{k})^* \boldsymbol{M} \widehat{\boldsymbol{\mathcal{L}}}_C^{\prime} (\boldsymbol{k}) \right)$ with respect to $\boldsymbol{M}^{-1}$, which is given by

$$\text{spec}_{\boldsymbol{M}^{-1}} \left(\left(\frac{k_1^0}{h} \right)^2 \boldsymbol{D}^\top \boldsymbol{M} \boldsymbol{D} \right) = \left\{ \left(\tfrac{k_1^0 (u_1 + c)}{h} \right)^2, \quad \left(\tfrac{k_1^0 u_1}{h} \right)^2, \quad \left(\tfrac{k_1^0 (u_1 - c)}{h} \right)^2 \right\}. \tag{3.46}$$

Therefore, the spectral radius is given by

$$\breve{\rho} \left(\left(\frac{k_1^0}{h} \right)^2 \boldsymbol{D}^\top \boldsymbol{M} \boldsymbol{D} \right) = \left(\frac{k_1^0 (u_1 + c)}{h} \right)^2. \tag{3.47}$$

If we take the simple diagonal expression for the transformed matrix of stabilization parameters $(\boldsymbol{\tau}')^{-1} = \text{diag} \left((\tau_\rho')^{-1}, (\tau_m')^{-1}, (\tau_{e_{\text{tot}}}')^{-1} \right)$, the spectrum is

$$\text{spec}_{\boldsymbol{M}^{-1}} \left((\boldsymbol{\tau}')^{-1} \boldsymbol{M} (\boldsymbol{\tau}')^{-1} \right) = \left\{ \left(\tfrac{1}{\tau_\rho'} \right)^2, \quad \left(\tfrac{1}{\tau_m'} \right)^2, \quad \left(\tfrac{1}{\tau_{e_{\text{tot}}}'} \right)^2 \right\}. \tag{3.48}$$

Our proposal is to equate the spectral radius of both problems

$$\breve{\rho} \left((\boldsymbol{\tau}')^{-1} \boldsymbol{M} (\boldsymbol{\tau}')^{-1} \right) = \breve{\rho} \left(\left(\frac{k_1^0}{h} \right)^2 \boldsymbol{D}^\top \boldsymbol{M} \boldsymbol{D} \right). \tag{3.49}$$

We can fulfill condition (3.49) by making each eigenvalue of (3.48) be equal to (3.47). As a result, the stabilization matrix after retrieving the transformation $\boldsymbol{\tau}^{-1} = \boldsymbol{T} (\boldsymbol{\tau}')^{-1} \boldsymbol{T}^{-1}$ is defined for the one-dimensional convective case as

$$\boldsymbol{\tau}^{-1} = \begin{bmatrix} \tau_\rho^{-1} & 0 & 0 \\ 0 & \tau_m^{-1} & 0 \\ 0 & 0 & \tau_{e_{\text{tot}}}^{-1} \end{bmatrix}, \quad \text{with,} \quad \tau_\rho^{-1} = \tau_m^{-1} = \tau_{e_{\text{tot}}}^{-1} = \frac{k_1^0 (u_1 + c)}{h}. \tag{3.50}$$

3.3.3.2 Diffusive term

Furthermore, we also account for the contribution of the one-dimensional diffusive term

$$\boldsymbol{\mathcal{L}}_D (\boldsymbol{U}) = \frac{\partial}{\partial x_1} \left(\boldsymbol{K} \frac{\partial \boldsymbol{U}}{\partial x_1} \right), \tag{3.51}$$

by decomposing the one-dimensional linearized diffusivity matrix

$$\boldsymbol{K} = \begin{bmatrix} 0 & 0 & 0 \\ -\frac{4\nu u_1}{3} & \frac{4\nu}{3} & 0 \\ -\frac{4\nu u_1^2}{3} - \frac{\alpha e_{\text{tot}}}{\rho} + \alpha u_1^2 & \frac{4\nu u_1}{3} - \alpha u_1 & \alpha \end{bmatrix}, \tag{3.52}$$

into $\boldsymbol{K} = \boldsymbol{T}\boldsymbol{D}\boldsymbol{T}^{-1}$, with

$$\boldsymbol{T} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & u_1 \\ u_1 & 1 & \frac{e_{\text{tot}}}{\rho} \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} \frac{4\nu}{3} & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{T}^{-1} = \begin{bmatrix} -u_1 & 1 & 0 \\ u_1^2 - \frac{e_{\text{tot}}}{\rho} & -u_1 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \tag{3.53}$$

Here ν is the kinematic viscosity and α is the thermal diffusivity. We also transform the one-dimensional diffusion problem (3.51) using the matrix of left-eigenvectors $\boldsymbol{T}^{-1}$. The derivatives of $\boldsymbol{T}$ and $\boldsymbol{T}^{-1}$ are neglected. Hence, the transformed problem possess a diagonal diffusive matrix of real eigenvalues $\boldsymbol{D}$, which multiplies the derivative of the transformed unknowns. It can be checked that the transformed problem is dimensionally well defined and that the scaling matrix can be defined as the identity matrix $\boldsymbol{M} = \boldsymbol{I}$ because products (3.32) are dimensionally correct.

Consequently, the spectrum of the transformed diffusion problem is

$$\text{spec}_{\boldsymbol{M}^{-1}}\left(\left(\frac{k_1^0 k_1^0}{h^2}\right)^2 \boldsymbol{D}^\top \boldsymbol{M}\boldsymbol{D}\right) = \left\{\left(\frac{k_1^0 k_1^0 4\nu}{h^2 3}\right)^2, \quad \left(\frac{k_1^0 k_1^0 \alpha}{h^2}\right)^2, \quad 0\right\}. \tag{3.54}$$

Note that each one of these eigenvalues may be an upper bound for the spectrum depending on the kinematic viscosity and thermal diffusion values. Therefore, we assure that the spectral radius of the transformed stabilization matrix problem

$$\text{spec}_{\boldsymbol{M}^{-1}}\left(\left(\boldsymbol{\tau}'\right)^{-1} \boldsymbol{M} \left(\boldsymbol{\tau}'\right)^{-1}\right) = \left\{\left(\frac{1}{\tau'_\rho}\right)^2, \quad \left(\frac{1}{\tau'_m}\right)^2, \quad \left(\frac{1}{\tau'_{e_{\text{tot}}}}\right)^2\right\}, \tag{3.55}$$

be greater than the spectral radius of (3.54) by forcing each one of the components to be

$$\frac{1}{\tau'_\rho} = \frac{1}{\tau'_m} = \frac{1}{\tau'_{e_{\text{tot}}}} = a_m \frac{k_1^0 k_1^0 4\nu}{h^2 3} + a_e \frac{k_1^0 k_1^0 \alpha}{h^2}. \tag{3.56}$$

Here we have introduced a_m and a_e as parameters of the problem. After retrieving the transformation, we get the same values for all terms of the matrix of stabilization as the right-hand side of (3.56). In practice, we make a_m and a_e zero for all terms, except for τ_m^{-1}, for which we choose $a_m = 1$ and for τ_e^{-1}, for which we also guarantee $a_e = 1$. That is, we define the matrix of stabilization for the one-dimensional diffusive term as

$$\boldsymbol{\tau}^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \tau_m^{-1} & 0 \\ 0 & 0 & \tau_{e_{\text{tot}}}^{-1} \end{bmatrix}, \quad \text{with} \quad \tau_m^{-1} = \frac{k_1^0 k_1^0 4\nu}{h^2 3} \quad \text{and} \quad \tau_{e_{\text{tot}}}^{-1} = \frac{k_1^0 k_1^0 \alpha}{h^2}. \tag{3.57}$$

This approximation has been tested and shown to give good results, even though it is not the complete bound definition for the diffusive term.

3.3.3.3 Reactive term

In the case of the reactive term contribution $\boldsymbol{S}\boldsymbol{U}$, the one-dimensional reactive matrix is given by

$$\boldsymbol{S} = \begin{bmatrix} 0 & 0 & 0 \\ f_1 & 0 & 0 \\ r & f_1 & 0 \end{bmatrix}, \tag{3.58}$$

where f_1 is the one-dimensional body source, and r is the heat source. This matrix must be scaled with a diagonal scaling matrix $\boldsymbol{Q}$, which can be designed for convenience using reference scaling values. At this point we can note that the scaling matrix $\boldsymbol{Q} = \text{diag}\,(c^4, c^2, 1)$, with the acoustic speed c, gives a dimensionally well defined product $\left[\boldsymbol{U}^\top \boldsymbol{Q}\boldsymbol{S}\boldsymbol{U}\right] = M^2L^{-2}T^{-5}$. From now on, let $\boldsymbol{U}, \boldsymbol{V}$ be elements in the domain of the scaled matrix $\boldsymbol{Q}\boldsymbol{S}$ and $\boldsymbol{G}, \boldsymbol{H}$ elements in its range. The scaling matrix $\boldsymbol{M} = \text{diag}\,(c^{-6}, c^{-4}, c^{-2})$ makes products (3.32) correct. If we make the product $\boldsymbol{G}^\top \boldsymbol{M}\boldsymbol{H}$, it gives the correct dimensional terms $[\rho^2 u_1^{10} h^{-2} c^{-6}] = [\rho^2 u_1^8 h^{-2} c^{-4}] = [\rho^2 u_1^6 h^{-2} c^{-2}] = M^2L^2T^{-4}$. The product $\boldsymbol{U}^\top \boldsymbol{M}^{-1}\boldsymbol{V}$ is also correct since $[\rho^2 c^6] = [\rho^2 u_1^2 c^4] = [\rho^2 u_1^4 c^2] = M^2L^6T^{-6}$.

Hence, we can calculate the spectrum of the scaled reactive term as

$$\text{spec}_{\boldsymbol{M}^{-1}}\left(\left(\boldsymbol{Q}\boldsymbol{S}\right)^\top \boldsymbol{M}\boldsymbol{Q}\boldsymbol{S}\right) = \left\{\tfrac{r^2+2c^2f_1^2+\sqrt{r^4+4c^2f_1^2r^2}}{2c^8},\quad \tfrac{r^2+2c^2f_1^2-\sqrt{r^4+4c^2f_1^2r^2}}{2c^8},\quad 0\right\}. \tag{3.59}$$

Likewise, the spectrum of the scaled matrix of stabilization parameters problem is

$$\text{spec}_{\boldsymbol{M}^{-1}}\left(\boldsymbol{Q}\boldsymbol{\tau}^{-1}\boldsymbol{M}\boldsymbol{Q}\boldsymbol{\tau}^{-1}\right) = \left\{\left(\tfrac{\tau_\rho^{-1}}{c^2}\right)^2,\quad \left(\tfrac{\tau_m^{-1}}{c^2}\right)^2,\quad \left(\tfrac{\tau_{e_{\text{tot}}}^{-1}}{c^2}\right)^2\right\}. \tag{3.60}$$

We can fulfill the condition of equating the spectral radius of both problems by making each eigenvalue of (3.60) be equal to the first eigenvalue of (3.59). As a result, the matrix of stabilization is defined for the one-dimensional reactive term as

$$\boldsymbol{\tau}^{-1} = \begin{bmatrix} \tau_\rho^{-1} & 0 & 0 \\ 0 & \tau_m^{-1} & 0 \\ 0 & 0 & \tau_{e_{\text{tot}}}^{-1} \end{bmatrix}, \quad \text{with,}$$

$$\tau_\rho^{-1} = \tau_m^{-1} = \tau_{e_{\text{tot}}}^{-1} = \left(\frac{r^2 + 2c^2f_1^2 + \sqrt{r^4 + 4c^2f_1^2r^2}}{2c^4}\right)^{1/2}. \tag{3.61}$$

3.3.3.4 Extension to multiple dimensions

Finally, we can extend the one-dimensional analysis in order to approximate the multidimensional stabilization matrix. Let us denote by $\boldsymbol{k}^0$ the multidimensional vector of algorithmic parameters $[\boldsymbol{k}^0] = k_i^0, \quad 1 \leq i \leq d$, where k_i^0 is the algorithmic parameter in the i-th direction. We propose to apply the one-dimensional momentum stabilization parameter equally into all dimensions of the momentum equation. Therefore, the stabilization matrix for multiple dimensions can be computed as

$$\boldsymbol{\tau}^{-1} = \begin{bmatrix} \tau_\rho^{-1} & \boldsymbol{0}^\top & 0 \\ \boldsymbol{0} & \tau_m^{-1}\boldsymbol{I} & \boldsymbol{0} \\ 0 & \boldsymbol{0}^\top & \tau_{e_{\text{tot}}}^{-1} \end{bmatrix}, \tag{3.62}$$

with $\mathbf{0}$ being the vector of $\mathbb{R}^d$ with zero in all its components and

$$\tau_{\rho}^{-1} = \frac{C_2\left(|\boldsymbol{u}|+c\right)}{h} + C_3\left(\frac{r^2+2c^2\left|\boldsymbol{f}\right|^2+\sqrt{r^4+4c^2\left|\boldsymbol{f}\right|^2r^2}}{2c^4}\right)^{1/2}, \tag{3.63}$$

$$\tau_{m}^{-1} = \frac{C_1}{h^2}\frac{4\nu}{3} + \frac{C_2\left(|\boldsymbol{u}|+c\right)}{h} + C_3\left(\frac{r^2+2c^2\left|\boldsymbol{f}\right|^2+\sqrt{r^4+4c^2\left|\boldsymbol{f}\right|^2r^2}}{2c^4}\right)^{1/2}, \tag{3.64}$$

$$\tau_{e_{\text{tot}}}^{-1} = \frac{C_1\alpha}{h^2} + \frac{C_2\left(|\boldsymbol{u}|+c\right)}{h} + C_3\left(\frac{r^2+2c^2\left|\boldsymbol{f}\right|^2+\sqrt{r^4+4c^2\left|\boldsymbol{f}\right|^2r^2}}{2c^4}\right)^{1/2}. \tag{3.65}$$

In these expressions C_1, C_2 and C_3 are algorithmic parameters. The algorithmic parameter C_1 approximates $\left|\boldsymbol{k}^0\right|^2$. On the other hand, the algorithmic constant C_2 approximates the $\left|\boldsymbol{k}^0\right|$ multiplied by the cosine of the angle formed by $\boldsymbol{k}^0$ and $\boldsymbol{u}$. The reactive contribution is multiplied by the algorithmic constant C_3. In this work we take $C_1 = 12p^2$, $C_2 = 2p$ and $C_3 = 1$, where p is the order of the finite element interpolation. The matrix of stabilization parameters depends on velocity and the acoustic speed. This non-linearity $\boldsymbol{\tau}\left(\boldsymbol{U}\right)$ is also considered for non-linear subscales.

3.3.4 Shock capturing technique

The previous stabilized finite element formulation yields a globally stable solution but does not guarantee stability in the presence of sharp gradients of the solution.

Indeed, gradients of the solution may appear in the supersonic compressible flow in the form of transonic shocks. Consequently, the globally stabilized formulation requires being complemented with a shock-capturing technique in order to yield stability and convergence in the entire domain. The main idea of such a shock-capturing technique is to increase the amount of numerical dissipation in the proximity of sharp gradients. The method adds some artificial kinematic viscosity ν_{SC} and thermal diffusivity α_{SC} into the diffusive Galerkin term of the finite element equation (3.23). We aim to introduce the numerical diffusion in such way that we modify diffusion in the momentum and energy equations but avoiding the modification of the mass equation.

We consider two non-linear methods in order to calculate the added diffusion values. The first non-linear method that we implement is a *residual* based technique, which is consistent, in the sense that if it is applied to the exact solution $\boldsymbol{U}$, the added diffusion is zero. For this technique we calculate the artificial kinematic viscosity using

$$\nu_{\text{SC}} = \left(\frac{1}{2}C_a h\right)\frac{\left|\boldsymbol{R}_{\boldsymbol{m}}\left(\boldsymbol{U},\boldsymbol{U}_h\right)\right|}{\left|\nabla\boldsymbol{m}_h\right|} \quad \text{if} \quad \left|\nabla\boldsymbol{m}_h\right| \neq 0, \quad \nu_{\text{SC}} = 0 \quad \text{otherwise}, \tag{3.66}$$

where C_a is an algorithmic constant, h is the characteristic length that gives dimensional consistency to the expression, and $\left|\nabla\boldsymbol{m}_h\right|$ is the Frobenius norm of the gradient of the finite

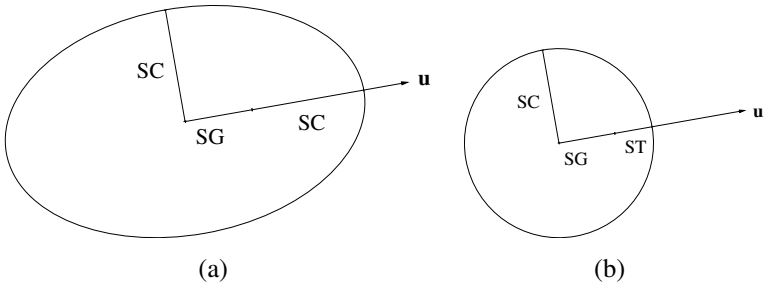


Figure 3.1: Shock capturing methods: (a) Isotropic diffusion ellipse, (b) Anisotropic diffusion ellipse.

element solution for momentum. Similarly, for the artificial thermal diffusivity we use

$$\alpha_{\text{SC}} = \left(\frac{1}{2}C_a h\right) \frac{|\boldsymbol{R}_{e_{\text{tot}}}(\boldsymbol{U}, \boldsymbol{U}_h)|}{|\nabla e_{\text{tot},h}|} \quad \text{if} \quad |\nabla e_{\text{tot},h}| \neq 0, \quad \alpha_{\text{SC}} = 0 \quad \text{otherwise.} \tag{3.67}$$

In the previous equation, $|\nabla e_{\text{tot},h}|$ is the norm of the gradient of the finite element solution for the total energy.

The second non-linear method is the weakly consistent *orthogonal projection* technique [76], which adds artificial diffusion only in the regions where sharp gradients are present. The orthogonal projection technique makes the added diffusion proportional to the projection of the gradient of the solution onto the space defined to be orthogonal to the finite element space, that is to say, we take the artificial viscosity as

$$\nu_{\text{SC}} = \left(\frac{1}{2}C_a |\boldsymbol{u}| h\right) \frac{\left|\boldsymbol{P}_h^{\perp}(\nabla \boldsymbol{m}_h)\right|}{|\nabla \boldsymbol{m}_h|} \quad \text{if} \quad |\nabla \boldsymbol{m}_h| \neq 0, \quad \nu_{\text{SC}} = 0 \quad \text{otherwise.} \tag{3.68}$$

The norm of the velocity $|\boldsymbol{u}|$ gives dimensional consistency to the orthogonal projection based calculation. In this method, we calculate the artificial thermal diffusivity as

$$\alpha_{\text{SC}} = \left(\frac{1}{2}C_a |\boldsymbol{u}| h\right) \frac{\left|\boldsymbol{P}_h^{\perp}(\nabla e_{\text{tot},h})\right|}{|\nabla e_{\text{tot},h}|} \quad \text{if} \quad |\nabla e_{\text{tot},h}| \neq 0, \quad \alpha_{\text{SC}} = 0 \quad \text{otherwise.} \tag{3.69}$$

The value used in this work for the algorithmic constant is $C_a = 0.8$ for both residual and orthogonal projection methods (see Section 5 for the discussion about this value).

In practice, the way to introduce the added numerical diffusion into the diffusive Galerkin term is to compute a modified viscous stress tensor $\check{\boldsymbol{\tau}} = [\tau_{ij}]$ and heat flux vector $\check{\boldsymbol{q}} = [q_i]$. Within the modification of those tensors, we propose two distinct ways which are explained below. Figure 3.1 presents the main idea of how the artificial diffusion is added by each method.

The first method is the so-called *isotropic* method, which is based on the addition of the artificial diffusion into all the components of the viscous stress tensor and heat flux vector, that is,

$$\check{\tau}_{ij} = \left(1 + \frac{\rho \nu_{\text{SC}}}{\mu}\right) \tau_{ij} \quad \text{and} \quad \check{q}_i = \left(1 + \frac{\rho c_v \alpha_{\text{SC}}}{\lambda}\right) q_i. \tag{3.70}$$

The second method that we propose is to add the numerical diffusion in an *anisotropic* fashion. The anisotropic diffusion method consists in the addition of the artificial diffusion into the streamline direction ν_{ST} and α_{ST}, excluding the already incorporated VMS stabilization diffusion quantity, which we roughly estimate as

$$\nu_{\text{SG}} = \tau_m \left|\boldsymbol{u}\right|^2 \quad \text{and} \quad \alpha_{\text{SG}} = \tau_{e_{\text{tot}}} \left|\boldsymbol{u}\right|^2, \tag{3.71}$$

but, not decreasing the stabilization,

$$\nu_{\text{ST}} = \max\left(0, \nu_{\text{SC}} - \nu_{\text{SG}}\right) \quad \text{and} \quad \alpha_{\text{ST}} = \max\left(0, \alpha_{\text{SC}} - \alpha_{\text{SG}}\right). \tag{3.72}$$

Moreover, as explained in [86], the shock capturing artificial diffusion should be introduced in the crosswind direction. This is done by adding the artificial diffusion with anisotropic tensors.

In order to introduce the artificial thermal diffusion into the heat flux vector, we use an anisotropic second order tensor of the form

$$\check{q}_i = \left(\delta_{ij} + \frac{\rho c_v \alpha_{\text{SC}}}{\lambda} O_{ij} + \frac{\rho c_v \alpha_{\text{ST}}}{\lambda} S_{ij}\right) q_j, \tag{3.73}$$

where the anisotropic tensor is defined in terms of the second order projector into the streamline direction $S_{ij} = \frac{m_i m_j}{|\boldsymbol{m}|^2}$, and in terms of the orthogonal projector $O_{ij} = \delta_{ij} - S_{ij}$. We also include the artificial viscosity into the viscous stress tensor using a fourth order anisotropic tensor,

$$\check{\tau}_{ij} = \left(\delta_{ij}\delta_{kl} + \frac{\rho \nu_{\text{SC}}}{\mu} O_{ijkl} + \frac{\rho \nu_{\text{ST}}}{\mu} S_{ijkl}\right) \tau_{kl}. \tag{3.74}$$

Here $\boldsymbol{I} = [\delta_{ij}\delta_{kl}]$ stands for the fourth order identity tensor. We exploit the symmetric property of the viscous stress tensor in order to construct the fourth order orthogonal $\boldsymbol{O} = [O_{ijkl}]$ and streamline $\boldsymbol{S} = [S_{ijkl}]$ tensors. First, we write the viscous stress tensor using Voigt's notation for the most general three-dimensional case,

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix} \equiv \begin{bmatrix} \tau_1 & \tau_6 & \tau_5 \\ \tau_6 & \tau_2 & \tau_4 \\ \tau_5 & \tau_4 & \tau_3 \end{bmatrix} \rightarrow \boldsymbol{\tau} = \left(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\right)^\top. \tag{3.75}$$

Then, after some algebraic operations the 9 independent elements of the fourth order orthotropic streamline tensor can be written using Voigt's notation as follows:

$$\boldsymbol{S} = \begin{bmatrix} \frac{m_1 m_1}{|\boldsymbol{m}|^2} & \frac{m_1 m_2}{|\boldsymbol{m}|^2} & \frac{m_1 m_3}{|\boldsymbol{m}|^2} & 0 & 0 & 0 \\ \frac{m_1 m_2}{|\boldsymbol{m}|^2} & \frac{m_2 m_2}{|\boldsymbol{m}|^2} & \frac{m_2 m_3}{|\boldsymbol{m}|^2} & 0 & 0 & 0 \\ \frac{m_1 m_3}{|\boldsymbol{m}|^2} & \frac{m_2 m_3}{|\boldsymbol{m}|^2} & \frac{m_3 m_3}{|\boldsymbol{m}|^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{m_2 m_3}{|\boldsymbol{m}|^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{m_1 m_3}{|\boldsymbol{m}|^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{m_1 m_2}{|\boldsymbol{m}|^2} \end{bmatrix}. \tag{3.76}$$

The orthogonal projection is calculated according to the fourth order definition $O_{ijkl} = \delta_{ij}\delta_{kl} - S_{ijkl}$ and is presented below using Voigt's notation:

$$\boldsymbol{O} = \begin{bmatrix} 1 - \frac{m_1 m_1}{|\boldsymbol{m}|^2} & -\frac{m_1 m_2}{|\boldsymbol{m}|^2} & -\frac{m_1 m_3}{|\boldsymbol{m}|^2} & 0 & 0 & 0 \\ -\frac{m_1 m_2}{|\boldsymbol{m}|^2} & 1 - \frac{m_2 m_2}{|\boldsymbol{m}|^2} & -\frac{m_2 m_3}{|\boldsymbol{m}|^2} & 0 & 0 & 0 \\ -\frac{m_1 m_3}{|\boldsymbol{m}|^2} & -\frac{m_2 m_3}{|\boldsymbol{m}|^2} & 1 - \frac{m_3 m_3}{|\boldsymbol{m}|^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 - \frac{m_2 m_3}{|\boldsymbol{m}|^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 - \frac{m_1 m_3}{|\boldsymbol{m}|^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 - \frac{m_1 m_2}{|\boldsymbol{m}|^2} \end{bmatrix}. \tag{3.77}$$

3.3.5 Explicit time integration

At this point, we have described the space discrete stabilized finite element formulation. Let us now comment how we discretize in time.

We partition the time interval $(0, t_f)$ in a sequence of discrete time steps $0 = t^0 < t^1 < ... < t^N = t_f$, with $\delta t > 0$ the time step-size defining $t^{n+1} = t^n + \delta t$ for $n = 0, 1, 2, ..., N$. We also partition each time step into intermediate stages $t^n < t_1 < t_2 < ... < t^{n+1}$, S being the number of stages, which are considered to have a constant size for simplicity. Let us use the superscript to denote the time step counter and the subscript to denote the intermediate stage counter.

Let us describe the transient term integration of the finite element problem (3.23). After assembling the elemental contributions of the space discrete stabilized finite element problem, the discrete matrix form of the transient problem can be written as

$$\mathsf{M}\dot{\mathsf{U}} = \mathsf{R}\left(\mathsf{U}\right), \tag{3.78}$$

where U is the array that contains the nodal unknowns, M is the mass matrix, and $\mathsf{R}\left(\mathsf{U}\right)$ is the residual of the discrete stabilized finite element problem. We use an explicit time integration scheme in order to integrate the temporal derivatives, more precisely, we implement the family of explicit Runge-Kutta (RK) methods. For any number of stages, the finite element solution at t^{n+1} is given by the quadrature:

$$\mathsf{U}^{n+1} = \mathsf{U}^n + \mathsf{M}^{-1}\delta t \sum_{i=1}^{S} \mathsf{b}_i \mathsf{K}_i, \tag{3.79}$$

where M^{-1} is the inverted mass matrix and K_i (for $1 \leq i \leq S$) are stage increments based on the evaluation of the space discrete stabilized finite element residual,

$$\mathsf{K}_i = \mathsf{R}\left(\mathsf{U}^n + \mathsf{M}^{-1}\delta t \sum_{j=1}^{i-1} \mathsf{a}_{ij}\mathsf{K}_j\right). \tag{3.80}$$

Moreover, we calculate the time integration method for the temporal derivative of the dynamic subscales exploiting the explicit RK scheme adopted for the finite element equation, that is, we solve the subscales at time t^{n+1} with

$$\widetilde{\boldsymbol{U}}^{n+1} = \widetilde{\boldsymbol{U}}^n + \delta t \sum_{i=1}^{S} \mathsf{b}_i \widetilde{\boldsymbol{K}}_i. \tag{3.81}$$

The subscales at the intermediate stage t_i, which are also incorporated in the time integration stage of the finite element equation, are calculated as follows,

$$\widetilde{\boldsymbol{U}}^{t_i} = \widetilde{\boldsymbol{U}}^{n} + \delta t \sum_{j=1}^{i-1} \mathsf{a}_{ij} \widetilde{\boldsymbol{K}}_j, \tag{3.82}$$

where $1 \leq j < i \leq S$. The intermediate increments of the subscales transient equation are evaluated explicitly at t_j with the subscales residual

$$\widetilde{\boldsymbol{K}}_j = \widetilde{\boldsymbol{P}} \left[\boldsymbol{R} \left(\boldsymbol{U}^{t_j}; \boldsymbol{U}_h^{t_j} \right) \right] - \boldsymbol{\tau}^{-1} \left(\boldsymbol{U}^{t_j} \right) \widetilde{\boldsymbol{U}}^{t_j}. \tag{3.83}$$

In the previous expressions, the non-linear subscales at the intermediate stage j are considered as $\boldsymbol{U}^{t_j} = \boldsymbol{U}_h^{t_j} + \widetilde{\boldsymbol{U}}^{t_j}$. As a final remark, we approximate the calculation of the non-linear subscales at intermediate stage j using the resulting values at the previous stage when the subscales are considered quasi-static, that is, $\boldsymbol{U}^{t_j} = \boldsymbol{U}_h^{t_j} + \widetilde{\boldsymbol{U}}^{t_{j-1}}$. Expresions (3.79) - (3.82) depend on the definition of the coefficients a_{ij} (for $1 \leq j < i \leq S$) and b_i (for $1 \leq i \leq S$) for a specific RK method. In particular, we have implemented a four-stages explicit RK method with coefficients:

$$\begin{array}{cccc} \mathsf{a}_{31} = \mathsf{a}_{41} = \mathsf{a}_{42} = 0, & \mathsf{a}_{21} = \frac{1}{2}, & \mathsf{a}_{32} = \frac{1}{2}, & \mathsf{a}_{43} = 1, \\ \mathsf{b}_1 = \frac{1}{6}, & \mathsf{b}_2 = \frac{1}{3}, & \mathsf{b}_3 = \frac{1}{3}, & \mathsf{b}_4 = \frac{1}{6}, \end{array}$$

which is fourth order in accuracy. This explicit method is subject to the Courant-Friedrich-Levi (CFL) stability criterion for hyperbolic systems, which implies

$$\frac{\delta t \left(|\boldsymbol{u}| + c \right)}{h} \leq 1, \tag{3.84}$$

as a necessary condition for numerical stability.

3.4 Numerical examples

In this section some numerical examples are presented. First, the three-dimensional lid-driven cavity problem is solved in order to demonstrate the multidimensional features of the proposed formulation. We also test including the body force and heat source into this problem. Then, we solve a periodic subsonic flow past a cylinder in order to test the VMS particular features. A convergence analysis is proposed for linear triangular elements. Finally, some compressible flow examples that exhibit supersonic shocks are solved in order to compare the proposed shock-capturing methods. The flow is considered as an ideal gas in all cases, with ratio of specific heats $\gamma = 1.4$ and physical properties $c_p = 1.010$ kJ/(kg K) and $c_v = 0.718$ kJ/(kg K).

3.4.1 Three-dimensional lid-driven cavity

The three-dimensional lid-driven cavity problem is a widely used benchmark in computational fluid dynamics. The problem domain is a prismatic cavity $[0, L] \times [0, L] \times [0, L]$, with $L = 1$ m. The upper wall (x_1, x_2, L) m is constantly moving at a fixed velocity of $(1, 0, 0)$ m/s,

and the density is 1 kg/m^3. The temperature is also set over this boundary depending on the compressibility regime. A no-slip condition for velocity, an adiabatic condition for energy, and an impermeable condition for mass are set over the other walls.

In this problem, we aim to test the multidimensional formulation including the convective, diffusive and source contributions into the compressible equations. For convenience, we include the orthogonal projection to the finite element residual, the time dependence, and the non-linearity of subscales. Comparissons are made by evaluating three Reynolds numbers, specifically Re $= 100$, Re $= 400$ and Re $= 1000$. The Prandtl number is fixed to $Pr = 0.71$ for all cases. Flow conditions are laminar in all cases, and simulations are run until the stationary state is reached. The mesh is constituted by 64000 hexahedral elements in a structured $40 \times 40 \times 40$ homogeneous distribution.

We first solve this problem without body force and heat source. This case is intended to compare our results with previous bench-marked solutions. To our knowledge, only incompressible results have been published in the case of a three dimensional cubed driven cavity (for example in [87]). Hence, in order to approximate the incompressibility condition the Mach number is fixed to a subsonic regime M $= 0.1$, and the temperature to $\theta_w = 0.2446$ K. Figure 3.2 shows the velocity profiles along the center-lines of the cavity.

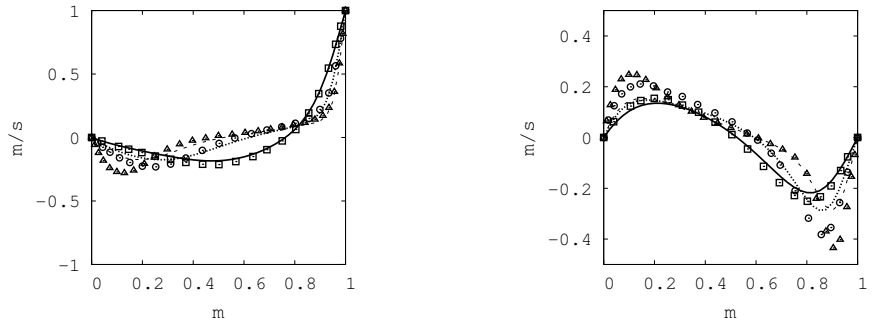
The compressible results are qualitatively not in good agreement with previous numerical and experimental investigations, mainly because the comparison is made with the incompressible results obtained in [87]. Compressible results are accurate in the location of the maximum and minimum velocities. However, the magnitude of the extreme values of the velocity differ substantially. This difference is explained by the lack of accuracy of the compressible Navier-Stokes equations written in conservation variables at the incompressible limit.

Nevertheless, we aim to test the stabilized variational formulation, especially when body forces and heat sources are included. For academic purposes, we solve this numerical example by setting the body force to $\boldsymbol{f} = (0, 0, -0.1)$ m/s^2. The heat source is also maintained homogeneous and constant $r = 1$ m^2/s^3. For this case, compressibility is fixed by setting M $= 1$ and therefore, the temperature at the wall is 0.0024 K. Figure 3.3 shows the velocity and temperature profiles along the centerlines of the cavity. We conclude that the numerical behavior of the proposed stabilized formulation including source terms appropriately approximates the physics: smooth solutions are obtained for all Reynolds numbers. Some dissipation can be observed for the location and magnitude of the extreme velocities when compared to the solution without including sources. Moreover, temperature results prove the correct solution for the heat and momentum equations.

The previous results assure a correct approximation given by the variational multiscale finite element formulation. That is, we guarantee a correct definition for the matrix τ of stabilization parameters, but also demonstrate the multi-dimensional capability of the variational formulation. Because there is not a dynamical behavior in this laminar problem, we solve some other unsteady numerical examples in order to test the time integration of the finite element scales and the subscales.

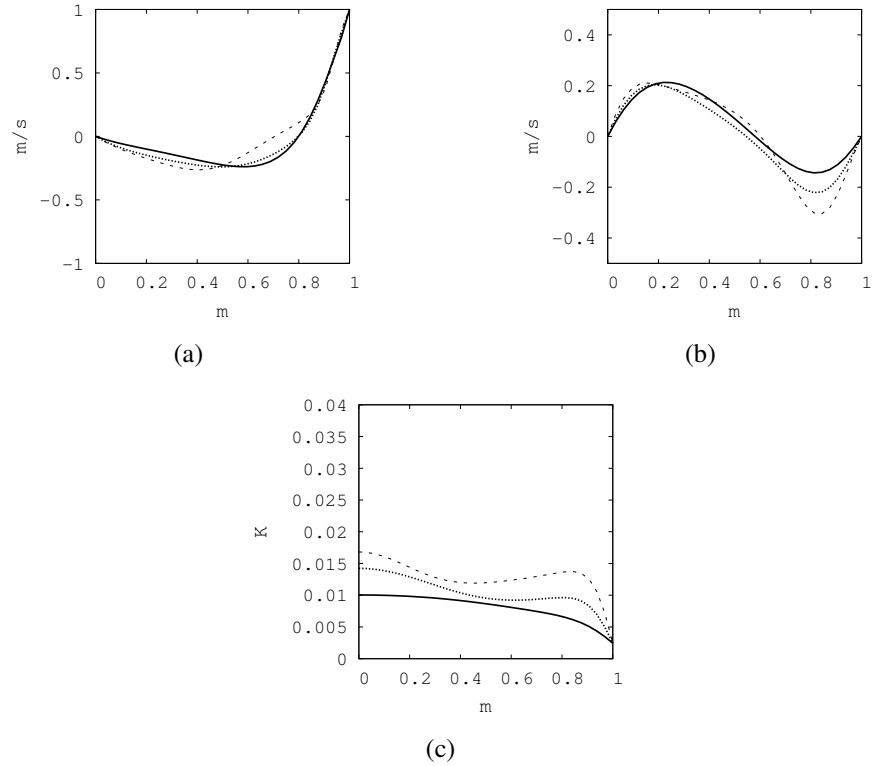
3.4.2 Subsonic flow past a cylinder

The problem is defined by a cylinder infinitely long in the axial direction, which is immersed in the compressible fluid and is subject to a uniform incident flow. The flow is solved as a



(a) (b)

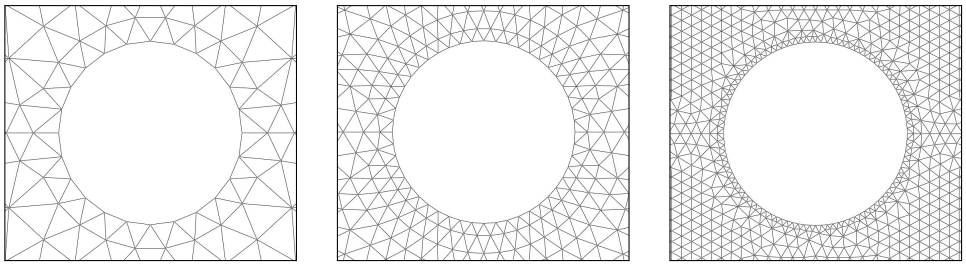
Figure 3.2: Three-dimensional lid-driven cavity results. Compressible problem without source terms: (a) velocity profile along the centerline $(0.5, 0.5, x_3)$; and (b) velocity profile along the centerline $(x_1, 0.5, 0.5)$. The Reynolds numbers are Re $= 100$ (solid line), Re $= 400$ (dotted line), and Re $= 1000$ (dashed line). The symbols $\square$ (Re $= 100$), $\bigcirc$ (Re $= 400$), $\triangle$ (Re $= 1000$) represent the extracted results from [87].



(a) (b)

(c)

Figure 3.3: Three-dimensional lid-driven cavity results. Compressible problem including source terms: (a) velocity profile along the centerline $(0.5, 0.5, x_3)$; (b) velocity profile along the centerline $(x_1, 0.5, 0.5)$; and (c) temperature profile along the centerline $(0.5, 0.5, x_3)$. The Reynolds numbers are Re $= 100$ (solid line), Re $= 400$ (dotted line), and Re $= 1000$ (dashed line).



(a) (b) (c)

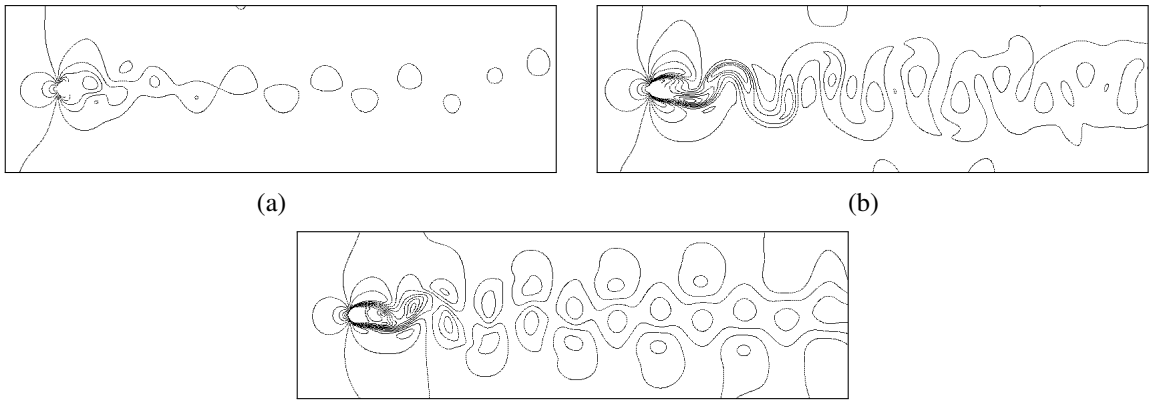
Figure 3.4: Detail of three unstructured P_1 meshes used for the analysis of flow past a cylinder: (a) mesh $h = 0.05$ m; (b) mesh $h = 0.025$ m; (c) mesh $h = 0.0125$ m.

two-dimensional flow, and it is intended to test the proposed variational multiscale stabilized formulations in an unsteady problem. We restrict the flow to the subsonic range in order to avoid the need of the shock capturing techniques.

The problem domain is $[0, L] \times [-H/2, H/2]$ with $L = 7$ m and $H = 2.4$ m, with the cylinder diameter $D = 0.2$ m centered at point $(0.8, 0)$ m. Boundary conditions are set as follows. The flow is injected from the left wall with uniform constant velocity $\boldsymbol{u}_{in} = (1, 0)$ m/s, temperature $\theta_{in} = 9.73 \times 10^{-3}$ K, and density $\rho_{in} = 1$ kg/m^3. Over the adiabatic upper and lower walls symmetric boundary conditions are imposed, with the vertical component of velocity and the horizontal component of the stress imposed to zero. On the cylinder surface Γ_w, a no-slip condition for velocity is imposed. Over the outflow wall, density is set to $\rho_{out} = 1$ kg/m^3; zero stress and adiabatic conditions are considered for the subsonic flow. The Mach number in this case is M=0.5. The viscosity and heat conductivity are $\mu = 0.002$ kg/(m s) and $\lambda = 2.8676$ kJ/(m K s), respectively, giving a Reynolds number of Re $= 100$

Three different unstructured meshes composed of P_1 elements are used to solve this problem. These meshes, shown in Fig. 3.4, are generated defining the mesh size $h_k = h_0/2^k$ in terms of successive divisions k. The mesh of Fig. 3.4 (a) is defined with $h_0 = 0.05$ m and 14008 elements; the mesh of Fig. 3.4 (b) is defined as half the size of the mesh of Fig. 3.4 (a), that is, $h_1 = 0.025$ m with 54884 elements; furthermore, the mesh of Fig. 3.4 (c) is defined as half the size of the mesh of Fig. 3.4 (b) as $h_2 = 0.0125$ m, with 217162 elements. The time step size of the explicit fourth order Runge-Kutta time integration scheme is set to a constant $t = 0.001$ s value for all simulations. This time step size is for all simulations lower than the explicit restriction given by the problem.

We use this numerical example to compare the proposed VMS stabilized finite element formulation. That is, we evaluate the numerical behavior of subscales, which can be defined orthogonal to the finite element residual space, dynamic, or can be incorporated in all the nonlinear terms. The following notation is used in order to clarify the different subscales definition in the numerical example. Defining the space where the subscales live as orthogonal to the finite element residual space is denoted by the "O" letter, otherwise defining it as the finite element residual space is denoted as "R". The letter "D" is used for dynamic subscales, and for the quasi-static definition of the subscales, the letter "Q" is employed. Finally, if the subscales are included in the non-linear terms the letter "N" is used, and if they are not considered, the letter "L" is used.



(a) (b)

(c)

Figure 3.5: Instantaneous contour fields of the flow past a cylinder: (a) Pressure, (b) temperature and (c) velocity magnitude calculated with the mesh of Fig. 3.4 (c). Results are obtained with the subscales defined as ODN.

Smooth result fields are obtained for the VMS formulation using the meshes presented above. The periodic transient solution for the Re $= 100$ case, which has been reported previously in [67, 79], is obtained in correspondence with the wake oscillations. All the simulations are run until the mean values of the forces exerted by the fluid over the cylinder reach a statistically stationary state. Due to the temporal dependence of the solution, we restrict the presentation of contour results to a given time step. Pressure, temperature, and velocity magnitude contours at a given instant of the flow are shown in Fig. 3.5. These results are obtained with the ODN method using the mesh of Fig. 3.4 (c). Solutions obtained with other subscales definitions, that we do not present for conciseness, give qualitatively similar contour results. Instead, some calculations are performed in order to quantify the numerical results obtained with the proposed formulations. We calculate the non-dimensional Strouhal number,

$$\mathrm{St} = \frac{\mathrm{f}_f D}{|\boldsymbol{u}_{in}|}, \tag{3.85}$$

based on the cylinder diameter D and the frequency of the wake oscillations f_f. In addition, the non-dimensional drag and lift coefficients,

$$\mathrm{C_d} = -\frac{F_{\Gamma_w,1}}{\frac{1}{2}\rho_{in}\left|\boldsymbol{u}_{in}\right|^2 D}, \qquad \mathrm{C_l} = \frac{F_{\Gamma_w,2}}{\frac{1}{2}\rho_{in}\left|\boldsymbol{u}_{in}\right|^2 D}, \tag{3.86}$$

are calculated using the exerted force of the fluid over the cylinder,

$$\boldsymbol{F}_{\Gamma_w} = \int_{\Gamma_w} -\left(-p\boldsymbol{I} + 2\mu\nabla^S\boldsymbol{u} - \frac{2}{3}\mu\left(\nabla\cdot\boldsymbol{u}\right)\boldsymbol{I}\right)\cdot\boldsymbol{n} \quad d\Gamma. \tag{3.87}$$

The time history of the non-dimensional drag an lift coefficients are shown in Fig. 3.6 for the solution obtained with the ODN method, and the mesh of Fig. 3.4 (c). We propose to calculate the convergence of the proposed formulations by using the discrete L^p-norm. Since the flow over a cylinder problem does not possess an analytical solution, the following convergence

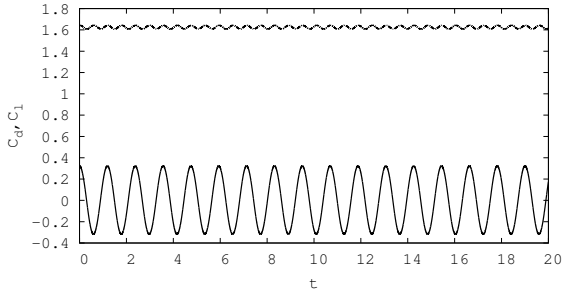


Figure 3.6: Time history of drag and lift coefficients for the subsonic flow past a cylinder. Lift (solid line) and drag (dashed line) results are calculated using the mesh of Fig. 3.4 (c), and defining the subscales as ODN.

analysis is proposed. For a fixed time t, the L^p-norm of a discrete solution $u_h(\boldsymbol{x}, h, \delta t)$ can be written as the composition of the L^p-norm of the analytical solution $u(\boldsymbol{x})$ plus some time and spatial integration dependent terms. If we set a constant time step size and the mesh size is defined in terms of consecutive divisions, we can write those discrete L^p-norms of the discrete solution as,

$$\|u_h(\boldsymbol{x}, h_k, \delta t)\|_p \approx \|u(\boldsymbol{x})\|_p + c_1(\delta t)^r + c_2(h_k)^q, \qquad k = 0, 1, 2. \tag{3.88}$$

Consequently, the calculation of the quotient between consecutive differences of the L^p-norm,

$$\mathcal{Q} = \frac{|\|u_h(\boldsymbol{x}, h_0, \delta t)\|_p - \|u_h(\boldsymbol{x}, h_1, \delta t)\|_p|}{|\|u_h(\boldsymbol{x}, h_1, \delta t)\|_p - \|u_h(\boldsymbol{x}, h_2, \delta t)\|_p|}, \tag{3.89}$$

and the substitution of mesh size definitions into the previous quotient,

$$\mathcal{Q} = \frac{c_2 h_0^q \left(1 - \frac{1}{2^q}\right)}{c_2 h_0^q \left(\frac{1}{2^q} - \frac{1}{2^{2q}}\right)} = \frac{2^{2q} - 2^q}{2^q - 1} = 2^q, \tag{3.90}$$

makes possible to get an expression for the convergence order q of the spatial integration term,

$$\log_2 \mathcal{Q} = q. \tag{3.91}$$

Time L^∞-norm of Strouhal numbers and time average of drag coefficients are presented in Tables (3.1) - (3.2), respectively. The convergence order q is also presented in these tables for the calculated values. It can be observed that both the time L^∞-norm of Strouhal numbers, and time average of drag coefficients converge to a greater value as the mesh size is refined. Our results agree with the Strouhal number values $[0.167, 0.17]$ of the vortex shedding, previously reported in [79]. This characteristic can be used together with the convergence order in order to compare the stabilization methods.

Defining the subscales as orthogonal to the finite element space gives the most well-approximated solutions. The results given by the calculated norms, both for the Strouhal number and for the drag coefficient indicate a less dissipative behavior of this method, compared

Table 3.1: Subsonic flow past a cylinder results. Time L^∞-norm of Strouhal number.

Mesh	SUPG	RQL	RQN	RDL	RDN	OQL	OQN	ODL	ODN
$h = 0.05$	0.1518	0.1527	0.1515	0.1539	0.1523	0.1523	0.1516	0.1523	0.1515
$h = 0.025$	0.1646	0.1647	0.1642	0.1654	0.1644	0.1652	0.1650	0.1652	0.1650
$h = 0.0125$	0.1673	0.1675	0.1670	0.1680	0.1675	0.1684	0.1683	0.1684	0.1683
q	2.2193	2.1142	2.1384	2.1192	2.0035	2.0190	2.0075	2.0190	2.0193

Table 3.2: Subsonic flow past a cylinder results. Time average of non-dimensional drag coefficient.

Mesh	SUPG	RQL	RQN	RDL	RDN	OQL	OQN	ODL	ODN
$h = 0.05$	1.3615	1.3853	1.3377	1.4008	1.3536	1.4108	1.3938	1.4106	1.3935
$h = 0.025$	1.5160	1.5317	1.5044	1.5407	1.5112	1.5321	1.5270	1.5321	1.5267
$h = 0.0125$	1.6053	1.6140	1.6013	1.6191	1.6052	1.6277	1.6257	1.6277	1.6257
q	0.7908	0.8309	0.7826	0.8354	0.7455	0.3434	0.4324	0.3458	0.4280

to the finite element residual definition for the space of subscales. However, the convergence order is lower for the orthogonal subscales. Moreover, including the time-dependency of the subscales improves the accuracy of the subscales defined in the finite element residual space. The solution obtained by RDL method gives accurate results together with a high convergence order. This method is more accurate in the drag results than the former stabilization SUPG method, and its convergence order is higher.

On the contrary, the non-linear contribution of subscales reduces a little the accuracy for both the subscales defined in the finite element residual space and for the orthogonal subscales. The combination between orthogonal, non-linear and dynamic subscales gives a highly accurate method. Nonetheless, the convergence order for this method is not the highest for the element sizes considered.

3.4.3 Supersonic inviscid shock reflection

We evaluate next the shock capturing methods by solving the inviscid shock reflection problem at M $= 2.9$. The supersonic reflected shock problem is widely used in order to test the compressible flow solvers. We also exploit this numerical example so that high order approximations are tested. For solving this problem we use the VMS formulation defining the space of subscales orthogonal to the finite element residual space, and considering the time dependence and non-linear contribution of subscales.

The problem domain is $[0, L] \times [-H/2, H/2]$, with $L = 4.1$ m and $H = 0.5$ m. The flow is injected from the left and upper walls. Over the inlet left wall the Mach number is set to M $= 3$, hence, fixed values of velocity $(2.9, 0)$ m/s, density 1 kg/m^3, and temperature 0.00247 K are set. Over the upper inlet wall the Mach number is M $= 2.378$, thus, fixed values of velocity $(2.6193, -0.5063)$ m/s, density 1.7 kg/m^3, and temperature 0.00311 K are also set. A

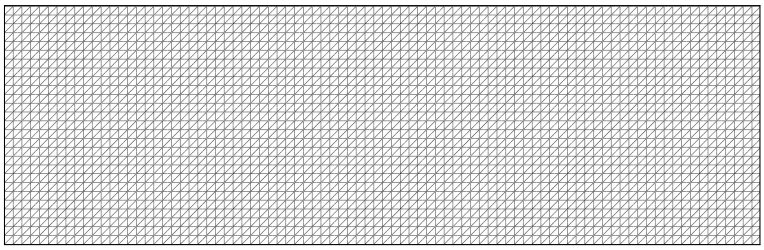


Figure 3.7: Detail of the structured mesh configuration used for the inviscid shock reflection.

(a)

(b)

Figure 3.8: Inviscid shock reflection results: (a) density contour and (b) velocity magnitude contour. Solution is obtained using the anisotropic orthogonal projection-based shock capturing method.

slip condition for the velocity and an adiabatic condition for energy are set to the lower wall. Over the outflow wall, no conditions need to be imposed because the flow is supersonic. The values of viscosity and conductivity are zero.

We use P_2 elements in order to solve a high order approximation. Nonetheless, P_1 elements are used to compare the shock capturing techniques. In both cases, the finite element mesh is composed by 13120 triangular elements distributed in a structured non-symmetric fashion. This structured configuration is presented in Fig. 3.7. All simulations are run until the steady state is reached considering the time-step size given by the stability condition (3.3.5). Adequate physical solutions are found using both the residual based and the orthogonal projection based shock capturing methods. Overshoots are smoothed according to the benchmarked solutions in [73, 79, 82, 83]. Figure 3.8 shows the steady state contours for density and velocity magnitude solved with the anisotropic orthogonal projection-based shock capturing method and the mesh composed by P_2 elements.

The analysis of the solutions obtained by the isotropic and anisotropic, as well as the residual-based and the orthogonal projection-based shock capturing methods, is made by comparing the horizontal component of the velocity solution at $(x_1, 0.25)$ m. Figure 3.9 plots the

solution obtained by the shock capturing methods for two distinct definitions of the algorithmic constant $C_a = 0.8$ and $C_a = 0.2$. These results are obtained with the mesh composed by P_1 elements. In Fig. 3.9 the analytical solution presented in [79], together with the solution found without including the shock capturing method, are also plotted.

A correct solution to the problem is found with the anisotropic method and the $C_a = 0.8$ value. The anisotropic residual-based shock capturing method gives an accurate and less diffusive solution than previous solutions obtained using the same mesh size, like the ones in [82, 83]. Because the anisotropic method adds the artificial diffusivity only in the crosswind direction, this method is less non-linear. In contrast, due to the high non-linearities in the problem introduced by the artificial diffusion of the isotropic method, this method only gives a stable solution with a lower $C_a = 0.2$ coefficient. With the lower $C_a = 0.2$ value, some overshoots in the solution obtained by both isotropic and anisotropic methods can be observed.

The orthogonal projection-based shock capturing method gives more diffusive results near the solution gradients when compared to the residual-based method. However, the overall solution obtained by this technique is adequate and gives accurate results, when compared to the solution without a shock capturing method.

3.4.4 Supersonic flow past a cylinder

The supersonic flow past a cylinder problem at Re $= 2000$ and M $= 2$ is presented as a benchmark problem in order to test the shock capturing techniques in a viscid supersonic case. The description of the problem is similar to the subsonic flow past a cylinder presented in Subsection 3.4.2. In this supersonic case, an adiabatic condition is fixed over the cylinder surface and the physical conditions are set to $\mu = 0.0001$ kg/(m s) and $\lambda = 0.14338$ kJ/(m K s).

For this problem, we aim to test the ODN-VMS formulation together with the anisotropic residual-based shock capturing technique, that gives the best approximation for the inviscid reflected shock example. The finite element mesh consists of an unstructured mesh composed by 31288 linear triangular elements. Smaller elements are used near the wall cylinder, whereas the mesh is coarser in the rest of the domain. The mesh size was fixed to $h = 0.005$ m in the finer region near the cylinder boundary. Due to the viscid condition of the problem, the solution is reached by setting the time-step size lower than the explicit scheme stability condition (3.3.5). Figure 3.10 shows the solution for the supersonic flow over a cylinder solved using the ODN stabilization formulation, together with anisotropic residual based shock capturing method.

Comparing this solution to the ones obtained in previous formulations, the fields are almost identical. The solution differs from the steady state solution obtained in [73] in the sense that no Sutherland law is used for viscosity in this example. Moreover, the shock upstream of the cylinder is formed as extensively described in the literature, as in [79]. The anisotropic shock capturing method gives an accurate solution for the thin shock layer of the supersonic expansion within two or three elements. The detached shock wave presents the oblique solution for the compressible problem. It also gives correct results where gradients of the solution are not too sharp, such as for the weak tail shock that is formed in the wake structure.

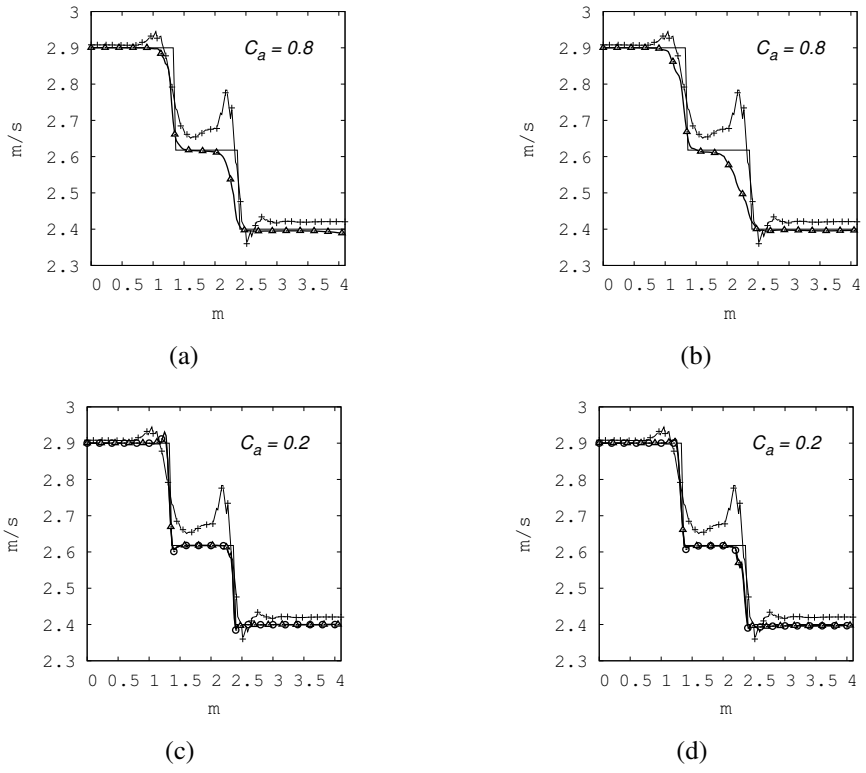
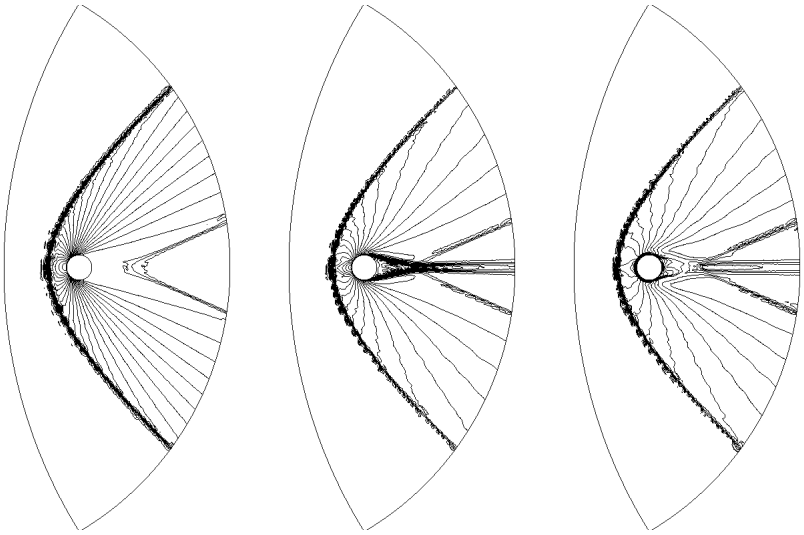


Figure 3.9: Inviscid shock reflection solution. Horizontal velocity component at $(x_1, 0.25)$ m: Residual-based method used in (a) and (c); Orthogonal projection-based method used in (b) and (d); The solid line represents the analytical solution presented in [79]. The solution obtained without shock capturing method is plotted using a solid marked line. Solid lines with symbols ◯ (isotropic), and △ (anisotropic) represent the solution obtained using shock capturing methods.



(a) (b) (c)

Figure 3.10: Supersonic flow past a cylinder results: (a) pressure contour; (b) velocity magnitude contour; (c) temperature contour. The solution is obtained using the ODN stabilization formulation together with the anisotropic residual based shock capturing method.

3.5 Conclusions

In this chapter, we have applied the VMS concept for stabilizing the compressible Navier-Stokes equations. We have presented a systematic way to design the matrix of algorithmic parameters using a Fourier analysis. We have implemented the orthogonal to the finite element space subscales, the dynamic subscales, and the non-linear tracking of subscales. The formulation has been implemented in an explicit fashion using a fourth order Runge-Kutta scheme. The subsonic three-dimensional lid-driven cavity and the flow past a cylinder problems have been used to test the numerical behavior of the VMS formulation. Adequate physical solutions have been obtained, and convergence has been demonstrated for linear elements. It has been found that including the orthogonal, dynamic, and the non-linear subscales improve the accuracy of the compressible formulation. The results indicate a less dissipative behavior of the orthogonal subscales compared to the finite element residual definition for the space of subscales.

Local instabilities have been aborded with shock capturing methods. The anisotropic shock capturing method has been formulated in contrast to the isotropic method, giving better solutions for the supersonic examples. Because the anisotropic method adds the artificial diffusivity only in the crosswind direction, the non-linearity introduced by the shock capturing method has less effect on the convergence behavior to the steady state. In order to calculate the artificial diffusion, we also proposed to use an orthogonal projection-based method, instead of the residual-based definition, with similar results.

Chapter 4

Solution of low Mach number aeroacoustic flows using a variational multi-scale finite element formulation of the compressible Navier-Stokes equations written in primitive variables

In this chapter, we solve the compressible Navier-Stokes equations written in primitive variables in order to simulate low Mach number aeroacoustic flows. We develop a variational multi-scale formulation to stabilize the finite element discretization by including the orthogonal, dynamic and non-linear subscales, together with an implicit scheme for advancing in time. Three additional features define the proposed numerical scheme: the splitting of the pressure and temperature variables into a relative and a reference part, the definition of the matrix of stabilization parameters in terms of a modified velocity that accounts for the local compressibility, and the approximation of the dynamic stabilization matrix for the time-dependent subscales. We also include a weak imposition of implicit non-reflecting boundary conditions in order to overcome the challenges that arise in the aeroacoustic simulations at low compressibility regimes.

4.1 Introduction

The increasing amount of computing resources available in scientific and industrial research has motivated the solution of realistic computational fluid dynamics applications. This is the case of aerodynamics and aeroacoustics; both described by a solution of the compressible Navier-Stokes equations that spans a wide range of scales. An illustrative application of this problem is the simulation of the human voice production inside the mouth, in which the solution of the relevant (small) scales of turbulence and sound waves must be achieved at low compressibility conditions. Indeed, compressible turbulent flows and sound waves propagation require an accurate computational method: the small fluctuating scales of sound need to be described, either by higher precision numerical schemes or by small mesh and time step

sizes. In any case, the number of operations required by the computational simulation is very demanding.

Another important requirement for such aeroacoustic problem is that the numerical approximation of the compressible Navier-Stokes equations at low compressibility conditions must be accurate, but many times those methods are not designed to work in these conditions. Most of the compressible flow solvers found in the literature exhibit a degradation in the solution accuracy when the free stream Mach number is progressively reduced. As explained in [88], most of the compressible flow solvers suffer a mismatch between the numerical and the continuous fluxes, and this is principally attributed to the broad difference in length and time scales of the solution. A possible approach for simulating aeroacoustic problems is to solve an aeroacoustic model together with the incompressible Navier-Stokes equations (e.g. reference [89]), but this leads directly to discarding the important physics related to the sound wave production, and consequently, this possibility is discarded in the present work. The type of methods that are intended to be suitable for either compressible or incompressible flows are the so-called *unified methods*. Among the most remarkable physical models, some modify the incompressible Navier-Stokes equations including extra term (like in the Boussinesq or Low Mach formulations described in [11], and references therein). There are also numerical approximations like [90–94], that aim to be valid at any compressibility regime. These unified approaches lead to more or less accurate descriptions of the fluid flow, but again, they cannot be used in wave propagation problems.

A formulation of the compressible Navier-Stokes equations that can be suitable for the low Mach number limit (or both for compressible and incompressible flows) is important for many reasons. One is that very low Mach numbers can coexist with regions where the flow compressibility becomes considerable. But also, several applications that are traditionally solved with incompressible formulations can be successfully handled by suitable compressible formulations. In this sense, it is important to remark that conservative variables (namely density, momentum, and total energy) are typically used in the compressible formulation, while primitive variables (pressure, velocity, and temperature) are used in the incompressible equations. Including density as a variable in the compressible problem (like in the conservative formulation presented in Chapter 3), results in singularities for the convective terms in the low Mach number limit. Possibilities restrain to primitive and entropic unknowns, which for certain regularity conditions can be shown to satisfy the incompressible equations in the zero Mach number limit. The entropy variable formulation assures a global entropy stability condition, but it is subject to the definition of the entropy function. For more details on entropy variable formulations we refer to [67, 95]. In the case of the compressible Navier-Stokes equations written in primitive variables, the formulation simplifies to the incompressible equations when the incompressible constraint is included (as explained in [96]), and consequently, it is well defined in the low Mach number limit.

Another particular aspect of the nearly incompressible aeroacoustic flows is that computational boundaries may cause artificial wave reflections related to the ingoing part of the sound waves. Ingoing waves may not only interfere with the acoustic signal relevant to the problem, but they can also produce numerical instabilities if the numerical method is not able to provide enough dissipation [97]. Several approaches to counteract spurious reflections have emerged in the computational aeroacoustics field, among the most popular are: the explicit damping of the compressible equations, the addition of an artificial counter signal, and the solution of

non-reflecting boundaries. Damping some terms of the compressible Navier-Stokes equations may be a robust approach that filters the solution in some local regions (called buffer zones), but the computational effort that is required to solve those extra terms is large, and it has led to other approaches, one of which has been to remove the computational domain that is not of prior interest and to solve a non-reflecting boundary condition. The recent literature on non-reflecting boundary conditions is too vast to survey here, but we mention the early work in [98] using asymptotic sets of non-reflecting conditions with pseudo-differential operators, the heuristic radiation arguments in [99], or the asymptotic expansion matching the linearized Euler solution in [100], and refer to the review article [101] for a deeper understanding. Most of the non-reflecting boundary methods attempt to damp the incident wave coming from the interior part of the domain.

As commented before, the small fluctuating scales of compressible turbulent flows and sound waves propagation require higher precision numerical schemes in both the temporal and the spatial discretization. This work is focused on the Finite Element Method (FEM) discretization. Among its advantages are the capability of using unstructured meshes to discretize complex geometries, and the possibility of developing efficient high order methods, which are two features highly desired in the Computational Fluid Dynamics (CFD) community. Nonetheless, when the Galerkin method is used to approximate this problem, which possesses non-symmetric operators, an unstable behavior of the solution might appear generated, for instance, by unresolved boundary layers. These local instabilities may arise when convection is dominant, and also due to the inf-sup stability condition that restrains the interpolation compatibility of the different variables. Stabilized numerical methods like the Petrov Galerkin Streamline Upwind (SUPG), and the methods that can be framed in the Variational Multi-Scale (VMS) concept, add a stabilization term to the Galerkin formulation and eliminate both the inf-sup and the convection restrictions. Stabilized finite element methods have been widely applied to both compressible and incompressible formulations, as shown in [17, 102], and references therein. See also [15] for a review of VMS methods in CFD. Because only subsonic flows are considered in this study, and more specifically, we restrict the attention to low Mach number flows, the extra control over the gradients of the solution given by discontinuity capturing operators is not required, and consequently, shock capturing techniques are not surveyed here.

In this context, we center the review to the well established VMS stabilization method. This approach splits the unknowns of the problem into a coarse-scale that belongs to the finite element space and a subgrid scale or subscale, which is the remainder. The main idea of the VMS formulation is to include the effect of the unresolved scales in order to stabilize the finite element equations. As a consequence, the correct design of the stabilization operator used to represent the subscales in terms of the resolved scales is crucial for the accuracy and stability of the discretization. Classical stabilized compressible flow formulations, including the compressible Euler equations, give inaccurate results near the low Mach number limit mainly because the standard definitions of the stabilization operator fail to provide adequate numerical diffusion, as demonstrated in the numerical examples of Chapter 3. In order to overcome this difficulty, nondiagonal definitions have been proposed (e.g. by [67]). Nonetheless, the diagonal definition can be used with few modifications: this has been the approach taken by [88, 103], in which dimensional analysis is used to design a compressible stabilization matrix that is suitable for both compressible and incompressible flows.

In the present chapter, the primitive variables compressible formulation is used because of its correspondence with the incompressible Navier-Stokes equations in the incompressible limit. Some other improvements are included in the formulation. One important aspect is the use of an implicit time integration scheme in order to avoid the inefficiency that arises from the aeroacoustic propagation in the low Mach number limit. In this sense, and depending on the system of units used for the solution of the problem, the transient problem can include very large quantities in the right-hand-side (RHS) of the continuity and energy equations, which degrade the iterative solution of the discrete linear system. From our perspective, this lack of scaling of the discrete linear system can be healed by splitting the primitive pressure and temperature unknowns into a relative and a reference part. The idea presented here is to change the primitive unknowns to the relative ones, but taking care for including the original unknowns in the non-linear terms of the problem. From the numerical point of view, this transformation not only scales the problem but also allows to implement iterative linear system solvers in the case of transient problems. We also incorporate some particular features of the VMS framework that we have applied in other flow problems (like in [76], for example). Since there are different ways to define the subscales, three different attributes are studied in this work. The first attribute is the definition of the space of the subscales, which can be either the space of finite element residuals or the space orthogonal to the finite element space [43]. The second attribute is the inclusion of the transient term of the subscale equation [46]. The third attribute is the inclusion of the subscales in all the non-linear terms of the problem [57]. In addition to these definitions, we design the stabilization operator in terms of the local compressibility of the flow, which, in the low Mach number limit, converges to the usual incompressible stabilization operator definition for the continuity and momentum equations. The proposed stabilization operator includes a physical parameter that defines the transition between the compressible and incompressible flow, hence, it combines the features of both stabilized formulations and is capable of describing the range of subsonic Mach numbers.

The focus of the present chapter is also to investigate the nearly incompressible applications of the compressible flow solver, which involve aeroacoustic applications. For doing so, we implement the characteristic decomposition of waves as a non-reflecting method over the boundaries and prescribe an implicit solution of this method in a weak sense by means of introducing some penalization terms to the compressible formulation. This decomposition of waves, introduced by Thompson in [104], together with the non-reflecting boundary conditions for the Navier–Stokes equations given by Poinsot and Lele [105], is able to obtain a damped solution at the boundary using a simplification of the conservation equations that are solved in the computational domain. Instead of solving the compressible Navier-Stokes equation over the boundary, the method annihilates the in-going sound wave by exploiting the characteristics of the Euler equations, where viscous and reaction terms are neglected. Research on annihilating methods for spurious sound wave reflections of the computational boundaries is not the original motivation for the present study. More details about the non-reflecting boundary conditions applied to aeroacoustic flow solvers may be found in articles such as [106, 107].

The chapter is organized as follows. We recall the primitive variable formulation of the compressible Navier-Stokes equations in the first part of Section 4.2. Then we describe the transformation of the primitive variables into a relative and a reference part. Later in that section we present the governing equations for the non-reflecting boundary conditions. In Section 4.3, we apply the VMS framework in order to discretize in space; we focus our attention on

the construction of the stabilization operator in order to properly reach the low Mach number limit. The implicit scheme for advancing in time and the weak imposition of the non-reflecting conditions are also included in that section. In Section 4.4, we validate the numerical method for two- and three-dimensional applications by using steady manufactured solutions. We also simulate the differential heated cavity and the flow past a cylinder unsteady problems. Two numerical examples are solved as aeroacoustic applications: the Aeolian tones generated by the flow past a cylinder, and the flow past an open cavity. Finally, we address some conclusions in Section 4.5.

4.2 Governing equations

In this section we present the compressible Navier-Stokes equations written in primitive variables. For doing this we depart from the conservative formulation, and express the system in terms of the primitive vector of unknowns. The Jacobian and diffusive matrices are therefore written, and the specific formulation for the ideal gas law is included. Then we implement a change of variables, which allows us to scale the discrete linear system. In the last part we describe the governing equations for the non-reflecting boundary conditions.

4.2.1 Compressible Navier-Stokes equations written in primitive variables

In order to formulate the compressible Navier-Stokes equations (3.1) - (3.3) using primitive variables, we express the conservative unknowns $\boldsymbol{U}$ as a function of the primitive variables $\boldsymbol{Y} = (p, \boldsymbol{u}, T)^\top$, where p is the pressure, $\boldsymbol{u}$ is the velocity, and T is the temperature of the fluid. Hence, the compressible Navier-Stokes equations written in system form (3.6) can now be written as

$$\partial_t \boldsymbol{U}(\boldsymbol{Y}) + \partial_j \boldsymbol{E}_j(\boldsymbol{U}(\boldsymbol{Y})) + \partial_j \boldsymbol{G}_j(\boldsymbol{U}(\boldsymbol{Y})) = \boldsymbol{F}, \tag{4.1}$$

together with initial and boundary conditions (3.11) - (3.13). In the previous equation, $\boldsymbol{E}_j$ and $\boldsymbol{G}_j$ are the convective and diffusive flux in the j-th-direction, respectively, which are defined in (3.7). The vector of forces $\boldsymbol{F} = (0, \rho\boldsymbol{f}, \rho\boldsymbol{f}\cdot\boldsymbol{u} + \rho r)^\top$, contains the right hand side terms of equations (3.1) - (3.3). Here ∂_t and ∂_j are short notations that indicate the Eulerian time derivative and $\partial/\partial x_j$, respectively. The usual summation convention is implied in the equation presented before, with indices running from 1 to d.

If $\boldsymbol{Y}$ is sufficiently smooth, it also satisfies the quasi-linear form

$$\boldsymbol{A}_0(\boldsymbol{Y})\,\partial_t \boldsymbol{Y} + \boldsymbol{A}_j(\boldsymbol{Y})\,\partial_j \boldsymbol{Y} - \partial_k\left(\boldsymbol{K}_{kj}(\boldsymbol{Y})\,\partial_j \boldsymbol{Y}\right) = \boldsymbol{F}. \tag{4.2}$$

Here we have written the divergence of the fluxes in Eq. (4.1) in a more convenient manner by defining the Jacobian matrices $\boldsymbol{A}_0(\boldsymbol{Y}) = \partial\boldsymbol{U}/\partial\boldsymbol{Y}$, $\boldsymbol{A}_j(\boldsymbol{Y}) = \partial\boldsymbol{E}_j(\boldsymbol{U}(\boldsymbol{Y}))/\partial\boldsymbol{Y}$, and the diffusivity matrix $\boldsymbol{K}(\boldsymbol{Y}) = [\boldsymbol{K}_{kj}(\boldsymbol{Y})]$, such that $\partial_j \boldsymbol{G}_j(\boldsymbol{U}(\boldsymbol{Y})) = -\partial_k\left(\boldsymbol{K}_{kj}(\boldsymbol{Y})\,\partial_j \boldsymbol{Y}\right)$.

In order to express all the previous relations in terms of the primitive variables alone, we make the supposition that the fluid is divariant and account for the ideal gas law. This makes it possible to formulate density as a function of the pressure and the temperature, and to rearrange

the Navier-Stokes equations (see [68, 96]). Hence, the Jacobian matrices $\boldsymbol{A}_j(\boldsymbol{Y}), j = 0, ..., d$, can be formulated as functions of primitive variables and thermodynamic coefficients:

$$\boldsymbol{A}_0(\boldsymbol{Y}) = \begin{bmatrix} \rho\beta_t & \mathbf{0}^\top & -\rho\alpha_p \\ \rho\beta_t \boldsymbol{u} & \rho\boldsymbol{I} & -\rho\alpha_p \boldsymbol{u} \\ \rho\beta_t a_1 - \alpha_p T & \rho\boldsymbol{u}^\top & -\rho\alpha_p a_1 + \rho c_p \end{bmatrix},$$

$$\boldsymbol{A}_j(\boldsymbol{Y}) = \begin{bmatrix} \rho\beta_t u_j & \rho\boldsymbol{e}_j^\top & -\rho\alpha_p u_j \\ \rho\beta_t \boldsymbol{u} u_j + \boldsymbol{e}_j & \rho\boldsymbol{I} u_j + \rho\boldsymbol{u} \otimes \boldsymbol{e}_j & -\rho\alpha_p \boldsymbol{u} u_j \\ (\rho\beta_t a_1 - \alpha_p T + 1)\, u_j & \rho\boldsymbol{u}^\top u_j + \rho a_1 \boldsymbol{e}_j^\top & (-\rho\alpha_p a_1 + \rho c_p)\, u_j \end{bmatrix},$$

for $j = 1, .., d,$. Hereafter $\mathbf{0}$ is the vector of $\mathbb{R}^d$ with zero in all its components. The thermodynamic relation a_1 stands for $a_1 = c_v T + p/\rho + |\boldsymbol{u}|^2/2$, α_p is the volume expansivity, and β_t is the isothermal compressibility, given by

$$\alpha_p = -\frac{1}{\rho}\left(\frac{\partial \rho}{\partial T}\right)_p \quad \text{and} \quad \beta_t = \frac{1}{\rho}\left(\frac{\partial \rho}{\partial p}\right)_T, \tag{4.3}$$

with $(\cdot)_p$ and $(\cdot)_T$ defining a constant pressure and temperature constrain, respectively. Additionally, the diffusive matrix $\boldsymbol{K}_{kj}(\boldsymbol{Y})$ is constructed as

$$\boldsymbol{K}_{jj}(\boldsymbol{Y}) = \begin{bmatrix} 0 & \mathbf{0}^\top & 0 \\ \mathbf{0} & \mu\boldsymbol{I} + \frac{1}{3}\mu \boldsymbol{e}_j \otimes \boldsymbol{e}_j & \mathbf{0} \\ 0 & \mu\boldsymbol{u}^\top + \frac{1}{3}\mu u_j \boldsymbol{e}_j^\top & \lambda \end{bmatrix} \quad \text{for } j = 1, .., d, \text{ and}$$

$$\boldsymbol{K}_{kj}(\boldsymbol{Y}) = \begin{bmatrix} 0 & \mathbf{0}^\top & 0 \\ \mathbf{0} & \mu\boldsymbol{e}_j \otimes \boldsymbol{e}_k - \frac{2}{3}\mu \boldsymbol{e}_k \otimes \boldsymbol{e}_j & \mathbf{0} \\ 0 & \mu u_j \boldsymbol{e}_k^\top - \frac{2}{3}\mu u_k \boldsymbol{e}_j^\top & 0 \end{bmatrix} \quad \text{for } k, j = 1, .., d, \text{ with } k \neq j.$$

In all these expressions, we have denote by $\boldsymbol{e}_i$ the unit vector in the i-th direction.

In this work the ideal law for gases is used, so that expressions (4.3) result in $\alpha_p = 1/T$, and $\beta_t = 1/p$. For ideal gases at the low Mach number limit, the fluid exhibits very large values of pressure and temperature. As a consequence, the volume expansivity and the isothermal compressibility tend to zero, and the Jacobian matrices recover the usual incompressible form. The divergence free terms present in both the convective and diffusive matrices constrain the incompressibility of the fluid in this regime. All the matrices stay bounded when the Mach number tends to zero because the density remains almost constant.

4.2.1.1 Relative variable formulation

As stated before, using the primitive formulation for ideal gases in the low Mach number limit results in having to operate with very large quantities in the continuity and energy equations (e.g. if the international system of units is used for typical gases). When an implicit numerical scheme is used, the linear system that needs to be solved at each time step contains very large values for these two equations, making it very inefficient to solve the discrete linear system by using iterative methods. In order to overcome this difficulty, we perform a decomposition of the primitive (absolute) variables $\boldsymbol{Y}$, into a relative (or gauge) part $\boldsymbol{Y}^*$, and a reference (or

atmospheric) part $\boldsymbol{Y}_{\text{atm}}$:

$$\underbrace{\begin{bmatrix} p \\ \boldsymbol{u} \\ T \end{bmatrix}}_{\boldsymbol{Y}} = \underbrace{\begin{bmatrix} p^* \\ \boldsymbol{u}^* \\ T^* \end{bmatrix}}_{\boldsymbol{Y}^*} + \underbrace{\begin{bmatrix} p_{\text{atm}} \\ \boldsymbol{0} \\ T_{\text{atm}} \end{bmatrix}}_{\boldsymbol{Y}_{\text{atm}}},$$

supposing that the reference part is constant. Then, we write Eq. (4.2) by using the relative variables as the unknowns, but caring to take into account the complete contribution for calculating non-linear terms; we have

$$\boldsymbol{A}_0\left(\boldsymbol{Y}\right)\partial_t \boldsymbol{Y}^* + \boldsymbol{A}_j\left(\boldsymbol{Y}\right)\partial_j \boldsymbol{Y}^* - \partial_k\left(\boldsymbol{K}_{kj}\left(\boldsymbol{Y}\right)\partial_j \boldsymbol{Y}^*\right) = \boldsymbol{F}, \tag{4.4}$$

supposing that the reference part related to $\boldsymbol{F}$ is negligible. Moreover, the previous equation can be written for convenience as

$$\boldsymbol{A}_0\left(\boldsymbol{Y}\right)\partial_t \boldsymbol{Y}^* + \boldsymbol{\mathcal{L}}\left(\boldsymbol{Y};\boldsymbol{Y}^*\right) = \boldsymbol{F}, \tag{4.5}$$

by introducing the nonlinear operator

$$\boldsymbol{\mathcal{L}}\left(\boldsymbol{Y};\boldsymbol{Y}^*\right) = \boldsymbol{A}_j\left(\boldsymbol{Y}\right)\partial_j \boldsymbol{Y}^* - \partial_k\left(\boldsymbol{K}_{kj}\left(\boldsymbol{Y}\right)\partial_j \boldsymbol{Y}^*\right), \tag{4.6}$$

which includes the definitions for the convective and diffusive terms. Initial and boundary conditions must be set for the primitive problem written using relative unknowns. We explicitly indicate in our examples how these are prescribed.

4.2.2 Weak form of the problem

Let $\boldsymbol{\mathcal{W}}$ be the space of functions where, for each $t \in (0, t_f)$, the unknowns are well defined, with appropriate regularity that we will not analyze here. Let us also denote by $(\cdot,\cdot)_\omega$ the integral of the product of two functions (scalar or vector valued) in a domain ω, ommiting the subscript when $\omega = \Omega$. Introducing the form

$$A(\boldsymbol{U};\boldsymbol{V},\boldsymbol{W}) := (\boldsymbol{V}, \boldsymbol{A}_j\left(\boldsymbol{U}\right)\partial_j \boldsymbol{W}) + (\partial_k \boldsymbol{V}, \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \boldsymbol{W}), \tag{4.7}$$

the variational form of the problem can be written as: find $\boldsymbol{Y}^* : (0, t_f) \to \boldsymbol{\mathcal{W}}$ such that

$$(\boldsymbol{V}, \boldsymbol{A}_0\left(\boldsymbol{Y}\right)\partial_t \boldsymbol{Y}^*) + A(\boldsymbol{Y};\boldsymbol{V},\boldsymbol{Y}^*) = (\boldsymbol{V},\boldsymbol{F}) + (\boldsymbol{V},\boldsymbol{H})_{\Gamma_N}, \quad t \in (0, t_f), \tag{4.8}$$

$$(\boldsymbol{V},\boldsymbol{Y}) = \left(\boldsymbol{V},\boldsymbol{Y}^0\right), \qquad t = 0, \tag{4.9}$$

for all $\boldsymbol{V}$ in the adequate test functions space. In the case of the compressible Navier-Stokes equations, the Neumann boundary operator is given by the diffusive fluxes

$$\boldsymbol{B}\left(\boldsymbol{Y}\right) = -n_j \boldsymbol{K}_{kj}\left(\boldsymbol{Y}\right)\partial_j \boldsymbol{Y}^*,$$

although part of the convective term could also be integrated by parts and contribute to the Neumann boundary conditions, in particular, the pressure term.

4.2.3 Non-reflecting boundary conditions

Another particular aspect of the nearly incompressible limit is that the computational boundaries, essentially the subsonic outlets, may cause artificial wave reflections related to the ingoing part of the propagated sound wave. In order to overcome this difficulty we incorporate non-reflecting boundary conditions to the compressible formulation.

4.2.3.1 Local one-dimensional characteristic wave equation

Specifically, the characteristic boundary conditions for the Navier–Stokes equations by Poinsot and Lele [105] are implemented to counteract spurious reflections. The characteristic boundary conditions aim to compute the flow at the boundary using an approximation of the governing equations that are solved inside the computational domain. This method employs the hypothesis that, as waves are associated with the hyperbolic part of the Navier-Stokes equations, not with the diffusive part, the flow at the boundary can be approximated as inviscid and one-dimensional. Hence, the Euler equations can be represented as a local one-dimensional characteristic wave equation.

The characteristics analysis is related to the flow description at the boundary in local coordinates instead of using the interior domain coordinates. Considering $d = 3$, a transformation from the computational domain with Cartesian coordinates (x_1, x_2, x_3), to a local reference frame with coordinates (ξ, η, ζ), is performed to determine the physical waves and their orientation at the boundary. In the new local reference system of coordinates, ξ defines the normal direction to the boundary Γ, and η and ζ define tangential directions.

According to the characteristic analysis performed by Thompson [104] over the Euler equations, these can be decomposed using the primitive variables (including the definition of the u_i flow velocity in the local reference frame $\boldsymbol{u} = (u_\xi, u_\eta, u_\zeta)^\top$), with the axial derivatives grouped into a constant source term, so that, the convective term can be diagonalized using a similarity transformation. The resulting nonlinear convective problem is then written as follows:

$$\frac{\partial p^*}{\partial t} + \frac{1}{2}(\chi_5 + \chi_1) = 0, \tag{4.10}$$

$$\frac{\partial u_\xi}{\partial t} + \frac{1}{2\rho c}(\chi_5 - \chi_1) = 0, \tag{4.11}$$

$$\frac{\partial u_\eta}{\partial t} + \chi_3 = 0, \tag{4.12}$$

$$\frac{\partial u_\zeta}{\partial t} + \chi_4 = 0, \tag{4.13}$$

$$\frac{\partial T^*}{\partial t} + \frac{T}{\rho c^2}\left[-\chi_2 + \frac{(\gamma - 1)}{2}(\chi_5 + \chi_1)\right] = 0. \tag{4.14}$$

Here χ_i, stands for the amplitude variations of the characteristic i-th wave, $i = 1, 2, 3, 4, 5$.

Those variations are calculated in terms of normal derivatives to the boundary as follows:

$$\boldsymbol{\chi} = \begin{pmatrix} \lambda_1 \left(\frac{\partial p^*}{\partial \xi} - \rho c \frac{\partial u_\xi}{\partial \xi} \right) \\ \lambda_2 \left(c^2 \frac{\partial \rho}{\partial \xi} - \frac{\partial p^*}{\partial \xi} \right) \\ \lambda_3 \frac{\partial u_\eta}{\partial \xi} \\ \lambda_4 \frac{\partial u_\zeta}{\partial \xi} \\ \lambda_5 \left(\frac{\partial p^*}{\partial \xi} + \rho c \frac{\partial u_\xi}{\partial \xi} \right) \end{pmatrix}, \tag{4.15}$$

where λ_i are the characteristic velocities:

$$\lambda_1 = u_\xi - c, \tag{4.16}$$
$$\lambda_2 = \lambda_3 = \lambda_4 = u_\xi, \tag{4.17}$$
$$\lambda_5 = u_\xi + c. \tag{4.18}$$

Therefore, the characteristics determine at each boundary which waves associated to those values are incoming or outgoing. The validity of this method, which has been pointed out in [108], is subject to the consideration that the characteristic waves represent a one-dimensional propagation in the stream-wise direction. As no genuine characteristics exist for multi-dimensional problems, these equations make a raw approximation of the wave propagation phenomena at the boundary. Because obliquely incident waves are neglected, the accuracy of the method must be kept with the appropriate design of the boundary (as done in the numerical examples to be presented next in the chapter), so that perpendicular waves impinging the non-reflecting boundaries are privileged.

Again, we propose to use a vector of primitive variables at the boundary $\boldsymbol{J} = (p, u_\xi, u_\eta, u_\zeta, T)^\top$, which is written conveniently by using the local reference system, but also including the decomposition of the unknowns into a relative and an absolute part. For convenience we introduce a general form, in which equations (4.10) - (4.14) can be written, that contains additional terms as explained next. We write the boundary problem in vector form as follows:

$$\partial_t \boldsymbol{J}^* + \boldsymbol{A}_B(\boldsymbol{J}) \frac{\partial \boldsymbol{J}^*}{\partial \xi} + \boldsymbol{S}_B(\boldsymbol{J})(\boldsymbol{J}^* - \widehat{\boldsymbol{J}}^*) = \boldsymbol{F}_B \qquad \text{in} \quad \Gamma \subset \mathbb{R}^d, \quad t \in (0, t_f)\,. \tag{4.19}$$

This arrangement makes it possible to deal with the non-linearities of the problem, and to separate the equation to be solved into a transient, a convective-like, a reactive, and a forcing term. The convective term is written as a non-linear $(d+2)\times(d+2)$ convective matrix $\boldsymbol{A}_B(\boldsymbol{J})$ multiplied by the normal derivative of the unknowns at the boundary. The introduction of the reactive and forcing terms allows to deal with the non-reflecting outflow condition (that is explained in detail in the next part). In this sense, we construct the reactive term as a non-linear reactive matrix $\boldsymbol{S}_B(\boldsymbol{J})$ that multiplies the perturbed solution at the boundary $(\boldsymbol{J}^* - \widehat{\boldsymbol{J}}^*)$, being $\widehat{\boldsymbol{J}}^* = (p^*_\infty, u_{\xi\infty}, u_{\eta\infty}, u_{\zeta\infty}, T^*_\infty)^\top$ the unperturbed solution at the far-field. Moreover, the convective, and reactive terms are also grouped into the non-linear boundary operator

$$\boldsymbol{\mathcal{L}}_B\left(\boldsymbol{J}; \boldsymbol{J}^*\right) = \boldsymbol{A}_B(\boldsymbol{J}) \frac{\partial \boldsymbol{J}^*}{\partial \xi} + \boldsymbol{S}_B(\boldsymbol{J}) \boldsymbol{J}^*. \tag{4.20}$$

The boundary problem can be solved without difficulty by transforming back (4.19) into a problem which posseses $\boldsymbol{Y}$ as unknowns. This is achieved using the transformation matrix $\boldsymbol{T}$ written in system form between the original reference system and the local boundary system:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & T_{\xi x_1} & T_{\xi x_2} & T_{\xi x_3} & 0 \\ 0 & T_{\eta x_1} & T_{\eta x_2} & T_{\eta x_3} & 0 \\ 0 & T_{\zeta x_1} & T_{\zeta x_2} & T_{\zeta x_3} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

At this point, let us note that, if $\boldsymbol{Y}^*$ is an element in the domain of $\boldsymbol{\mathcal{L}}(\boldsymbol{Y};\boldsymbol{Y}^*)$ and $\boldsymbol{H}$ is an element in its range, and, if $\boldsymbol{J}^*$ is an element in the domain of $\boldsymbol{\mathcal{L}}_B(\boldsymbol{J},\boldsymbol{J}^*)$ and $\boldsymbol{F}_B$ an element in its range, the unknowns and force vectors can be transformed respectively as

$$\boldsymbol{Y}^* = \boldsymbol{T}^{-1}\boldsymbol{J}^* \quad \text{and} \quad \boldsymbol{H} = \boldsymbol{T}^{-1}\boldsymbol{F}_B.$$

Here we employ the fact that the derivatives of $\boldsymbol{T}$ and $\boldsymbol{T}^{-1}$ are null, and that the solution is sufficiently smooth. Consequently, the local boundary problem can be transformed onto the original reference system. For convenience we multiply both sides of the transformed boundary problem with the transient Jacobian matrix $\boldsymbol{A}_0(\boldsymbol{Y})$, and thus the problem is now written as

$$\begin{aligned} \boldsymbol{A}_0(\boldsymbol{Y})\,\partial_t \boldsymbol{Y}^* + \boldsymbol{A}_0(\boldsymbol{Y})\,\boldsymbol{T}^{-1}\boldsymbol{\mathcal{L}}_B(\boldsymbol{J},\boldsymbol{T}\boldsymbol{Y}^*) = \\ \boldsymbol{A}_0(\boldsymbol{Y})\,\boldsymbol{T}^{-1}\boldsymbol{F}_B + \boldsymbol{A}_0(\boldsymbol{Y})\,\boldsymbol{T}^{-1}\boldsymbol{S}_B(\boldsymbol{J})\widehat{\boldsymbol{J}}^* \qquad \text{in} \quad \Gamma \subset \mathbb{R}^d. \end{aligned} \tag{4.21}$$

4.2.3.2 Non-reflecting subsonic outflow

Non-reflecting boundaries are constructed by taking control over the amplitude variations of the characteristic waves. In particular, for subsonic outlets, the velocity of sound propagation waves along the negative and positive stream-wise directions λ_1 and λ_5 are crucial. An appropriate non-reflecting boundary condition for this case must set the value of the wave amplitude variation χ_1 associated to the characteristic velocity $\lambda_1 = u_\xi - c$, which is the ingoing part of the wave. In this work we implement the Rudy and Strikwerda relation [99]:

$$\chi_1 = \sigma c \frac{(1-\mathrm{M}^2)}{l}(p^* - p^*_\infty), \tag{4.22}$$

in order to calculate the incoming wave amplitude variation at the subsonic outflow. In the previous relation σ stands for a pressure relaxation parameter, l is the characteristic length-scale of the propagation domain, and p^*_∞ is the unperturbed value for pressure. At low Mach numbers, the previous relation simplifies to a scaled pressure relaxation parameter multiplying the pressure perturbation.

For the three-dimensional non-reflecting subsonic outflow, matrices $\boldsymbol{A}_B(\boldsymbol{J})$, and $\boldsymbol{S}_B(\boldsymbol{J})$ are defined respectively as

$$\boldsymbol{A}_B(\boldsymbol{J}) = \begin{bmatrix} \frac{1}{2}(u_\xi + c) & \frac{\rho c}{2}(u_\xi + c) & 0 & 0 & 0 \\ \frac{1}{2\rho c}(u_\xi + c) & \frac{1}{2}(u_\xi + c) & 0 & 0 & 0 \\ 0 & 0 & u_\xi & 0 & 0 \\ 0 & 0 & 0 & u_\xi & 0 \\ \frac{T}{\rho c^2}\left[\frac{1}{2}(\gamma-1)(u_\xi + c) + u_\xi\right] - \frac{u_\xi}{\rho R} & \frac{T}{2c}(\gamma-1)(u_\xi + c) & 0 & 0 & u_\xi \end{bmatrix},$$

and

$$\boldsymbol{S}_B(\boldsymbol{J}) = \begin{bmatrix} \sigma c\frac{(1-M^2)}{2l} & 0 & 0 & 0 & 0 \\ -\sigma\frac{(1-M^2)}{2\rho l} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{T\sigma(\gamma-1)(1-M^2)}{2\rho c l} & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The forcing term contains zero in all its $(d+2)$ components.

Note that for this non-reflecting condition, χ_1 is the only amplitude variation that is modeled (using (4.22) instead of calculating it with (4.15)). The remaining components of $\boldsymbol{\chi}$ are calculated from relation (4.15).

4.2.3.3 Non-reflecting subsonic inflow

In the case of a non-reflecting subsonic inflow, four characteristic waves are entering the domain ($\chi_1, \chi_2, \chi_3,$ and χ_4), and only one (χ_5) is leaving. This type of boundary condition relies on the imposition of (the given) Dirichlet conditions for velocity and temperature, but calculates the pressure depending on the outgoing wave crossing the boundary. Since χ_1 and χ_5 are the characteristic waves involved in the pressure equation, the method calculates the outgoing amplitude χ_5 from the interior domain, and controls χ_1 (related to the reflection) by using the following equation obtained from (4.11):

$$\chi_1 = \chi_5 + 2\rho c\frac{\partial u_\xi}{\partial t}. \tag{4.23}$$

Hence, using Eq (4.10) the resulting equation for pressure is such that

$$\frac{\partial p^*}{\partial t} + (u_\xi + c)\left(\frac{\partial p^*}{\partial \xi} + \rho c\frac{\partial u_\xi}{\partial \xi}\right) = -\rho c\frac{\partial u_\xi}{\partial t}. \tag{4.24}$$

In this work we only consider constant and homogeneous inlet velocities. Therefore, the RHS of the previous equation vanishes, and the components of $\boldsymbol{F}_B$ are identically zero. In the case of the $\boldsymbol{A}_B(\boldsymbol{J})$ matrix, it is composed by $(u_\xi + c)$ in the first position of the first row, and by $\rho c(u_\xi + c)$ in the second position of the first row. The rest of terms are zero. Finally, the $\boldsymbol{S}_B(\boldsymbol{J})$ matrix contains zero in all its components.

4.3 Numerical methods

In the previous section, we have developed a compressible Navier-Stokes formulation conceived for ideal gases, which cares for splitting the primitive unknowns into a relative and an absolute part. The resulting model is designed to be suitable at the low Mach number limit. Non-reflecting boundary conditions have been also described for solving aeroacoustic applications. As explained before, we discretize Eqs. (4.5) and (4.21) in space using the finite element method. However, when the Galerkin method is used to approximate the Navier-Stokes equations (which possess non-symmetric operators), an unstable behavior of the solution might

appear when convection is dominant, and due to the incompatibility between the interpolation spaces of the different variables.

In this section, we recall the VMS framework and apply it to the compressible flow equations in order to stabilize the finite element formulation. We then explain the particular features of the VMS method that we include in the numerical formulation. Later, we construct the approximation of the stabilization matrix that is suitable for the low Mach number limit. In the present chapter, the extra diffusion that is given by shock capturing methods (see [102]) is not needed because our interest is restricted to low Mach regimes. At the end of this section, we present the implicit scheme for advancing in time the compressible equations, the approximation for the time tracking of the subscales, and the numerical scheme that we apply to obtain the solution at the non-reflecting boundaries.

4.3.1 The space discrete variational multi-scale stabilized finite element formulation

Let us first consider a finite-element partition $\mathcal{T}_h = \{K\}$ of the domain Ω. The diameter of the element partition is denoted by h. We define the finite element space $\boldsymbol{\mathcal{W}}_h \subset \boldsymbol{\mathcal{W}}$ as made of continuous piecewise polynomial functions in space. The Galerkin approximation to problem (4.8)-(4.9) can be stated as follows: find $\boldsymbol{Y}_h^* : (0, t_f) \to \boldsymbol{\mathcal{W}}_h$ such that

$$\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{Y}_h\right)\partial_t \boldsymbol{Y}_h^*\right) + A(\boldsymbol{Y}_h; \boldsymbol{V}_h, \boldsymbol{Y}_h) = \left(\boldsymbol{V}_h, \boldsymbol{F}\right) + \left(\boldsymbol{V}_h, \boldsymbol{H}\right)_{\Gamma_N}, \quad t \in (0, t_f), \tag{4.25}$$

$$\left(\boldsymbol{V}_h, \boldsymbol{Y}_h\right) = \left(\boldsymbol{V}_h, \boldsymbol{Y}^0\right), \qquad t = 0, \tag{4.26}$$

for all $\boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h^0$, the discrete space of test functions (i.e., with components vanishing where Dirichlet conditions are prescribed on the boundary).

This approximation suffers from instability problems, which vary according to the way to construct $\boldsymbol{\mathcal{W}}_h$ (e.g. in the case of equally interpolating spaces), and to the weight of the non-linear convective term.

We stabilize the compressible Navier-Stokes equations based on the VMS approach introduced in [17]. The basic idea is to approximate the effect of the components of the solution of the continuous problem that cannot be solved by the finite element mesh. The method decomposes the space of the unknowns into a finite-dimensional space $\boldsymbol{\mathcal{W}}_h$, and an infinite-dimensional one, $\widetilde{\boldsymbol{\mathcal{W}}}$, so that $\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{W}}_h \oplus \widetilde{\boldsymbol{\mathcal{W}}}$. The unknown and the test functions are accordingly split as $\boldsymbol{Y}^* = \boldsymbol{Y}_h^* + \widetilde{\boldsymbol{Y}}^*$ and $\boldsymbol{V} = \boldsymbol{V}_h + \widetilde{\boldsymbol{V}}$, respectively. Equation (4.8) can be equivalently written as the system of equations

$$\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{Y}\right)\partial_t \boldsymbol{Y}^*\right) + A(\boldsymbol{Y}; \boldsymbol{V}_h, \boldsymbol{Y}^*) = \left(\boldsymbol{V}_h, \boldsymbol{F}\right) + \left(\boldsymbol{V}_h, \boldsymbol{H}\right)_{\Gamma_N}, \tag{4.27}$$

for all $\boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h^0,\ t \in (0, t_f)$, and

$$\left(\widetilde{\boldsymbol{V}}, \boldsymbol{A}_0\left(\boldsymbol{Y}\right)\partial_t \boldsymbol{Y}^*\right) + A(\boldsymbol{U}; \widetilde{\boldsymbol{V}}, \boldsymbol{U}) = \left(\widetilde{\boldsymbol{V}}, \boldsymbol{F}\right) + \left(\widetilde{\boldsymbol{V}}, \boldsymbol{H}\right)_{\Gamma_N}, \tag{4.28}$$

for all $\widetilde{\boldsymbol{V}} \in \widetilde{\boldsymbol{\mathcal{W}}}^0,\ t \in (0, t_f)$, and likewise for the initial condition, Eq. (4.9). In Eq. (4.28), $\widetilde{\boldsymbol{\mathcal{W}}}^0$ is the space of subscale test functions.

4.3.1.1 Finite element equation

We first analyze the equation for the finite element scale. The second term in the left-hand-side (LHS) of Eq. (4.27) can be split as

$$A(\boldsymbol{Y};\boldsymbol{V}_h,\boldsymbol{Y}^*) = A\left(\boldsymbol{Y};\boldsymbol{V}_h,\boldsymbol{Y}_h^*\right) + A\left(\boldsymbol{Y};\boldsymbol{V}_h,\widetilde{\boldsymbol{Y}}^*\right), \tag{4.29}$$

defining the *Galerkin* terms related to the application of the non-linear operator to the finite element unknown as

$$A\left(\boldsymbol{Y};\boldsymbol{V}_h,\boldsymbol{Y}_h^*\right) = \left(\boldsymbol{V}_h,\boldsymbol{A}_j\left(\boldsymbol{Y}\right)\partial_j\boldsymbol{Y}_h^*\right) + \left(\partial_k\boldsymbol{V}_h,\boldsymbol{K}_{kj}\left(\boldsymbol{Y}\right)\partial_j\boldsymbol{Y}_h^*\right), \tag{4.30}$$

and the terms related to the subscales as

$$A\left(\boldsymbol{Y};\boldsymbol{V}_h,\widetilde{\boldsymbol{Y}}^*\right) = \left(\boldsymbol{Y}_h,\boldsymbol{A}_j\left(\boldsymbol{Y}\right)\partial_j\widetilde{\boldsymbol{Y}}^*\right) + \left(\partial_k\boldsymbol{V}_h,\boldsymbol{K}_{kj}\left(\boldsymbol{Y}\right)\partial_j\widetilde{\boldsymbol{Y}}^*\right). \tag{4.31}$$

The objective is to approximate the subscales in order to end up with a problem for the finite element scale alone. For this, we start by integrating by parts the two terms in the RHS of (4.31), so that we obtain

$$A\left(\boldsymbol{Y};\boldsymbol{V}_h,\widetilde{\boldsymbol{Y}}^*\right) = \sum_K\left(\boldsymbol{\mathcal{L}}^*\left(\boldsymbol{Y};\boldsymbol{V}_h\right),\widetilde{\boldsymbol{Y}}^*\right)_K + \sum_K\left(n_j\boldsymbol{K}_{kj}^\top\left(\boldsymbol{Y}\right)\partial_k\boldsymbol{V}_h,\widetilde{\boldsymbol{Y}}^*\right)_{\partial K}. \tag{4.32}$$

Here we have introduced the formal adjoint $\boldsymbol{\mathcal{L}}^*\left(\boldsymbol{U};\cdot\right)$ of the operator $\boldsymbol{\mathcal{L}}\left(\boldsymbol{U};\cdot\right)$, which is

$$\boldsymbol{\mathcal{L}}^*\left(\boldsymbol{Y};\boldsymbol{V}_h\right) = -\partial_j\left(\boldsymbol{A}_j^\top\left(\boldsymbol{Y}\right)\boldsymbol{V}_h\right) - \partial_k\left(\boldsymbol{K}_{kj}^\top\left(\boldsymbol{Y}\right)\partial_j\boldsymbol{V}_h\right). \tag{4.33}$$

Note that (4.29) involves inter-element jumps. For continuous solution finite element spaces, the convective term jump at the element boundaries is continuous because it is a function of the variables and, therefore, its sum across adjacent element boundaries is zero. Instead, the diffusive term at the element boundaries in (4.32) contains derivatives of the variables and it is discontinuous even for continuous finite element spaces. However, this inter-element terms can be neglected by supposing that the subscales vanish at the element boundaries. A further explanation on inter-element subscales can be found in [109]. In this work, we approximate the derivatives of the first and second terms on the RHS of the previous expression, respectively as

$$\partial_j\left(\boldsymbol{A}_j^\top\left(\boldsymbol{Y}\right)\boldsymbol{V}_h\right) \approx \boldsymbol{A}_j^\top\left(\boldsymbol{Y}\right)\partial_j\boldsymbol{V}_h + \frac{\partial\boldsymbol{A}_j^\top\left(\boldsymbol{Y}\right)}{\partial\boldsymbol{Y}}\partial_j\boldsymbol{Y}_h\boldsymbol{V}_h,$$
$$\partial_j\left(\boldsymbol{K}_{kj}^\top\left(\boldsymbol{Y}\right)\partial_k\boldsymbol{V}_h\right) \approx \boldsymbol{K}_{kj}^\top\left(\boldsymbol{Y}\right)\partial_j\partial_k\boldsymbol{V}_h + \frac{\partial\boldsymbol{K}_{kj}^\top\left(\boldsymbol{Y}\right)}{\partial\boldsymbol{Y}}\partial_j\boldsymbol{Y}_h\partial_k\boldsymbol{V}_h.$$

This approximation considers $\partial_j\boldsymbol{Y}\approx\partial_j\boldsymbol{Y}_h$ for the derivatives of the continuous unknowns.

4.3.1.2 Subscale equation

If $\widetilde{\boldsymbol{P}}$ denotes the L^2 projection onto the space of subscales, it is readily seen that, after integrating by parts the second term of the LHS of (4.28), the equation for the subscales can be written as

$$\widetilde{\boldsymbol{P}}\left[\boldsymbol{A}_0\left(\boldsymbol{Y}\right)\partial_t\widetilde{\boldsymbol{Y}}^* + \boldsymbol{\mathcal{L}}\left(\boldsymbol{Y};\widetilde{\boldsymbol{Y}}^*\right)\right] = \widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{Y};\boldsymbol{Y}_h^*\right)\right], \tag{4.34}$$

where $\boldsymbol{R}(\cdot,\cdot)$ stands for the finite residual, and is formally defined as $\boldsymbol{R}(\boldsymbol{Y};\boldsymbol{Y}^*) = \boldsymbol{F} - \boldsymbol{A}_0(\boldsymbol{Y})\,\partial_t\boldsymbol{Y}^* - \boldsymbol{\mathcal{L}}(\boldsymbol{Y};\boldsymbol{Y}^*)$. Since the subscales cannot be represented by the finite element mesh, the effect of the nonlinear operator applied to the subscales needs to be approximated. For this, we adopt a diagonal matrix of stabilization parameters that depends on the unknowns $\boldsymbol{\tau}(\boldsymbol{Y})$, such that an approximation of the nonlinear operator applied to the subscales in each element of the form $\boldsymbol{\mathcal{L}}\left(\boldsymbol{Y};\widetilde{\boldsymbol{Y}}^*\right) \approx \boldsymbol{\tau}^{-1}(\boldsymbol{Y})\,\widetilde{\boldsymbol{Y}}^*$ is made. Hence, for an adequate definition of the projection onto the subscale space, the subscale equation becomes

$$\widetilde{\boldsymbol{P}}\left[\boldsymbol{A}_0(\boldsymbol{Y})\,\partial_t\widetilde{\boldsymbol{Y}}^* + \boldsymbol{\tau}^{-1}(\boldsymbol{Y})\,\widetilde{\boldsymbol{Y}}^*\right] = \widetilde{\boldsymbol{P}}\left[\boldsymbol{R}(\boldsymbol{Y};\boldsymbol{Y}_h^*)\right]. \tag{4.35}$$

The previous equation is a nonlinear ordinary differential equation, which must be solved at the integration points. Here we use two possibilities to construct the space where the subscales belong. The first and the most common choice is to take it equal to the space of the finite element residuals. That is, to define the projection onto the subscale space as the identity $\widetilde{\boldsymbol{P}} = \boldsymbol{I}$ onto the space of finite element residuals. We call this type of subscales as *algebraic* subscales (ASGS). The second possibility is the so called *orthogonal* subscales method (OSGS), which defines the subscales orthogonal to the finite element space, $\widetilde{\boldsymbol{W}} = \boldsymbol{W}_h^{\perp}$. In this case, the projection is defined to be the orthogonal projection onto the finite element space $\widetilde{\boldsymbol{P}} = \boldsymbol{P}_h^{\perp} = \boldsymbol{I} - \boldsymbol{P}_h$, being $\boldsymbol{P}_h$ the L^2-projection onto the finite element space.

Apart from the construction of the spaces where the subscales belong, we call the subscales *dynamic* if the temporal derivative of the subscales is taken into account. Instead, if this temporal derivative is neglected we call the subscales *quasi-static*. Besides that, another possibility is to neglect the subscale effect in all the non-linear terms, for example, in the Jacobian and diffusivity matrices, whereas, if we take it into account we call the subscales *non-linear*. Further results and discussion about these definitions of the subscales may be found in articles such as [53, 110–112] applied to other flow problems.

4.3.2 The matrix $\boldsymbol{\tau}$ of stabilization parameters

The usual compressible definition for the $\boldsymbol{\tau}$ matrix (presented in Chapter 3), includes the local sound velocity that arises from the linearized characteristic compressible flow problem. At the low Mach number limit the sound speed tends to infinity ($c \to \infty$), and therefore, that stabilization matrix definition is not suitable. Some authors (e. g. [67, 96]) tried to design $\boldsymbol{\tau}$ matrices that were suitable for both the low Mach number limit and high Mach number flows. A necessary requirement for those matrices is the *a priori* knowledge of numerical parameters depending on the compressibility of the flow. In this sense, we propose the following stabilization matrix for multiple dimensions:

$$\boldsymbol{\tau}^{-1}(\boldsymbol{Y}) = \begin{bmatrix} \tau_c^{-1}(\boldsymbol{Y}) & \boldsymbol{0}^\top & 0 \\ \boldsymbol{0} & \tau_m^{-1}(\boldsymbol{Y})\,\boldsymbol{I} & \boldsymbol{0} \\ 0 & \boldsymbol{0}^\top & \tau_e^{-1}(\boldsymbol{Y}) \end{bmatrix}, \tag{4.36}$$

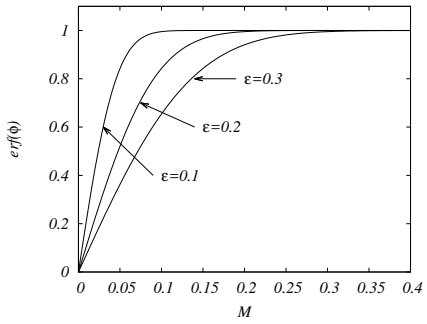


Figure 4.1: Gauss error function against the Mach number. The error function is calculated for different values of parameter ϵ.

where each component is defined as

$$\tau_c^{-1}(\boldsymbol{Y}) = \frac{\rho\tau_m}{h^2}, \tag{4.37}$$

$$\tau_m^{-1}(\boldsymbol{Y}) = \frac{C_1\mu}{h^2} + \frac{C_2\rho u^\star}{h}, \tag{4.38}$$

$$\tau_e^{-1}(\boldsymbol{Y}) = \frac{C_1\lambda}{h^2} + \frac{C_2\rho c_v u^\star}{h}. \tag{4.39}$$

In these expressions C_1 and C_2 are algorithmic parameters that we take as $C_1 = 12p^4$ and $C_2 = 2p$, where p is here the order of the finite element interpolation (not the pressure). In the case of the largest characteristic velocity in the convective contribution $(|\boldsymbol{u}| + c)$, instead of using the common definition, we introduce a modified velocity $u^\star$, that is calculated with the Gauss error function (commonly denoted as erf),

$$u^\star = |\boldsymbol{u}| + \text{erf}(\phi)c, \tag{4.40}$$

where ϕ is defined as $\phi = 2 - 2(\epsilon - \text{M})/\epsilon$. Here ϵ is an algorithmic parameter that determines a certain transition from the compressible to the incompressible regime, which we take as $\epsilon = 0.1$ in the numerical examples presented in this chapter. This definition for the characteristic velocity includes the sound speed, and it converges rapidly to the incompressible definition at the zero Mach limit as shown in Fig. 4.1. The non-linearity of $\boldsymbol{\tau}(\boldsymbol{Y})$, depending on the velocity and the acoustic speed, is also considered for the subscale equation.

4.3.3 Time integration method

At this point, we have described the space discrete stabilized finite element formulation. Let us now comment how we discretize in time.

We partition the time interval $(0, t_f)$ in a sequence of discrete time steps $0 = t^0 < t^1 < ... < t^N = t_f$, with $\delta t > 0$ the time step size, being $t^{n+1} = t^n + \delta t$ for $n = 0, 1, 2, ..., N$.

In this particular application of compressible flows, more precisely, at low Mach number flows, the acoustic speed tends to infinity. This restricts explicit time stepping schemes to very small time step sizes. Therefore, we avoid this restriction by using an implicit monolithic

time integration scheme in order to integrate the time derivatives of Eqs. (4.27) and (4.28). More specifically, we use the Backward Differentiation Formula (BDF) scheme. For the time dependent function $y(t)$, the BDF approximation of order $k = 1, 2, ..,$ is given by $\delta_k y^{n+1}/\delta t$, with

$$\delta_k y^{n+1} = \frac{1}{\gamma_k}\left(y^{n+1} - \sum_{i=0}^{k-1} \phi_k^i y^{n-i}\right),$$

where γ_k and ϕ_k^i are numerical parameters. In particular, for the first and second order BDF schemes, we have

$$\begin{aligned}
\delta_1 y^{n+1} &= y^{n+1} - y^n, \\
\delta_2 y^{n+1} &= \frac{3}{2}\left(y^{n+1} - \frac{4}{3}y^n + \frac{1}{3}y^{n-1}\right).
\end{aligned}$$

As mentioned before, we use the implicit BDF scheme of the first order for discretizing the transient term of the subscales in Eq. (4.35). This is, we obtain a solution of the subscales at the $n+1$ time step after solving

$$\widetilde{\boldsymbol{Y}}^{*n+1} = \boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right)\left(\widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{Y}^{n+1}; \boldsymbol{Y}_h^{*n+1}\right)\right] + \boldsymbol{A}_0\left(\boldsymbol{Y}^{n+1}\right)\frac{\widetilde{\boldsymbol{Y}}^{*n}}{\delta t}\right), \tag{4.41}$$

where the dynamic operator $\boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right)$ is defined as

$$\boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right) = \left(\frac{1}{\delta t}\boldsymbol{A}_0\left(\boldsymbol{Y}^{n+1}\right) + \boldsymbol{\tau}^{-1}\left(\boldsymbol{Y}^{n+1}\right)\right)^{-1}.$$

Because the Jacobian matrix of the transient term is a full matrix, the result of the previous expression would lead to a non-diagonal dynamic operator. Our proposal to avoid off-diagonal terms in the stabilization matrix is to choose a diagonal approximation to $\boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right)$ as follows:

$$\boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right) \approx \begin{bmatrix} (\frac{1}{p^{n+1}}\frac{\rho^{n+1}}{\delta t} + \tau_c^{-1})^{-1} & \mathbf{0}^\top & 0 \\ \mathbf{0} & (\frac{\rho^{n+1}}{\delta t} + \tau_m^{-1})^{-1}\boldsymbol{I} & \mathbf{0} \\ 0 & \mathbf{0}^\top & (\frac{\rho^{n+1}c_p}{\delta t} + \tau_e^{-1})^{-1} \end{bmatrix}. \tag{4.42}$$

From [46], we realize that the previous expression is similar to the dynamic operator used in the VMS formulation of the incompressible Navier-Stokes equations and the energy equation.

In the case of the finite element equation (4.27), we discretize it in time as follows: given the initial conditions $\boldsymbol{Y}^{*0}$, and supposing that the subscales at the initial time step are identically

zero, for $n = 1, 2, ..$, find $\boldsymbol{Y}_h^{*n+1} \in \boldsymbol{\mathcal{W}}_h$, such that:

$$\begin{aligned}
&\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{Y}^{n+1}\right)\frac{\delta_k \boldsymbol{Y}_h^{*n+1}}{\delta t}\right) + \left(\boldsymbol{V}_h, \boldsymbol{A}_j\left(\boldsymbol{Y}^{n+1}\right)\partial_j \boldsymbol{Y}_h^{*n+1}\right) \\
&+ \sum_K \left(\boldsymbol{V}_h, \left(\boldsymbol{I} - \boldsymbol{\tau}^{-1}\left(\boldsymbol{Y}^{n+1}\right)\boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right)\right)\widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{Y}^{n+1}; \boldsymbol{Y}_h^{*n+1}\right)\right]\right)_K \\
&- \sum_K \left(\boldsymbol{V}_h, \boldsymbol{\tau}^{-1}\left(\boldsymbol{Y}^{n+1}\right)\boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right)\boldsymbol{A}_0\left(\boldsymbol{Y}^{n+1}\right)\frac{\widetilde{\boldsymbol{Y}}^{*n}}{\delta t}\right)_K \\
&+ \sum_K \left(\boldsymbol{\mathcal{L}}^*\left(\boldsymbol{Y}^{n+1}; \boldsymbol{V}_h\right), \boldsymbol{\tau}_t\left(\boldsymbol{Y}^{n+1}\right)\left(\widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{Y}^{n+1}; \boldsymbol{Y}_h^{*n+1}\right)\right] + \boldsymbol{A}_0\left(\boldsymbol{Y}^{n+1}\right)\frac{\widetilde{\boldsymbol{Y}}^{*n}}{\delta t}\right)\right)_K \\
&+ \left(\partial_k \boldsymbol{V}_h, \boldsymbol{K}_{kj}\left(\boldsymbol{Y}^{n+1}\right)\partial_j \boldsymbol{Y}_h^{*n+1}\right) = \left(\boldsymbol{V}_h, \boldsymbol{F}\right) \quad \forall \boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h^0.
\end{aligned} \tag{4.43}$$

The previous equation, together with Eq. (4.41), defines the spatial and temporal VMS discretization of the compressible Navier-Stokes equations.

4.3.4 Weak imposition of the non-reflecting boundary conditions

Including the non-reflecting boundary equations (4.21) into the discrete VMS formulation of the compressible problem (4.43) is a challenging topic: an explicit solution of the boundary problem requires to fulfill a time step size restriction for stability that is excessive in the case of low Mach number flows. In contrast, it becomes necessary to write the non-reflecting boundary equations in terms of the compressible Navier-Stokes problem and to incorporate them in the boundary terms of the variational formulation (via the diffusive fluxes across the domain boundaries) if one aims to solve them fully implicitly.

In order to overcome this difficulty, we prescribe an implicit solution of the non-reflecting boundary conditions in a weak sense by means of introducing some penalization terms to (4.43). The implicit solution of the non-reflecting boundary conditions can even exploit the time advancing scheme that is already implemented in the stabilized compressible formulation. This is, given the finite element discretization of the boundary problem, we discretize in time the transient term of the characteristic wave equation by using the implicit BDF method described before in this section for the compressible flow problem. In this sense, the same accuracy of the interior domain equations is used to annihilate the propagation signal.

For some positive parameter $\eta > 0$, the penalty term containing the non-reflecting equations over the boundary that we add to the LHS of the stabilized implicit formulation is

$$\eta\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{Y}^{n+1}\right)\left(\frac{\delta_k}{\delta t}\boldsymbol{Y}_h^{*,n+1} + \boldsymbol{T}^{-1}\boldsymbol{\mathcal{L}}_B\left(\boldsymbol{T}\boldsymbol{Y}_h^{n+1}; \boldsymbol{T}\boldsymbol{Y}_h^{*,n+1}\right)\right)\right)_\Gamma. \tag{4.44}$$

In the same way, the RHS of the stabilized formulation must include the penalty term

$$\eta\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{Y}^{n+1}\right)\left(\boldsymbol{T}^{-1}\boldsymbol{F}_B + \boldsymbol{T}^{-1}\boldsymbol{S}_B(\boldsymbol{T}\boldsymbol{Y}_h^{n+1})\widehat{\boldsymbol{J}}^*\right)\right)_\Gamma. \tag{4.45}$$

The parameter η is defined numerically as $\eta = h\eta_0$; in this way, the boundary integral is dimensionally consistent with the variational problem (4.43), and we have observed from numerical experimentation that η needs to scale as the element size h. The value of η_0 is specifically defined for each numerical example.

4.3.5 Linearization strategy

The implicit scheme brings the difficulty of solving the non-linearities of the discrete spatial problem. To treat this issue we implement a linearization strategy that is based on *Picard's* method. At each step $n+1$, we introduce as a superscript an iteration counter i, and given $\boldsymbol{Y}_h^{n+1,i}$ and $\widetilde{\boldsymbol{Y}}^{n+1,i}$, we compute the finite element unknowns by considering either $\boldsymbol{Y}^{n+1,i} \approx \boldsymbol{Y}_h^{n+1,i}$ for the linear subscales, or $\boldsymbol{Y}^{n+1,i} \approx \boldsymbol{Y}_h^{n+1,i} + \widetilde{\boldsymbol{Y}}^{n+1,i}$ for the non-linear tracking of the subscales. We use this value in all non-linear terms of Eq. (4.43). The loop is iterated until the L^2 norm of the difference between consecutive finite element solutions is below a given convergence criteria, $|\phi_h^{n+1,i+1} - \phi_h^{n+1,i}| < \varepsilon |\phi_h^{n+1,i+1}|$, where ϕ_h stands for any finite element unknown and ε is the tolerance.

Introducing a separated iterative nested loop of index j for the non-linear equation of the subscales at each iteration $i+1$ of the finite element loop is another possibility. In this case, the subscale unknowns $\widetilde{\boldsymbol{Y}}^{n+1,j+1}$ in Eq. (4.41) are solved using the finite element unknowns $\boldsymbol{Y}_h^{*n+1,i+1}$, and the subscales resulting from the previous iteration $\widetilde{\boldsymbol{Y}}^{n+1,j}$. In some previous articles (e.g. in [112]) this possibility has improved the convergence of the linearization scheme. Nevertheless, we have evaluated it in the problem that we consider now, yielding no substantial difference in the convergence rate for the finite element solution; therefore, it has been discarded in the numerical examples to be presented later in the chapter.

When orthogonal subscales are accounted for, the orthogonal projection of the residual is approximated as

$$\boldsymbol{P}^{\perp}\left[\boldsymbol{R}\left(\boldsymbol{Y}^{n+1,i};\boldsymbol{Y}_h^{*n+1,i+1}\right)\right] \approx \boldsymbol{R}\left(\boldsymbol{Y}^{n+1,i};\boldsymbol{Y}_h^{*n+1,i+1}\right) - \boldsymbol{P}_h\left[\boldsymbol{R}\left(\boldsymbol{Y}^{n+1,i};\boldsymbol{Y}_h^{*n+1,i}\right)\right]$$

that is to say, the L^2-projection onto the finite element space $\boldsymbol{P}_h$ is evaluated with the unknowns at the end of the previous iteration. Recall that for algebraic subscales $\widetilde{\boldsymbol{P}} = \boldsymbol{I}$, and projection $\boldsymbol{P}_h$ is not required. In our implementation, the finite element projection is computed using a lumped mass matrix.

4.4 Numerical examples

In the first part of this section, we evaluate the spatial order of accuracy of the numerical formulation in several low Mach number flow cases at steady-state. The method of manufactured solutions is used for two- and three-dimensional domains. Both linear and quadratic approximations are evaluated for the two-dimensional domain. The differential heated cavity and the flow past a cylinder problems are solved next. In both transient problems, we illustrate the performance of the proposed VMS method applied to an unsteady low Mach flow. More precisely, we investigate the temporal behavior of the subscales and address the mesh convergence results for the error against a reference solution. The flow around a cylinder is also used to compare the compressible solution at several low Mach numbers with the established incompressible one, and to simulate the acoustic propagation of waves, in what is commonly referred as the Aeolian tones aeroacoustic problem. Special emphasis is given to the non-reflecting conditions ability to minimize the wave reflection from the computational boundaries. In the last part of this section, the sound generated by the flow past an open cavity is used to test non-linear interactions between the main structures of the flow and the acoustic part of it. The cavity

problem at subsonic condition is an aeroacoustic problem that includes very nonlinear flow characteristics and singular pressure points near the corners of the walls.

In all numerical examples the flow is considered as an ideal gas, with ratio of specific heats $\gamma = 1.4$ and physical properties $c_p = 1.010$ kJ/(kg K) and $c_v = 0.718$ kJ/(kg K). At each time step we solve the non-linearities of Eqs. (4.41) and (4.43) by using Picard's scheme. This leads to a monotonically decreasing relative error between consecutive iterations, with the subsequent convergence of the numerical method. At most fifteen iterations are performed, fulfilling the maximum relative numerical tolerance for the L^2 norm iteration residual of 10^{-10}. As discussed before, iterative solvers can be used for the solution of the linear system of equations as a result of the change of variables in the formulation. Otherwise, the only plausible way to solve transient problems at low Mach numbers is to implement costly direct solvers. In this chapter, we use the enhanced BiCGstab algorithm [113], which is already implemented in the PETSc parallel solver library [114]. We find that using this method, together with an additive Schwarz method and a block ILU preconditioning [115], greatly improves the convergence and the numerical accuracy of the linear solver.

4.4.1 Manufactured solutions

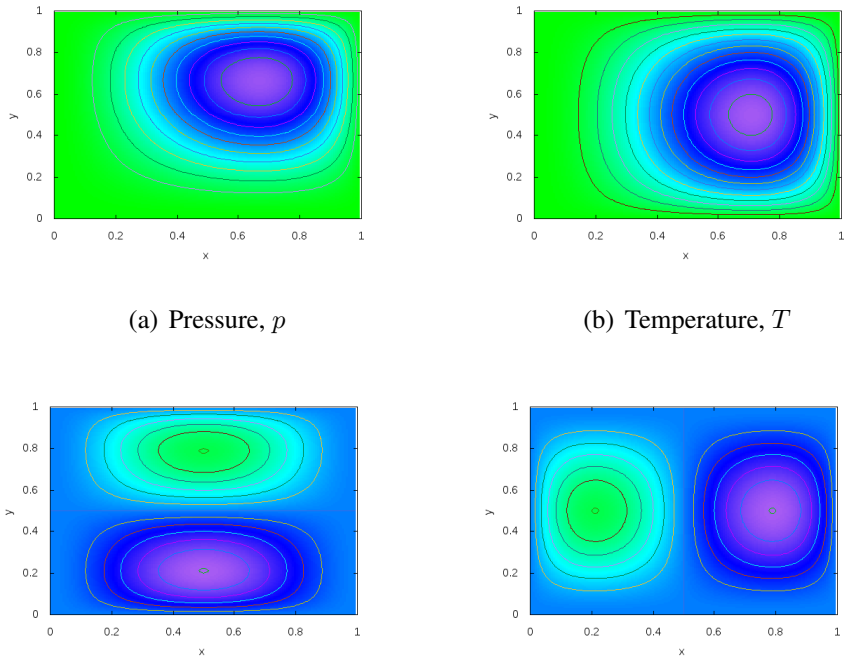
The first numerical example involves steady state compressible flows at low Mach numbers, which are used to quantify the accuracy of the numerical schemes. The method of manufactured solutions has been traditionally used to quantify the numerical error of partial differential equations solvers. The idea is to generate an exact analytical solution *a priori*, that is substituted into the continuum equations to obtain a forcing term. This forcing term, which satisfies the compressible Navier-Stokes equations exactly, is applied to the discrete solver. In this example, an exact solution of pressure, velocity, and temperature is specified in the computational domain. The manufactured solutions are composed of smooth polynomial analytic functions, which are defined to have non-trivial derivatives and no physical meaning. Reference values of pressure and temperature are fixed as $p_{\text{atm}} = 10^5$, and $T_{\text{atm}} = 300$. Dirichlet boundary conditions are fixed over the boundaries of the computational domain using the relative part of the manufactured solutions. In the following, we demonstrate the spatial order of accuracy of several two- and three-dimensional elements.

4.4.1.1 Two dimensions

The polynomial functions for the two-dimensional manufactured solutions are given as follows:

$$
\begin{aligned}
p^* &= x_1^2 x_2^2 (x_1 - 1)(x_2 - 1), \\
u_1 &= 2x_1^2 x_2 (x_1 - 1)^2 (x_2 - 1)(2x_2 - 1), \\
u_2 &= -2x_1 x_2^2 (x_1 - 1)(x_2 - 1)^2 (2x_1 - 1), \\
T^* &= 2x_1^2 x_2 (x_1^2 - 1)(x_2 - 1),
\end{aligned}
$$

and the contour plots of these fields are presented in Fig 4.2. In this two-dimensional problem, the viscosity is fixed to $\mu = 10^{-4}$ kg/(m s), and the thermal conductivity to $\lambda = 1.0$ W/(m K). Therefore, the compressibility regime ranges in the M $= (0.0, 3.5 \times 10^{-5})$ interval, and the local Reynolds number is below Re $= 150$, measured with respect to the domain length.



(a) Pressure, p (b) Temperature, T

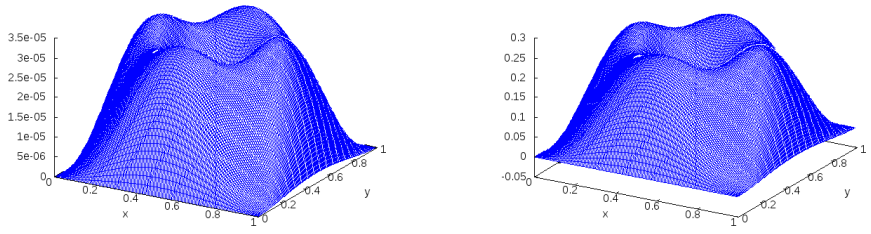
(c) x_1-velocity (d) x_2-velocity

Figure 4.2: Manufactured two-dimensional solutions for the compressible Navier-Stokes equations.

To illustrate one of the key ingredients of the proposed VMS discretization, which is the stabilization matrix design, Fig. 4.3 shows the local compressibility measured with the Mach number and the corresponding modified velocity that is used in the stabilization parameter's calculation. It can be noticed that this velocity tends to zero in the zero Mach limit, but at the same time it converges to the acoustic speed at higher compressibility regions. The Gaussian error function used in the calculation of the modified velocity depends on the normalized compressibility, and allows to include the acoustic speed in the stabilization matrix for the compressible regions of the flow and to equate the compressible and the incompressible stabilization parameters for the momentum and continuity equations at the zero Mach number limit.

To evaluate the accuracy of the numerical method, a series of refined coarse, medium and fine meshes are used in the calculations. The meshes are composed of Q_1 and Q_2 elements distributed in a structured fashion. The characteristic element sizes for the Q_1 meshes are 0.071 m, 0.041 m, and 0.022 m, and for the Q_2 meshes they are 0.1 m, 0.05 m, and 0.025 m. The L^2 norm of the numerical solution error with respect to the exact manufactured solution is used to quantify the accuracy of the computed results.

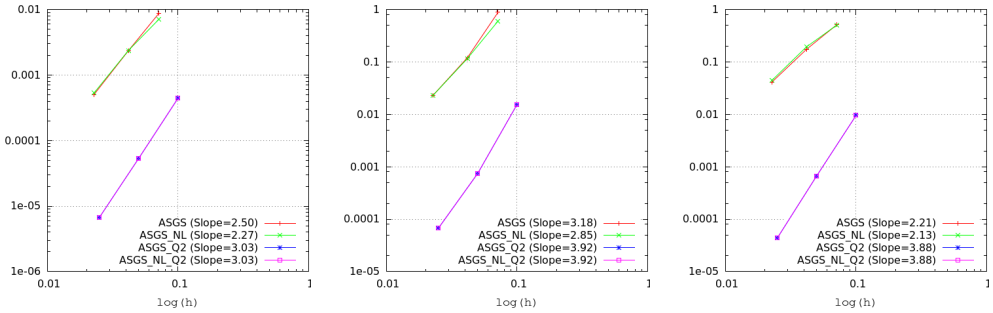
We first study the results obtained by the VMS discretization comparing the algebraic definition of the subscales, in contrast to the orthogonal subscales. We also include the subscales in the non-linear terms of the variational problem (NL). Figures 4.4 and 4.5 depict how the linear and quadratic approximations converge as the mesh is refined. The L^2 norm of the ex-



(a) Local Mach number, M

(b) Local sound speed contribution to $u^\star$, $\text{erf}(\phi)c$

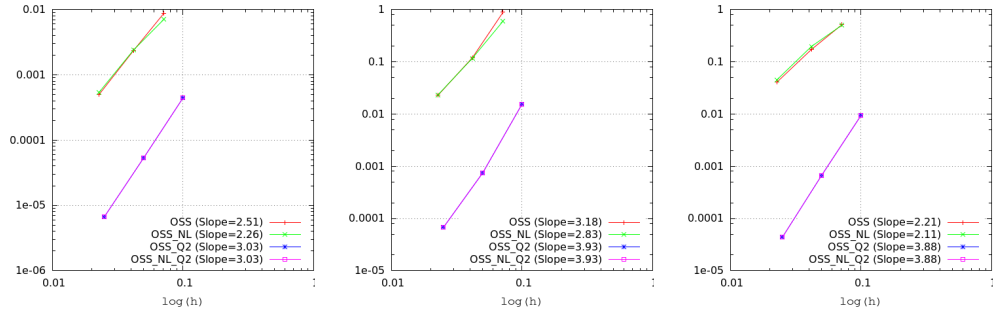
Figure 4.3: Characteristic velocity for the two-dimensional manufactured solutions.



(a) $||p - p_{\text{exact}}||_2$ (b) $||T - T_{\text{exact}}||_2$ (c) $||u - u_{\text{exact}}||_2$

Figure 4.4: Manufactured solutions for the two-dimensional compressible Navier-Stokes equations: L^2 norm of the exact error defining the space where the subscales live as the finite element residual space (ASGS).

act error for pressure, temperature and velocity over the computational domain is presented as a function of the characteristic element sizes. The order of accuracy for the ASGS method, with linear and quadratic approximations, is demonstrated in Fig. 4.4, whereas, for the OSGS method is presented in Fig. 4.5. These figures also show the comparison between including the subscales in the non-linear terms and not including them. Results indicate that the order of accuracy is above two for linear elements and above three for quadratic elements. The order of accuracy for temperature and velocity is better than predicted for linear and quadratic approximations, but as predicted for the quadratic approximation of pressure. For the linear approximation, the accuracy given by the finest mesh is comparable to the coarsest mesh using the quadratic approximation. In this numerical example, it can be observed that the space where the subscales live does not influence the accuracy of the solution. However, including the subscales in the non-linear terms lowers slightly the accuracy and the order of convergence for the Q_1 approximation. On the contrary, for the quadratic approximation, this effect is not representative.



(a) $||p - p_{\text{exact}}||_2$ (b) $||T - T_{\text{exact}}||_2$ (c) $||u - u_{\text{exact}}||_2$

Figure 4.5: Manufactured solutions for the two-dimensional compressible Navier-Stokes equations: L^2 norm of the exact error defining the space where the subscales live as the orthogonal projection to the finite element space (OSGS).

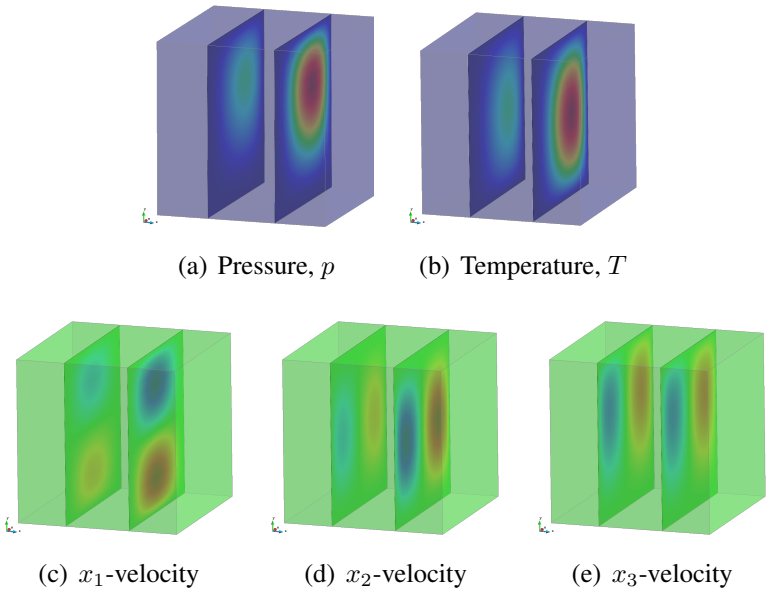
4.4.1.2 Three dimensions

Let us consider now the case of three-dimensional computational domains. Similar functions as those used in the previous two-dimensional problem are used. For the pressure, velocity and temperature fields we use, respectively,

$$
\begin{aligned}
p^* &= x_1^2 x_2^2 x_3^2 (x_1 - 1)(x_2 - 1)(1 - x_3), \\
u_1 &= 2x_1^2 x_2 x_3^2 (x_1 - 1)^2 (x_2 - 1)(2x_2 - 1)(1 - x_3), \\
u_2 &= -2x_1 x_2^2 x_3^2 (x_1 - 1)(x_2 - 1)^2 (2x_1 - 1)(1 - x_3), \\
u_3 &= -2x_1 x_2^2 x_3^2 (x_1 - 1)(2x_1 - 1)(x_2 - 1)(1 - x_3)^2, \\
T^* &= 2x_1^2 x_2 x_3^2 (x_1^2 - 1)(x_2 - 1)(1 - x_3).
\end{aligned}
$$

These three-dimensional fields are presented in Fig. 4.6 as two-dimensional contours at intersecting planes. In order to preserve the low Mach number condition that has been studied in the two-dimensional problem, the physical properties are fixed to $\mu = 10^{-3}$ kg/(m s), and $\lambda = 1.0$ W/(m K). In this case, the order of accuracy of the method is calculated over a sequence of tetrahedral and hexahedral meshes, with 0.052 m, 0.034 m, and 0.025 m of characteristic element sizes. Linear approximations are considered for both types of three-dimensional elements.

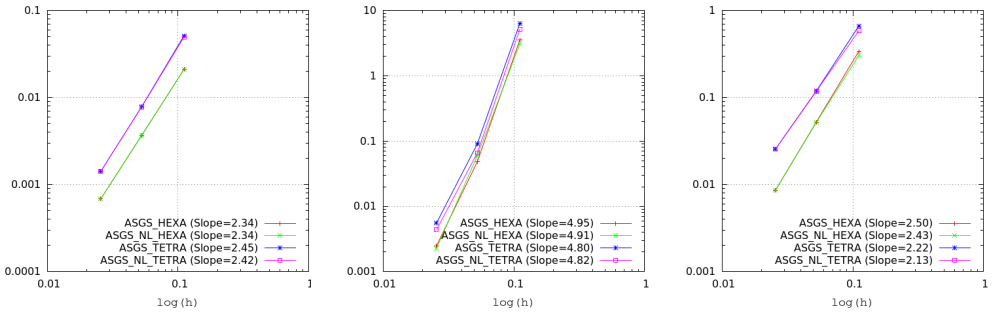
Figures 4.7 and 4.8 depict the L^2 norm of the exact error for pressure, temperature and velocity as a function of the characteristic element sizes. As expected, the observed accuracy is higher for hexahedral than for tetrahedral element meshes. The order of accuracy for both element types is better than predicted for linear approximations. In particular, for the temperature field, the order of accuracy is almost five. As in the results of the two-dimensional problem, neither the space where the subscales live nor the non-linear definition of the subscales affects the accuracy of the method.



(a) Pressure, p (b) Temperature, T

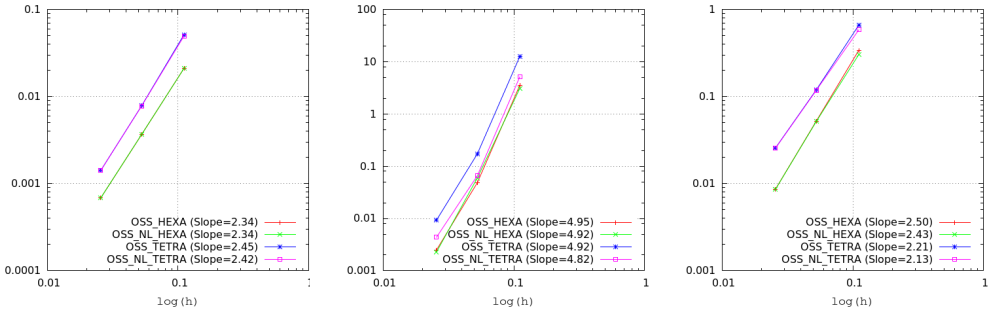
(c) x_1-velocity (d) x_2-velocity (e) x_3-velocity

Figure 4.6: Manufactured three-dimensional solutions for the compressible Navier-Stokes equations.



(a) $||p - p_{\text{exact}}||_2$ (b) $||T - T_{\text{exact}}||_2$ (c) $||u - u_{\text{exact}}||_2$

Figure 4.7: Manufactured solutions for the three-dimensional compressible Navier-Stokes equations: L^2 norm of the exact error defining the space where the subscales live as the finite element residual space.



(a) $||p - p_{\text{exact}}||_2$ (b) $||T - T_{\text{exact}}||_2$ (c) $||u - u_{\text{exact}}||_2$

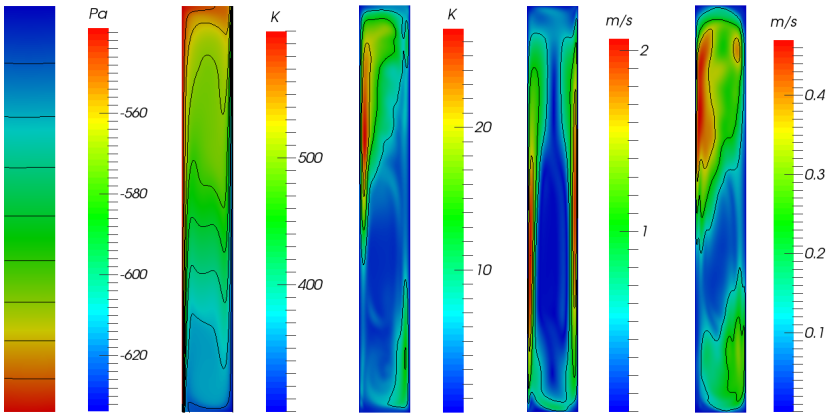
Figure 4.8: Manufactured solutions for the three-dimensional compressible Navier-Stokes equations: L^2 norm of the exact error defining the space where the subscales live as the orthogonal projection to the finite element space.

4.4.2 Differentially heated cavity

The second numerical example is the differentially heated cavity. This is a natural convection flow problem in which the fluid is driven both, by a large temperature gradient between the walls, and by a gravity force. In this numerical example, the flow is considered as a two-dimensional problem with the forcing term of the momentum equation playing a significant role in the development of the buoyancy flow patterns. The computational domain is given by a rectangular cavity $[0, L]\times[0, H]$ of aspect ratio $H/L = 8$, with $L = 1$ m. The temperature at the left (hot) wall is fixed to $T_H = 600$ K, and the temperature at the right (cold) wall to $T_C = 300$ K. No slip and impermeable conditions are set over the cavity walls, together with adiabatic boundary conditions for the upper and lower walls. Gravity is specified to be acting in the negative x_2 direction as $\boldsymbol{g} = (0, -9.8)$ m/s^2. The initial atmospheric pressure $p^0_{\text{atm}} = 152525$ Pa, and the initial temperature of the fluid $T^0 = 450$ K, give an initial uniform density of $\rho^0 = 1.16$ kg/m^3. The viscosity and thermal conductivity are set to $\mu = 2.5 \times 10^{-3}$ kg/(m s), and $\lambda = 3.55$ W/(m K), respectively. The non-dimensional Rayleigh number is defined as the product of the Grashof number and the Prandtl number. In this example, we calculate the Rayleigh number as $\text{Ra} = |\boldsymbol{g}|\theta\rho^2 c_p/(\mu\lambda) = 10^6$, where θ stands for the dimensionless temperature ratio $\theta = 2(T_H - T_C)/(T_H + T_C) = 0.66$.

In order to overcome the mechanical restriction of the pressure imposition for transient and variable flows at closed computational domains, an iterative penalization to the mass conservation equation, of the form $(q_h, \psi(p_h^{*i+1} - p_h^{*i}))$ at iteration $i + 1$, is included in the stabilized formulation. This penalization guarantees that p_h^* is solved correctly, up to a constant, when the relative value of pressure is not set at the computational boundary. The factor ψ is selected numerically as $\psi = 10^{-3}\rho/\mu$, in a way that it does not detriment neither the nonlinear scheme convergence (when ψ is large) nor the algebraic solver convergence (when $\psi \to 0$).

In this numerical example, we use uniform structured meshes composed of Q_2 elements. This type of elements helps to provide sensitivity to high order terms of the discrete equations. Specifically, higher order interpolations make possible to include the second order derivatives present in the residual diffusive term and in the adjoint operator, and therefore, to evaluate all terms of the discrete stabilized formulation. We solve three different grid sizes, corresponding



(a) $\overline{p_h^*}$ (b) $\overline{T_h}$ (c) $T_{h,\text{rms}}$ (d) $|\overline{\boldsymbol{u}_h}|$ (e) $|\boldsymbol{u}_{h,\text{rms}}|$

Figure 4.9: Contour results of the differential heated cavity calculated with an ASGS stabilization method and a structured homogeneous mesh containing Q_2 elements of $h = 0.0125$ m size.

to $h_0 = 0.2$ m, $h_1 = h_0/2$ m, and $h_2 = h_1/2$ m. The size of these meshes is defined in terms of two successive divisions $h_k = h_0/2^k$, where $k = 0, 1, 2$. An additional simulation is performed by solving the problem with the ASGS method and a fine grid of mesh size $h = 0.0125$ m. This additional solution is used as a reference solution, which allows us to test the accuracy of the VMS stabilization method presented in Section 4.3. We choose to use the ASGS method as a reference solution because this method is similar to the SUPG method when linear elements are used, and the later has been widely tested in the literature in several compressible flow problems [79]. In all cases, we use the second order accurate BDF as the time integration scheme, with a constant time step size of $\delta t = 0.01$ s. The simulations are run until the statistically stationary state (measured as the relative error between consecutive transient results of time-averaged variables) is reached.

The reference flow is essentially characterized by an unsteady behavior that is related to the buoyancy produced near the lateral hot and cold walls. The buoyancy is formed as a horizontal shear layer of the velocity component in the direction of gravity, which is in the positive x_2 direction for the hot wall, and in the negative x_2 direction for the cold wall. The superposition of the hot and cold wall shear layers generates a broad flow stream cycling around the cavity. The numerical results obtained in the present simulations correspond to the previously referenced solutions in literature, specifically, we obtain similar results to the instant contours of fields presented in [116] for the same cavity ratio and Grashof number. Both the time average and the root mean square values of the reference solution are depicted in Fig. 4.9. In order to quantify the VMS stabilization attributes, we calculate some statistic measures over the discrete time-dependent variables. With regard to the time-dependent discrete solution $\phi_h(\boldsymbol{x}^a, t)$ of a certain variable ϕ at node $\boldsymbol{x}^a$, the time-averaged discrete solution is denoted as $\overline{\phi}_h(\boldsymbol{x}^a)$, and the root mean square as $\phi_{h,\text{rms}}(\boldsymbol{x}^a)$. We include the calculation of the root mean square since the time-dependency of the flow is particularly important in this numerical example, and with this calculation, we retrieve a measure of the

fluctuations strength. In particular, we observe important temperature fluctuations at the top of the hot wall and in the bottom of the cold wall, where the buoyancy reaches its maximum. Root mean square results for temperature also indicate that the hot wall produces most of the fluctuations. The time-averaged temperature results, on the other hand, yield a smooth distribution from the hot to the cold wall. We also observe that temperature fluctuations are closely related to fluctuations in the velocity field (expressed by the root mean square values of the velocity magnitude) and that the time-averaged velocity magnitude field exhibits high-velocity values near the mid-length of the walls, where the action of the buoyancy is important. On the contrary, the velocity field is almost steady at the center of the cavity, expressed by a small value of both the time-averaged and root mean square velocity results. On the other hand, the time-averaged pressure field gives an almost-linear distribution along the x_2 axis, mainly described by the gravity action in that direction. The compressibility range of the simulation is between $[0, 0.005]$, measured in terms of the time-averaged non-dimensional Mach number. This compressibility indicates that at any instant of the simulation the flow is subsonic, and even nearly incompressible for certain local regions of the flow.

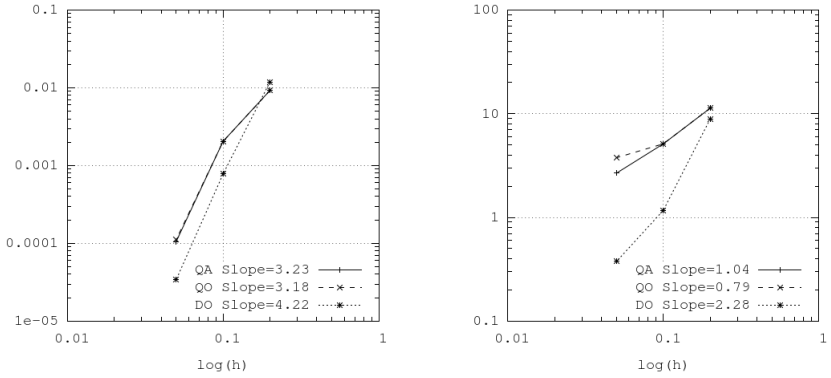
Table 4.1: Differential heated cavity error results.

		ASGS	OSGS	
		quasi-static	quasi-static	Dynamic
Error($\overline{T}$)	$h = 0.2$	9.26×10^{-3}	9.21×10^{-3}	1.19×10^{-2}
	$h = 0.1$	2.05×10^{-3}	2.05×10^{-3}	7.81×10^{-4}
	$h = 0.05$	1.05×10^{-4}	1.11×10^{-4}	3.41×10^{-5}
Error($\overline{\lvert \boldsymbol{u} \rvert}$)	$h = 0.2$	11.2991	11.3208	8.8726
	$h = 0.1$	5.127	5.1267	1.1703
	$h = 0.05$	2.6753	3.7687	0.37605
Error($\lvert \boldsymbol{u}_{\text{rms}} \rvert$)	$h = 0.2$	7.1079	7.237	3.3427
	$h = 0.1$	4.5886	4.5858	0.31857
	$h = 0.05$	0.13295	0.18367	0.084087

In order to perform a quantitative comparison between the VMS stabilization methods, we calculate the discrete L^2 norm of the error by taking

$$\text{Error}(\overline{\phi}) = \frac{\sum_a \left(\overline{\phi}_h(\boldsymbol{x}^a) - \overline{\phi}(\boldsymbol{x}^a)\right)^2}{\sum_a \left(\overline{\phi}(\boldsymbol{x}^a)\right)^2},$$

where $\overline{\phi}(\boldsymbol{x}^a)$ refers to the time average of the reference solution at node $\boldsymbol{x}^a$. As described before, we consider as the reference solution the one obtained using the ASGS solution and the $h = 0.0125$ m mesh. In this numerical example, we aim to compare the quasi-static definition for the subscales in contrast to the dynamic subscales. For simplicity, we consider the space where the subscales live as the orthogonal to the finite element space, and the linear definition of the subscales. We also compare with the ASGS quasi-static solution, that can be considered as the traditional VMS method in the literature. The error between coarse grid results and the reference solution is presented in Table 4.1 for the different mesh sizes. Error calculations are



(a) $\overline{T}$ (b) $|\overline{\boldsymbol{u}}|$

Figure 4.10: Convergence of the differential heated cavity results calculated with a structured mesh containing Q_2 elements of $h = 0.0125$ m size: (a) Time-average of temperature, and (b) Time-averaged velocity magnitude.

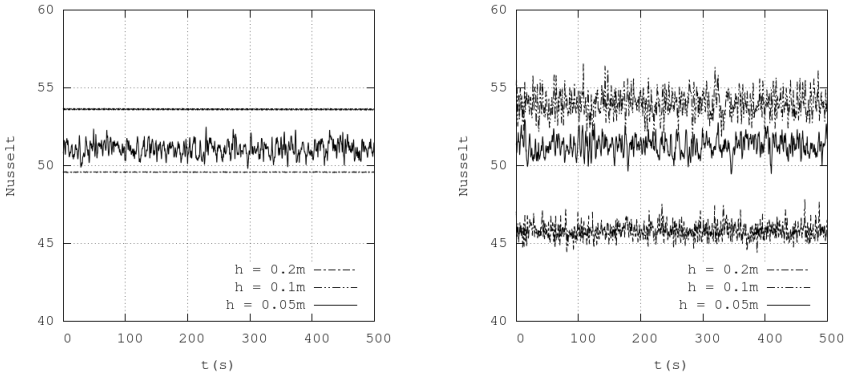
performed for the time-averaged temperature and velocity fields, and for the root mean square of the velocity magnitude. Considering the time-averaged temperature error, this is almost the same for quasi-static subscales in the ASGS and OSGS methods. Instead, the dynamic subscales improve the accuracy for the OSGS method, being considerably more precise by almost an order of magnitude for the smaller mesh sizes. The error of the time-averaged velocity magnitude and the error for the root mean square of the velocity magnitude field are also in line with the error results for temperature.

The mesh convergence results for the error are presented in Fig. 4.10. Convergence plots are displayed for the error of the time-averaged temperature and velocity magnitude fields. Convergence results for the error of the time-averaged temperature field give a slope greater than three. Oppositely, the convergence order is smaller for the error of the velocity field, both for the time-averaged value and for the root mean square value of the magnitude of velocity; possibly, these results are not inside the asymptotic range. Nevertheless, the most accurate method is the OSGS with the inclusion of the dynamic subscales.

Finally, we present the calculations of the non-dimensional *Nusselt* number to investigate the transient behavior of the flow field. The Nusselt number relates the heat transferred from the hot to the cold wall, and it is calculated as

$$\text{Nusselt}(\boldsymbol{x}, t) = \frac{L}{T_H - T_C} n_j \partial_j T(\boldsymbol{x}, t), \qquad \boldsymbol{x} \in \Gamma, t > 0.$$

In particular, the transient behavior of the Nusselt number is evaluated by integrating the previous equation over the hot wall. We observe in the transient plots that are displayed in Fig. 4.11 for the different mesh sizes and stabilizing methods, that the unsteady character of the Nusselt number is subject to the inclusion of the dynamic subscales. This unsteady behavior, which is related to the buoyancy production near the wall, is observed in the reference solution, as well in the $h = 0.05$ m mesh size simulations. If we compare the unsteady behavior for the $h = 0.2$ m and $h = 0.1$ m grids, we observe that the solution given by the dynamic subscales is fluctuating, in contrast to the steady result of the quasi-static subscales. This is, the buoyancy



(a) Quasi-static subscales (b) Dynamic subscales

Figure 4.11: Nusselt results of the differential heated cavity calculated with the orthogonal definition of the subscales for three different mesh sizes.

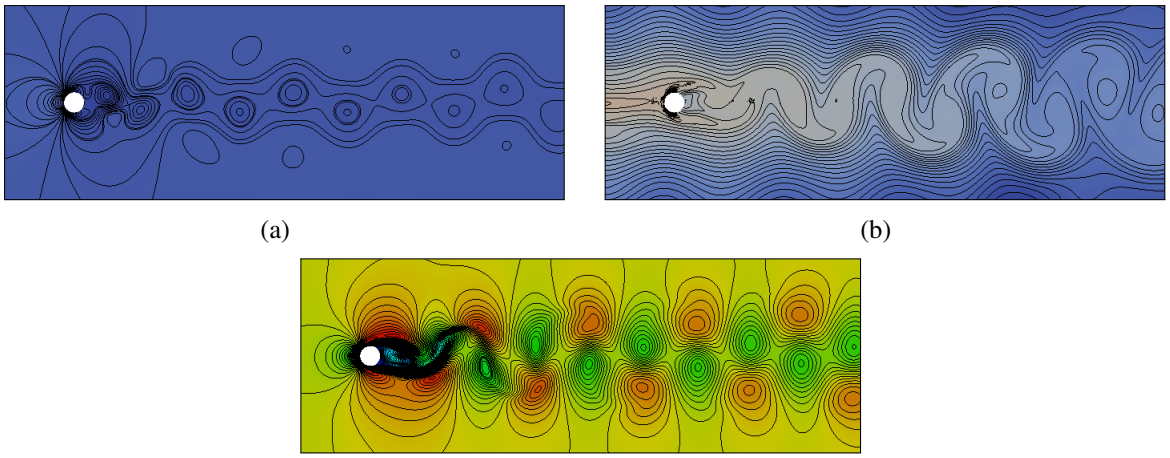
production, and the consequent unsteady behavior of the flow is sensible to the inclusion of the dynamic subscales. This result is related to the lower error given by the dynamic subscales when compared to the reference solution.

4.4.3 Flow past a cylinder

The third numerical example is the laminar viscous flow past a cylinder. The cylinder is defined to be infinitely long in the axial direction and immersed in a compressible viscous flow that impinges it uniformly. For a Re $= 100$ number (based on the cylinder diameter), the flow is unsteady and laminar, developing an oscillating wake behind the cylinder. This example is solved as a two-dimensional problem, and it is a classical unsteady test for viscous incompressible flow solvers which has been used by several authors to quantify the amount of dissipation introduced by a numerical scheme. We use this example to evaluate the performance of the VMS formulation in the low Mach number limit and to investigate its behavior in unsteady flows. The flow around a cylinder is also commonly used to simulate the acoustic propagation of waves. Strong vortices are generated in the wake of the cylinder flow, which are transported downstream and cross the outflow, and cause sound waves due to the perturbation of the pressure field. The waves generated by the vortices behind the cylinder are commonly referred as the Aeolian tones, with a sound wave frequency stable at the fixed value of the wake fluctuation. In the case of the aeroacoustic propagation, special emphasis is given to evaluate the non-reflecting conditions ability to minimize the wave reflection from the computational outlet boundaries.

4.4.3.1 Tracking the dynamic subscales

This first part is intended to further investigate the dynamic subscales behavior in the variational formulation. For doing this, we fix the flow conditions to the free-stream Mach number of M $= 0.001$, the Reynolds number to Re $= 100$, and the Prandtl number to Pr $= 0.71$. Inlet flow conditions are set by fixing the relative pressure and temperature variables, together with



(a) (b)

(c)

Figure 4.12: Instantaneous contour fields of the flow past a cylinder at M $= 0.001$: (a) Pressure, (b) temperature and (c) velocity magnitude calculated with an unstructured mesh containing P_1 elements of $h = 0.001$ m size.

the velocity components. Only the relative pressure needs to be fixed over the outflow boundaries due to the low Mach number condition of the problem. A depiction of the developed flow is presented in Fig. 4.12. This figure shows the instantaneous pressure, temperature and velocity magnitude contours computed using the present VMS formulation over a fine unstructured mesh of $h = 0.001$ m element size and linear triangular elements. The time integration scheme used in the calculation of these results is second order, and the time step size has been kept constant at $\delta t = 0.001$ s. It can be observed in the figure that, in fact, for this flow conditions, an oscillating wake is developed after the cylinder. Because no qualitative differences can be observed among the results given by the different stabilization methods, we compare them quantitatively by calculating some integral values of the flow, more precisely, we calculate time-integrated values of the lift and drag non-dimensional coefficients. To illustrate this, Fig. 4.13 shows the time history of the non-dimensional drag and lift coefficients obtained for the previously described flow simulation. Hence, we calculate the time average of drag $\overline{C_d}$, the time L^∞ norm of the lift coefficient, and the time L^∞ norm of the Strouhal coefficient, over a time-window of 20 s, once the computations have converged to the statistically steady state. For the flow simulation presented in Fig. 4.12, we obtain the following results: $\overline{C_{d\text{Ref}}} = 1.3993$, $|C_{l\text{Ref}}|_\infty = 0.323$, and $|\text{St}_{\text{Ref}}|_\infty = 0.17498$. These results agree with the experimental values from the literature, such as those published in [79] for the same free stream conditions, and serve as a reference solution henceforth.

The order of accuracy and the mesh convergence of the stabilization methods is demonstrated by comparing the obtained numerical results and the reference values described before. The meshes that are used to calculate the order of accuracy are also generated in terms of two successive divisions, so that, computations are performed over three different P_1 unstructured meshes composed of $h_0 = 0.1$ m and 3874 elements, $h_1 = 0.05$ m and 14008 elements, and $h_2 = 0.025$ m and 54884 elements, respectively. The time integration order is $k = 2$ for all cases with a constant time step size of $\delta t = 0.1$ s.

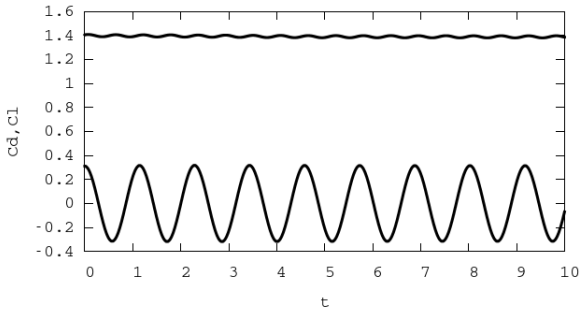


Figure 4.13: Time history of drag and lift coefficients for the flow past a cylinder at M $= 0.001$. Lift and drag results are calculated with an unstructured mesh containing P_1 elements of $h = 0.001$ m size.

The calculated non-dimensional coefficients are presented in Table 4.2 for the different VMS definitions. We compare the results obtained including the dynamic definition of subscales, against the quasi-static subscales, by fixing the space where the subscales live as the orthogonal projection to the finite element space (OSGS). As explained in Section 4.3, the dynamic subscales take into account the temporal derivative of the subscales. It can be observed that the dynamic subscales are the most accurate, which exhibit a higher order of accuracy than the quasi-static subscales. In this sense, it is demonstrated that the approximation of the dynamic stabilization operator defined in Eq. (4.42) is adequate in the low Mach number condition, stabilizing the numerical approximation and improving the accuracy of the variational method. We also test the difference between the accuracy of the method using algebraic (ASGS) and orthogonal subscales, by intentionally defining the subscales as dynamic. In this case, defining the space where the subscales live as the orthogonal to the finite space yields better results for the drag coefficient, but very similar results for the lift and the Strouhal number. No substantial difference in the results has been obtained when the subscales are taken into account in the nonlinear terms of the equations (values not reported).

4.4.3.2 Low Mach number limit

In this second part of the flow past a cylinder example, we compare the results obtained by the compressible stabilized formulation with those obtained by the incompressible one. We address the differences in the solution given by both formulations by decreasing the compressibility of the flow from the subsonic to the nearly incompressible regime. For this, we calculate the difference between the integral non-dimensional coefficients obtained by the compressible flow solver at free-stream Mach numbers of $0.001, 0.01, 0.1, 0.2,$ and 0.5, and those obtained by the incompressible solver (introduced in [46]). The same VMS method is used for both solvers. More specifically, we define the subscales as algebraic, quasi-static and linear, and use the $h = 0.025$ m mesh defined in the previous convergence analysis.

Figure 4.14 shows the difference between the non-dimensional coefficients obtained by

Table 4.2: Flow past a cylinder at M $= 0.001$ results.

		OSGS		ASGS
		Quasi-static	Dynamic	Dynamic
Time average of C_d $\overline{C_{d\text{Ref}}} = 1.3993$	$h = 0.1$	1.2185	1.2401	1.1716
	$h = 0.05$	1.2488	1.2746	1.2393
	$h = 0.025$	1.2933	1.2924	1.2722
Time L^∞ norm of C_l $\lvert C_{l\text{Ref}}\rvert_\infty = 0.323$	$h = 0.1$	0.0006	0.0170	0.0260
	$h = 0.05$	0.1154	0.1258	0.1189
	$h = 0.025$	0.1294	0.1344	0.1346
Time L^∞ norm of St $\lvert \text{St}_{\text{Ref}}\rvert_\infty = 0.17498$	$h = 0.1$	0.1250	0.1250	0.1250
	$h = 0.05$	0.1333	0.1333	0.1333
	$h = 0.025$	0.1538	0.1538	0.1538

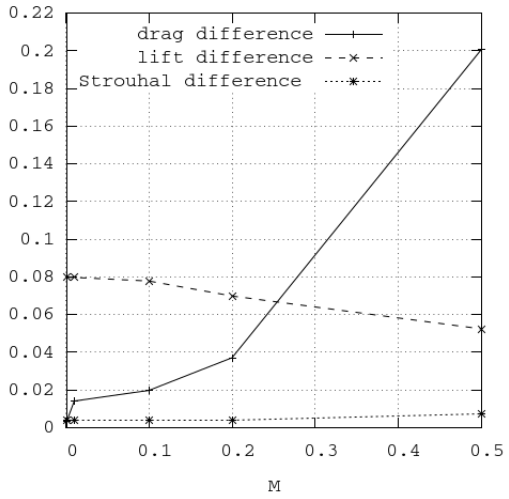
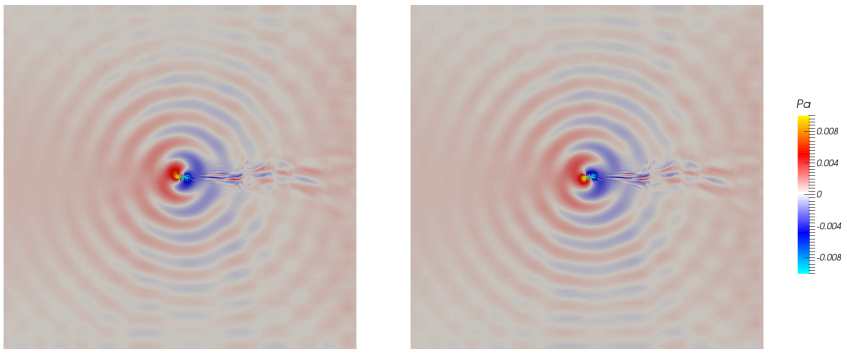


Figure 4.14: Flow past a cylinder results compared to the incompressible solution.

the incompressible solver and the ones obtained by the compressible formulation as a function of the free-stream Mach number. The difference between the compressible and incompressible solutions with respect to the time-averaged drag coefficient scales as the square of the Mach number. It also can be observed that the calculation of the time L^∞ norm of the non-dimensional lift coefficient is sensible to the mesh definition, and the difference with respect to the incompressible solution decreases with the Mach number. The difference in the Strouhal number is maintained constant for the range of the evaluated Mach numbers.

4.4.3.3 Aeolian tones

The last part of this numerical example demonstrates the ability of the compressible solver to deal with aeroacoustic problems at low Mach number conditions. We exploit the fact that the strong vortices at the wake of the cylinder cause a perturbation of the pressure field which



(a) (b)

Figure 4.15: Aeolian tones: instantaneous pressure contour in the far field. Two different, (a) and (b), times of the vortex shedding cycle. Relative values of pressure are depicted within a limited range of values.

is propagated in the form of sound waves to the far field. We adjust the problem conditions to Re $= 1000$ and M $= 0.0583$ in order to be able to benchmark the obtained aeroacoustic solution with referenced acoustic simulations. A big portion of the radiated acoustic field is solved with a computational domain that extends 750 times the diameter of the cylinder in each Cartesian direction. The domain is discretized using an unstructured mesh composed of 9293 Q_2 elements, with an element size of $\sim 0.03D$ at the cylinder surface and $\sim 150D$ at the external boundaries.

The minimization of the incident wave reflection at the boundaries is mandatory to avoid spurious waves that affect the wake structure. In this case, a square computational domain is used with the incident waves impinging the boundaries at different oblique angles, so that the non-reflecting boundary conditions are tested. We set the boundary conditions as follows: we fix velocity and temperature for the inlet left-most boundary, while the non-reflecting subsonic boundary condition is solved for the pressure. The non-reflecting subsonic outlet is set for the rest of the external boundaries. For modeling the non-reflecting subsonic outlet with Eq. (4.22) we fix the pressure relaxation parameter to $\sigma = 20$, the wave characteristic length-scale to $l = 0.3D$, and the unperturbed value for pressure as $p^*_\infty = 0$. The weak condition (4.44) - (4.45) is enforced in every non-reflecting boundary by setting the numerical parameter $\eta_0 = 10^{-3}$, for which we have found good convergence of the numerical method.

The implicit solution of the non-reflecting boundaries allows to set a time step size $\delta t = 0.01$ s, that is only restricted to completely describe the aeroacoustic signal. The simulation is computed using the orthogonal, dynamic and non-linear subscales until the statistical stationary state is reached at about $t = 200$ s. A depiction of the developed flow is presented in Fig. 4.15 for two different times of the vortex shedding cycle. In this sense, we accomplish the direct simulation of the acoustic pressure propagation in the Aeolian tones problem using the stabilized compressible formulation. Typically, for far-field fluctuations, three orders of magnitude smaller than near-field fluctuations, the accurate resolution of acoustic fields requires grids containing up to hundred thousands elements. In this case, we can accurately approximate the scattered pressure wave using only 9293 elements in the mesh. Moreover, in our (direct numerical) solution we can observe that the wake structure of the flow is not

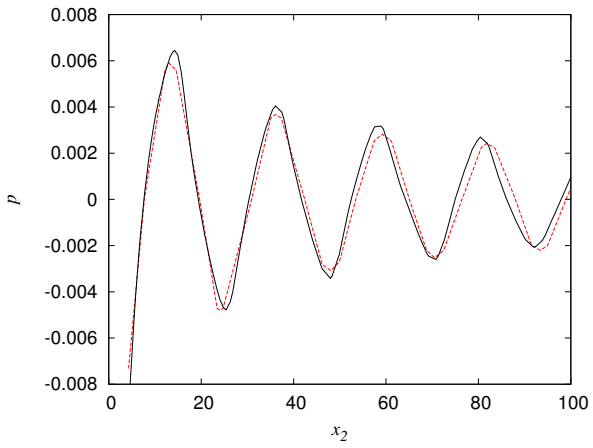


Figure 4.16: Aeolian tones: instantaneous pressure along the positive x_2 direction from the center of the cylinder. Present results are indicated using a solid line. Reference results [89] are denoted with a dashed line.

affected by reflections coming from the artificial computational boundaries, and therefore the non-reflecting boundary conditions exhibit the ability to damp the propagated sound waves and the wake vortices that cross the outflow. In the case of low Mach number flows the characteristic wave approach leads to indistinguishable pressure reflections and distortions at the computational boundaries. At the domain corners, where the waves impinge transversely, the wave is annihilated and no spurious instabilities occur. Although the inlet boundary does not completely damp the pressure wave, and some pile-up is present near the boundary, we have observed that this effect is negligible and does not affect the upstream propagation of sound waves. The inclusion of non-reflecting boundary conditions is crucial in this problem, in which reflections at the boundaries develop oscillations and instabilities at the computational boundaries that end affecting the simulation.

The plot of the pressure wave along the positive x_2 direction is depicted in Fig. 4.16 for the same instant of the vortex shedding cycle. A reference solution that was reported in [89] is also included in the plot. In order to test the method, we use the same type of elements and element size, ranging from $\sim 3\times10^{-3}D$ near the cylinder surface to $\sim 30D$ at the far field, that was reported in the reference solution. Even though the reference sound wave was obtained in that previous work by applying Lighthill's acoustic analogy over the incompressible flow solution, we observe that the acoustic wave propagation corresponds well in both simulations: the frequency and the amplitude of the radiated sound match the vortex shedding, and the dissipation of the wave is in agreement with the reference solution.

4.4.4 Flow past an open cavity

As a final numerical example, the flow past an open cavity problem at high Reynolds number is simulated. This is a challenging aeroacoustic problem that has been solved in [117–119]. The problem definition is an infinitely long rectangular cavity, of aspect ratio two, with length $L = 0.0518$ m and depth $D = 0.0254$ m, that is commonly placed in a Cartesian domain that

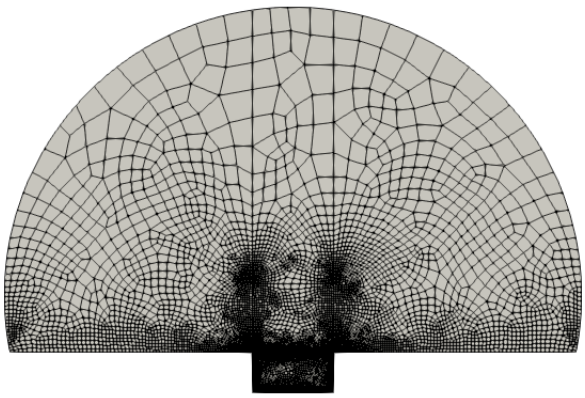


Figure 4.17: Unstructured mesh used in the flow past an open cavity example containing 9520 total Q_2 elements.

extends vertically upward from the cavity, and horizontally at each side of the cavity, so that a portion of the radiated acoustic field is solved. The Cartesian distribution of the domain is typically used for finite-difference solvers. Instead, our compressible flow solver is capable of using unstructured meshes corresponding to complex geometries, and also to include high order interpolations. In this sense, we solve this problem using a curved far-field boundary, which is sufficiently apart from the cavity, so that a portion of the radiated acoustic field can be resolved. The curvature of the far-field boundary is designed to allow for normal impinging of the scattered sound waves. Setting the origin at the top-center of the cavity, the outer boundary is defined by the circumference that passes through the points $(7D, 0)$, $(-7D, 0)$, and $(0, 8D)$. For this problem, we consider the M $= 0.6$ Mach number condition of the free stream flow, and a Re $= 41000$ Reynolds number condition (based on the cavity depth) that has been studied by several authors. We set the sound speed to 408 m/s, the free stream air temperature to 408 K, and the pressure to 1.38 atm in order to accomplish the compressible condition with a free stream velocity of $(245, 0)$ m/s. The Reynolds and Prandtl numbers, on the other hand, impose a free stream density of 1.16 kg/m^3, a dynamic viscosity of 1.76×10^{-4} kg/(m s), and a thermal conductivity of 0.25 kW/(m K).

The accurate simulation of the acoustic radiation of the cavity is tightly connected with the definition of the boundaries: because the cavity walls are treated as non slip, impermeable, and isothermal (given by the free stream flow temperature), and as the flow is injected at the left-most side of the curved boundary, a thick boundary layer is generated when the flow passes through the lower non-slip wall. The interaction of the flow disturbances inside the cavity and the upstream boundary layer is the main hydrodynamic feedback mechanism in charge of producing the flow-acoustic interaction. Setting accurately the non-reflecting far field boundaries is crucial to assure that wave reflections do not produce further disturbances to the upstream boundary layer, neither to the flow inside the cavity. The non-reflecting outlet coefficients are set to $\sigma = 1$, and $l = 2D$, with the numerical parameter for the weak implicit solution set to $\eta_0 = 10^{-2}$. Simulations are performed on the structured mesh depicted in Fig. 4.17 composed of 9520 bilinear quadrilateral elements using the orthogonal, dynamic and

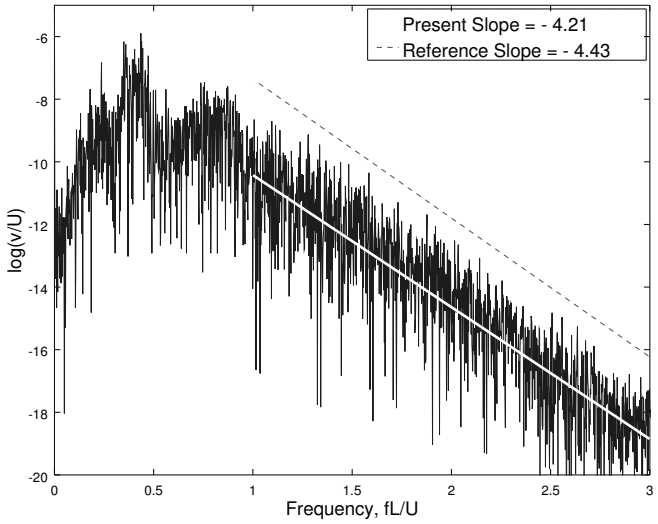
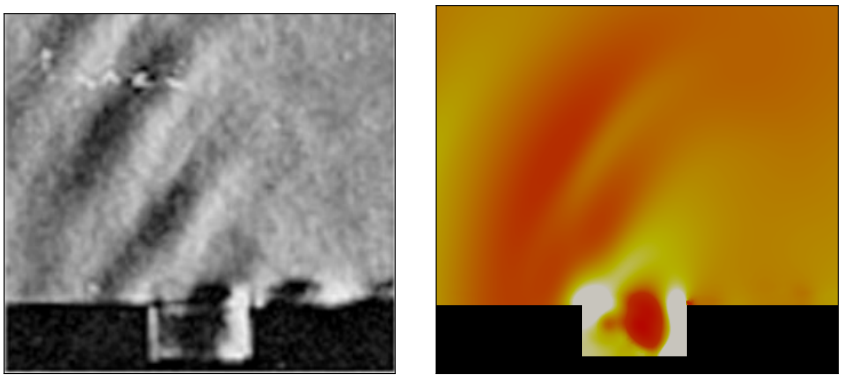


Figure 4.18: Flow past an open cavity: normalized spectrum of the scaled x_2-component of velocity at $(1.57D, 0)$.

non-linear subscales definition for the compressible solver.

The fluctuating character of the flow can be seen in Fig. 4.18, in which the normalized spectrum of the scaled x_2-component of velocity at the point $(1.57D, 0)$ is presented. The frequency in this plot is normalized with respect to the cavity length and the free stream flow velocity. As reported in previous Direct Numerical Simulations (DNS) [117, 118], the present simulation of the flow field is characterized by the circular rolling of vorticity inside the cavity and its unstable impingement over the downstream edge of the cavity. This can be observed as the low-frequency modes in the spectrum correspond to the large-scale vortex frequencies. The development of the vorticity is produced by singular pressure points located at the top corners of the cavity walls (generated by the boundary layer and the steady flow inside the cavity). For the fully developed flow, the flow-acoustic interaction triggers a highly chaotic behavior inside the cavity. Although turbulence is inherently a three-dimensional phenomena, here we refer to the chaotic character of the flow meaning that the flow is unstable and that the flow is composed of very different and active frequencies, as depicted in the spectrum.

The spectrum of Fig. 4.18 is also characterized by a cascade that is similar to that encountered in fluid flow turbulence, containing a range that behaves as $(fL/U)^{-4.2}$. This decay corresponds with the one obtained by the DNS solution in [117] using a sixth-order accurate finite difference solver with half a million grid points. The two lower peaks are also reported in [117]: one at around $fL/U = 0.4$, and another at $fL/U = 0.7$. In this sense, we observe that low-frequency modes correspond well in both simulations to the vortex shedding frequencies and that the present simulation is able to represent the scales of sound using a grid that is two orders of magnitude smaller than the benchmark simulation. The frequency peak reported by [120] for the acoustic radiation measurement using an experimental setting for the same problem definition is $fL/U = 0.8$, which is higher than the present simulation results mainly due to three-dimensional effects.



(a) (b)

Figure 4.19: Flow past an open cavity comparison: (a) Schlieren photographs [120] at M $=$ 0.64, (b) instantaneous contour of the pressure field at M $= 0.6$.

Scattering of acoustic waves can be visualized and compared in Fig. 4.19. In this figure we depict the instantaneous pressure contour obtained in the present simulation, in contrast to the photograph from [120] (depicted at the left side of the figure) showing the structure of the radiated field. There is a qualitative agreement between the present simulation and the experimental setting: visual comparisons of the scattered pressure waves coincide. The sound generated in those pressure waves is involved with the mixing and the non-linear interaction between the very different scales, and consequently, for mesh sizes that cannot resolve all the flow scales (such as the mesh of Fig. 4.17), the production of sound demonstrates a strong sensitivity to the amount of diffusion given by the numerical method [121]. In this sense, the accurate definition of the sound waves given in the present simulation relies only on the numerical diffusion given the VMS stabilization method, without any modification of the continuous problem or the inclusion of any turbulence model for the subgrid scales. This kind of approximation has been recently related to the Implicit LES (ILES) methods [47], which accurately represent the underlying turbulent behavior by the addition of dissipative numerical terms solely, even if the mesh is not too fine to resolve the majority of the flow scales.

It is also important to note that the curved far field boundary does not affect the solution as it damps completely the reflection of the sound wave. The pressure scattering shown in Fig. 4.19 is mainly a combination of direct and reflected acoustic waves, and this agreement cannot be achieved without the use of non-reflecting boundary conditions (or with the inclusion of some damping technique). Spurious oscillations, that lead to instabilities in the flow, also appear in this highly non-linear aeroacoustic problem if some reflection of waves occur. In this sense, we obtain accurate results for a grid size which is composed only by 9520 elements, by applying an accurate approximation of the compressible Navier-Stokes equations and the numerical strategy for damping wave reflections at the computational boundaries.

4.5 Conclusions

In this chapter, a finite element approximation of the compressible Navier-Stokes equations written in primitive variables has been developed. From the numerical point of view, the compressible model has been approximated by a stabilized VMS method, and an implicit scheme has been used to advance in time. Some other ingredients, such as the orthogonal, dynamic and non-linear definition of the subscales, and the weak imposition of non-reflecting boundary conditions for simulating aeroacoustic flows, have been investigated. In particular, the design of the static and transient stabilization matrix, and the decomposition of the pressure and temperature unknowns into a relative and a reference part allow solving nearly incompressible cases.

The accuracy of the method with the mesh size has been verified for two- and three-dimensional linear elements, as well as for two-dimensional quadratic elements, using steady and low Mach manufactured solutions. The differential heated cavity problem and the flow past a cylinder problem have been used to test the accuracy of the method for dynamic cases. It has been observed that including the temporal derivatives of the subscales, and defining the space where the subscales live as the orthogonal to the finite element space, improve the accuracy of the variational method. Finally, the possibility of directly computing the acoustic pressure waves with the compressible Navier-Stokes equations has been validated with the Aeolian tones problem and with the flow past an open cavity problem. Accurate simulations of acoustic radiation have been obtained by solving directly the compressible Navier-Stokes equations and damping wave reflections at the computational boundaries. The weak imposition of non-reflecting boundary conditions allows for the free propagation of acoustic waves within the implicit time stepping scheme.

Chapter 5

Global conservation restrictions of the compressible Navier-Stokes equations written in primitive variables

In this chapter, we apply the interpolation with restrictions concept to compensate the lack of conservativity of the compressible Navier-Stokes equations written in primitive variables. The main idea is to ensure the global conservation of mass, momentum, and total energy through the solution of a small optimization problem on the primitive spatial solution. The optimization problem arises from the enforcement of the conservative fluxes over the boundaries and of the forces inside the domain, on a new globally conserved solution using Lagrange multipliers.

5.1 Introduction

The conservative form of hyperbolic equations is the form that admits physically meaningful solutions when they develop discontinuities. This is due to the fact that the divergence of the fluxes is balanced over any control volume, so that, the conservative property of quantities holds even for local discontinuities of the solution. The compressible Navier-Stokes equations written in conservative form in the inviscid limit is a hyperbolic system that fulfills the conservative character described before. Hyperbolic conservation laws often describe other engineering problems; concretely: combustion, shallow waters, or magneto-hydrodynamics, are described by hyperbolic systems in several space variables.

A rigorous analysis of the conservative character of numerical methods applied to the hyperbolic conservation laws began with the early work by Lax and Wendroff [122], where they proved that every piece-wise continuous weak solution of the variational form of the hyperbolic conservation problem must satisfy the Rankine-Hugoniot relation across a line of discontinuity, and therefore that, if a conservative numerical method converges, it does so to a weak solution of the hyperbolic system. Another substantial contribution about hyperbolic conservation laws in several space variables, specifically in the case of the compressible Navier-Stokes equations, was achieved in [4, 123], where it was demonstrated that jump discontinuities in solutions follow characteristic hyper-surfaces up to an instant when the existence and structural stability of discontinuous solutions transported through the characteristics deteriorate.

However, even for the conservative form of the compressible Navier-Stokes equations, some numerical methods fail to guarantee the conservation of mass, momentum and total energy: that is, for example, the case of discontinuous Galerkin approximations. Due to the discontinuous function approximation of this method, flux terms are not uniquely defined at the finite element interfaces, and the numerical conservation may degrade leading to wrong shock strength and propagation speeds. To overcome this issue, several types of reconstruction schemes have raised. The literature of reconstruction schemes for the fluxes is too vast to be surveyed here, but we refer to the local decomposition of the physical variables onto the respective characteristic space in [124–126], where the characteristics are reconstructed and projected back to the physical space, and the high-order accurate approximations to the Riemann problem in [127], being the most used schemes in finite differences, finite volumes, and discontinuous Galerkin methods.

A particular problem with the performance of the compressible flow solvers based on the Navier-Stokes equations written in conservative form arises at low Mach flow regimes: at these regimes, the numerical approximation can be inaccurate. But that is not the case for the compressible flow solver based on the primitive variables formulation, which is accurate in the zero Mach limit as it corresponds to the incompressible Navier-Stokes equations. Nevertheless, when the divergence of the (convective and viscous) fluxes is written as a term involving the spatial derivatives over the unknowns, the formulation may not admit physically meaningful solutions in the case of discontinuities; this is due to the fact that the derivative of discontinuous solutions is not defined. In other words, the conservative character of the system of equations is modified and so, the solution of supersonic flow problems is degraded (in the case of the primitive variables formulation of the compressible Navier-Stokes equations). Therefore, for the compressible flow equations written in primitive variables, we want to enforce the *conservation of mass, conservation of momentum, and conservation of total energy*.

The re-meshing strategy for flow problems in an Arbitrary Lagrangian Eulerian (ALE) setting presented in [128], which enforces that the interpolated fields on a new mesh conserve relevant physical properties (like mass and kinetic energy), is the starting point towards a concept that may allow correcting non-conservative flow solutions. The conservative correction was used in that original work to compensate a diffusive character of the standard Lagrangian interpolations and was applied *a posteriori* by solving a small optimization problem with restrictions using Lagrange multipliers (based on [129]). The computational cost of this methodology was negligible in contrast to the flow problem solution.

In this work, we extend the interpolation with restrictions idea to compensate the non-conservative character of the compressible flow formulations based on primitive variables and so, to guarantee the global conservation of physical quantities like mass, momentum, and total energy. The objective of this correction is to allow the primitive variables formulation to accurately solve jump discontinuities in the solution arising from supersonic regimes. As commented before, this idea follows from the fact that solving the (cheap) global optimization problem of conservative restrictions, may lead to the extension of the flow regimes in which the primitive variables formulation is accurate, incurring in a negligible increment in the computational cost.

One problem that arises is that the restriction must be satisfied over the variables of the conservative formulation: the strong coupling between the compressible problem variables requires the correction of the mass, momentum, and total energy. For the incompressible Navier-

Stokes equations in [128] the conservation of relevant magnitudes only affected the velocity, and therefore, restrictions were applied only to the interpolation of that physical quantity. Instead, the optimization problem with restrictions of the present formulation is applied separately to each component of the vector of conservative variables. The ideal approach would consist in coupling the restrictions over all the variables of the compressible problem, so that, a minimization of the overall distance between the calculated and the corrected conservative solutions can be included. However, the construction of such a norm, with an appropriate scaling between the different variables, is not straightforward.

The chapter is organized as follows. In Section 5.2, we present the concept of interpolation with restrictions applied to the global conservative correction for the Navier-Stokes equations written in primitive variables. First, we describe the conservation of mass, momentum, and total energy provided by the compressible Navier-Stokes equations written in conservative form. Then, we introduce the concept of interpolation with restrictions and apply the global conservative restriction to the primitive variables solution. So that, in Section 5.3 we test the numerical method for one- and two- dimensional applications, including inviscid and viscous supersonic flows. Finally, we address some conclusions in Section 5.4.

5.2 Global conservation restrictions formulation

As commented before, it is readily known that the discrete compressible flow formulation written in primitive variables, namely $\boldsymbol{Y} = (p, \boldsymbol{u}, T)^\top$, being the pressure p, the velocity $\boldsymbol{u}$, and the temperature T, is not conservative. The fact that some terms of the primitive formulation (i.e., the convective fluxes), are not equal to the ones of the conservative variables formulation, arises from the existence of spatial derivatives applied to the primitive variables solution. In the case of supersonic shocks, chemical reactions, or sharp interfaces between different flows, a discontinuity in the solution appears, and those derivatives become undefined with the consequent deterioration of the physical quantities.

5.2.1 Conservation of quantities

Let us first recall the conservative variables formulation of the compressible Navier-Stokes equations (presented in Chapter 3). We define the time interval $(0, t_f)$ and the domain $\Omega \subset \mathbb{R}^d$, being d the number of space dimensions ($d = 2$ or 3). Let $t \in (0, t_f)$ be a given time instant in the temporal domain, and $\boldsymbol{x} \in \Omega$ a given point in the spatial domain. Let Γ be the boundary of the domain Ω, and $\boldsymbol{n}$ the geometric unit outward normal vector on Γ. Using the usual summation convention of the indices running from 1 to d, the conservation of mass, momentum, and total energy given by the conservative form of the compressible equations in (3.6) can be written in system form as

$$\frac{d}{dt}\int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}\right)\, d\Omega = -\int_\Gamma n_j \boldsymbol{Q}_j\left(\boldsymbol{U}\left(\boldsymbol{Y}\right)\right)\, d\Gamma + \int_\Omega \boldsymbol{F}\, d\Omega. \tag{5.1}$$

For conciseness, here we have defined the conserved quantities, which are the so-called conservative variables $\boldsymbol{U} = (\rho, \boldsymbol{m}, e_{\text{tot}})^\top$, that can be calculated from the primitive variables (if wished); indeed, ρ is the density, $\boldsymbol{m} = \rho\boldsymbol{u}$ is the momentum, and e_{tot} is the total energy which

is defined as $e_{\text{tot}} = \rho\left(e + \boldsymbol{u}\cdot\boldsymbol{u}/2\right)$, where $\boldsymbol{u}$ is the velocity and e is the internal energy. In the previous equation, the total fluxes across the boundary are defined as

$$\boldsymbol{Q}_j\left(\boldsymbol{U}\left(\boldsymbol{Y}\right)\right) = \boldsymbol{E}_j\left(\boldsymbol{U}\left(\boldsymbol{Y}\right)\right) + \boldsymbol{G}_j\left(\boldsymbol{U}\left(\boldsymbol{Y}\right)\right),$$

with $\boldsymbol{E}_j$ denoting the convective fluxes in the $j-$th direction, and $\boldsymbol{G}_j$ as the diffusive fluxes, also in the $j-$th direction. The forces are defined in vector form using the notation $\boldsymbol{F}$.

Let us now introduce the discretization of time. We partition the time interval $(0, t_f)$ in a sequence of discrete time steps $0 = t^0 < t^1 < ... < t^N = t_f$, with $\delta t > 0$ the time step size, being $t^{n+1} = t^n + \delta t$ for $n = 0, 1, 2, ..., N$. Following the Fundamental Theorem of Calculus, the integration of (5.1) in the time interval (t^n, t^{n+1}) implies that the conservative variables at t^{n+1} satisfy

$$\int_{t^n}^{t^{n+1}} \frac{d}{dt}\int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}\right)\, d\Omega\, dt = \int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}^{t^{n+1}}\right)\, d\Omega - \int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}^{t^n}\right)\, d\Omega. \tag{5.2}$$

Therefore, equation (5.1) can be written as

$$\begin{aligned}\int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}^{t^{n+1}}\right)\, d\Omega - \int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}^{t^n}\right)\, d\Omega = &-\int_{t^n}^{t^{n+1}}\int_\Gamma n_j \boldsymbol{Q}_j\left(\boldsymbol{U}\left(\boldsymbol{Y}\right)\right)\, d\Gamma\, dt \\ &+ \int_{t^n}^{t^{n+1}}\int_\Omega \boldsymbol{F}\, d\Omega\, dt.\end{aligned} \tag{5.3}$$

The previous equation is only subject to the temporal integration scheme. In this work we integrate in the discrete time step the right-hand-side of the previous equation by using the family of Adams-Bashforth methods of order $k = 1, 2, ..$, given by

$$\begin{aligned}\int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}^{n+1}\right)\, d\Omega = &\int_\Omega \boldsymbol{U}\left(\boldsymbol{Y}^{n}\right)\, d\Omega - \delta t\sum_{s=0}^{k-1}\xi_{ks}\int_\Gamma n_j \boldsymbol{Q}_j\left(\boldsymbol{U}\left(\boldsymbol{Y}^{n-s}\right)\right)\, d\Gamma \\ &+ \delta t\sum_{s=0}^{k-1}\xi_{ks}\int_\Omega \boldsymbol{F}^{n-s}\, d\Omega + \mathcal{O}\left(\delta t^{k+1}\right)\end{aligned} \tag{5.4}$$

where ξ_{ks} are numerical parameters depending on the scheme. Specifically, the first order approximation $k = 1$ leads to the scheme with $\xi_{10} = 1$. The second order $k = 2$ method is obtained with $\xi_{20} = 3/2$, and $\xi_{21} = -1/2$ coefficients.

5.2.2 Restrictions

Now, we describe the restrictions concept at an abstract level. Suppose a solution $\widehat{u}_h$ of a finite element problem that is defined in the functional space $\mathcal{V}_h$ over the finite element partition $\mathcal{T}_h = \{K\}$. This solution is intended to satisfy two important features:

- It must remain the nearest solution in the L^2-norm to a solution u_h, being u_h, for example, the calculated non-conservative solution.
- It must fulfill the conservation of a set of quantities. For example, it must globally conserve the physical quantities involved in the solution u_h of a previous time step.

For imposing the second feature, let us define the restriction operators (or forms). When applied to the finite element functions, these operators give a scalar result corresponding to a relevant magnitude of the physical problem being calculated. We write them as

$$R_k : \mathcal{V}_h \longrightarrow \mathbb{R}, \tag{5.5}$$

where the subindex $k = 1, ..., m$ refers to a restriction counter. The linear type of operators R_k can be written in terms of their nodal values, so that

$$R_k(u_h) = \sum_a R_k^a \mathsf{U}^a,$$

where U^a denotes the nodal values associated to u_h, and $R_k^a = R_k(N^a)$ are the nodal values of the restrictions, being N^a the shape functions of $\mathcal{T}_h$. In the previous relation, the superscript a refers to the nodes and runs from 1 to the number of nodes of the mesh.

Since it has been stated that the restriction consists in conserving values of these selected magnitudes, the following equalities must hold:

$$\sum_a R_k^a \widehat{\mathsf{U}}^a = \sum_b R_k^b \mathsf{U}^b, \quad k = 1, \ldots, m, \tag{5.6}$$

where $\widehat{\mathsf{U}}^a$ denotes the nodal values associated to $\widehat{u}_h$. Then, the restrictions can be enforced through Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m) \in \mathbb{R}^m$ and the following functional $L : \mathcal{V}_h \times \mathbb{R}^m \longrightarrow \mathbb{R}$ can be defined:

$$L(v_h, \boldsymbol{\mu}) = \frac{1}{2} \left\| \sum_a N_2^a (\mathsf{V}^a - \mathsf{U}^a) \right\|^2_{L^2(\Omega)} - \sum_{k=1}^m \mu_k \left(\sum_a R_k^a \mathsf{V}^a - \sum_b R_k^b \mathsf{U}^b \right), \tag{5.7}$$

where V^a are the nodal values of v_h, $\boldsymbol{\mu} \in \mathbb{R}^m$ is an admissible set of Lagrange multipliers, and the subscript in the norm indicates that it is the $L^2(\Omega)-$norm. The solution that minimizes the distance to u_h, and imposes the restrictions, will be:

$$[\widehat{u}_h, \boldsymbol{\lambda}] = \arg \inf_{v_h \in \mathcal{V}_h} \sup_{\boldsymbol{\mu} \in \mathbb{R}^m} \left[L\left(v_h, \boldsymbol{\mu}\right) \right]. \tag{5.8}$$

The previous problem is a saddle point problem. A necessary and sufficient condition for it to be well posed is that the finite element space $\mathcal{V}_h$ and $\mathbb{R}^m$ satisfy the appropriate inf-sup condition. This, in particular, restricts the number of Lagrange multipliers. However, since the number of restrictions m is usually very small, no stability problems are expected.

The equations to be solved are obtained by differentiation of the functional L with respect to the unknowns:

$$\frac{\partial L}{\partial \mathsf{V}^b} = 0 \Longrightarrow \sum_a \int_\Omega N^b N^a \widehat{\mathsf{U}}^a - \sum_{i=1}^m \lambda_k R_k^b = \sum_a \int_\Omega N^b N^a \mathsf{U}^a, \quad \text{for all } b, \tag{5.9}$$

$$\frac{\partial L}{\partial \mu_k} = 0 \Longrightarrow \sum_a R_k^a \widehat{\mathsf{U}}^a = \sum_b R_k^b \mathsf{U}^b, \quad \text{for all } k. \tag{5.10}$$

5.2.3 Conservation restrictions

The formulation presented in the previous paragraphs is now extended to the conservation problem of the compressible Navier-Stokes equations written in primitive variables.

As the main hypothesis of the method, we suppose that a conservative solution $\boldsymbol{U}$ is given for the compressible flow problem. Most of the cases this solution is given in the form of an initial condition.

Say that we obtain a primitive solution $\boldsymbol{Y}_h^{n+1}$ at time step $n+1$, which has been computed with the compressible Navier-Stokes equations written in primitive variables.

We can calculate a discrete version of $\boldsymbol{U}_h^{n+1}$ from the primitive solution $\boldsymbol{U}_h^{n+1} = \boldsymbol{U}_h^{n+1}(\boldsymbol{Y}_h^{n+1})$, but it is clear that this solution does not hold the conservative properties inherent to the conservative variables formulation.

Following the same notation as before, and denoting $\widehat{\boldsymbol{U}}_h^{n+1}$ as the corrected solution of $\boldsymbol{U}_h^{n+1}$ on $\mathcal{T}_h$, a correction of the $i-$th component of the calculated solution might consist in the minimization of the functional (5.7) subject to the conservation restrictions. In other words, the objective is to construct a $\widehat{U}_{h,i}^{n+1}$, an approximation to $U_{h,i}^{n+1}$, satisfying the minimization of the distance in the L^2-norm, and subject to the conservation properties of the (actual) conservative solution $U_{h,i}^n$ at the previous time step n. The conservation restriction that is applied to $U_{h,i}^{n+1}$ can be written as,

$$\begin{aligned}\int_\Omega \widehat{U}_{h,i}^{n+1}\, d\Omega = \int_\Omega U_{h,i}^n\, d\Omega - \delta t \sum_{s=0}^{k-1} \xi_{ks} \int_\Gamma \left[n_j \boldsymbol{Q}_j \left(\boldsymbol{U}_h^{n-s} \right) \right]_i \, d\Gamma \\ + \delta t \sum_{s=0}^{k-1} \xi_{ks} \int_\Omega F_i^{n-s}\, d\Omega, \end{aligned} \tag{5.11}$$

for all $i = 1, ..., d+2$.

Denoting by U_i^a the corresponding nodal values of the $i-$th variable, and assuming a standard Lagrangian interpolation, we write the calculated solution as $U_{h,i}^{n+1} = \sum_a N^a \mathsf{U}_i^{a,n+1}$, and the corrected solution as $\widehat{U}_{h,i}^{n+1} = \sum_a N^a \widehat{\mathsf{U}}_i^{a,n+1}$, for all $i = 1, ..., d+2$.

Let $\lambda_i, i = 1, ..., d+2$, be the Lagrange multipliers vector to enforce the conservative variables in (5.11). In this case, the Lagrangian functional to be minimized is given by

$$\begin{aligned} L_i \left(\widehat{U}_{h,i}^{n+1}, \lambda_i \right) =& \frac{1}{2} \int_\Omega \left(\sum_a N^a \left(\widehat{\mathsf{U}}_i^{a,n+1} - \mathsf{U}_i^{a,n+1} \right) \right)^2 \, d\Omega \\ & - \lambda_i \int_\Omega \left(\sum_a N^a \left(\widehat{\mathsf{U}}_i^{a,n+1} - \mathsf{U}_i^{a,n} \right) \right) \, d\Omega \\ & + \lambda_i \delta t \sum_{s=0}^{k-1} \xi_{ks} \int_\Gamma \left[\sum_{j=1}^{d} n_j \boldsymbol{Q}_j \left(\boldsymbol{U}_h^{n-s} \right) \right]_i \, d\Gamma \\ & - \lambda_i \delta t \sum_{s=0}^{k-1} \xi_{ks} \int_\Omega F_i^{n-s}\, d\Omega, \end{aligned} \tag{5.12}$$

for all $i = 1, ..., d+2$.

The derivatives of L with respect to the nodal unknowns and the Lagrange multipliers must be now calculated to obtain the optimality conditions. The optimality conditions for this case are

$$\frac{\partial L_i}{\partial \widehat{\mathsf{U}}_i^{b,n+1}} = 0, \qquad \text{and} \qquad \frac{\partial L_i}{\partial \lambda_i} = 0. \tag{5.13}$$

In the following development, we seek for the solution for each optimality condition. The first optimality condition gives for each $i-$th component, ranging from 1 to $d+2$, as

$$\int_\Omega \sum_a N^b N^a \widehat{\mathsf{U}}_i^{a,n+1}\, d\Omega - \lambda_i \int_\Omega N^b\, d\Omega = \int_\Omega \sum_a N^b N^a \mathsf{U}_i^{a,n+1}\, d\Omega, \tag{5.14}$$

for all nodes b. Whereas, the second optimality condition reads for each one of those $i-$th components:

$$\begin{aligned}\int_\Omega \sum_a N^a \widehat{\mathsf{U}}_i^{a,n+1}\, d\Omega = &\int_\Omega \sum_a N^a \mathsf{U}_i^{a,n}\, d\Omega \\ &+ \delta t \sum_{s=0}^{k-1} \xi_{ks} \int_\Gamma \left[\sum_{j=1}^{d} n_j \boldsymbol{Q}_j\left(\boldsymbol{U}_h^{n-s}\right)\right]_i d\Gamma - \delta t \sum_{s=0}^{k-1} \xi_{ks} \int_\Omega F_i^{n-s}\, d\Omega.\end{aligned} \tag{5.15}$$

Using the bold notation as a short hand notation for the nodal arrays, these two equations can be arranged into the following algebraic system for each $i-$th component:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{R}_i^\top \\ \mathbf{R}_i & 0 \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{U}}_i^{n+1} \\ \lambda_i \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{U}_i^{n+1} \\ \mathbf{R}_i\mathbf{U}_i^n + \delta t \sum_{s=0}^{k-1} \xi_{ks} \mathrm{T}_i^{n-s} \end{bmatrix} \tag{5.16}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{R}_i$ stands for the restriction in the left-hand-side of (5.15), and T_i^{n-s} comes from the integration of the fluxes and forces in (5.15). We can avoid solving the full linear system by writing the Schur complement problem for the Lagrange multiplier, and later computing the nodal values $\widehat{\mathbf{U}}_i^{n+1}$:

$$\mathbf{R}_i\mathbf{M}^{-1}\mathbf{R}_i^\top \lambda_i = \mathbf{R}_i\mathbf{U}_i^n + \delta t \sum_{s=0}^{k-1} \xi_{ks} \mathrm{T}_i^{n-s} - \mathbf{R}_i\mathbf{U}_i^{n+1} \tag{5.17}$$

$$\widehat{\mathbf{U}}_i^{n+1} = \mathbf{U}_i^{n+1} + \mathbf{M}^{-1}\mathbf{R}_i^\top \lambda_i \tag{5.18}$$

Equation (5.17) is a scalar equation, with the only difficulty of calculating $\mathbf{M}^{-1}$.

Finally, we can recover a conserved primitive solution $\widehat{\boldsymbol{Y}}_h^{n+1}$ computed from the restricted solution $\widehat{\boldsymbol{U}}_h^{n+1}$ in (5.18), which can be used to advance in time with the primitive variables compressible Navier-Stokes formulation. The restricted solution, satisfying the global conservation of physical quantities, is supposed as the conservative solution at the next time step (in this numerical technique).

5.3 Numerical examples

The previously exposed numerical methodology is tested in this section. We first solve a one-dimensional inviscid shock tube that makes it possible to compare the obtained results against exact solutions. Then, we solve the two-dimensional inviscid shock reflection problem. Finally, we investigate the effect of the conservation restriction solving the viscous supersonic flow past a compression corner.

In the following numerical examples the flow is considered as an ideal gas, with a compressibility condition of $\gamma = 1.4$, and physical properties $c_p = 1.010$ kJ/(kg K) and $c_v = 0.718$ kJ/(kg K).

5.3.1 One-dimensional shock tube

The spatial domain for the transient one-dimensional problem is $x \in [0, 1]$ m. The initial condition is set as follows: two different uniform states for the fluid are separated by a diaphragm at 0.5 m. The flow is initially at rest $u(x, t = 0) = 0$ m/s. At the left part of the diaphragm the fluid has an uniform density $\rho_L = 1$ kg/m^3, a temperature of $T_L = 3.481 \times 10^{-3}$ K, and a pressure of $P_L = 1$ Pa. At the right part of the diaphragm, the fluid initiates with an uniform density $\rho_R = 0.125$ kg/m^3, a temperature $T_R = 2.785 \times 10^{-3}$ K, and a pressure $P_R = 0.1$ Pa. The time interval to be analyzed finishes when the fluid expansion affects the boundary conditions at $x = 0$ m and $x = 1$ m. Specifically, the results are reported at $t = 0.2$ s of simulation. The mesh used in the simulations has a fixed size of $h = 0.01$ m.

Figure 5.1 shows the results for the conservative variables, primitive variables, and primitive variables with conservation restrictions formulations at $t = 0.2$ s. It can be observed that the solution obtained with the conservative variables formulation matches accurately the exact solution. As expected, the propagation of the initial shock cannot be described by the compressible Navier-Stokes formulation in primitive variables. We observe that the corrected solution is wrong, matching the primitive variables solution in almost the entire domain, except for the boundaries. The global conservation given by the proposed method is not able to correct the wrong solution given by the primitive variables formulation. We can conclude that neither the primitive variables formulation nor the primitive solution coupled with the conservation restrictions is accurate.

We further investigate the amount of correction given by the conservation restrictions methodology for each primitive variable at the final time step. This is displayed in Fig. 5.2, where we observe that, although the correction is following the propagation of the flow, it is adding a very small quantity to the primitive variables solution. We see that most of the correction is taking place near the boundaries, where the conserved solution seems to give a slightly better approximation than the primitive variables solution alone.

We also present the global amount of correction in Table 5.1, where the global quantities are written for the reference conservative solution $\boldsymbol{U}_h^n$, the solution obtained with the primitive variables formulation $\boldsymbol{U}_h^{n+1}$, and the conserved solution after the conservation restriction is applied $\widehat{\boldsymbol{U}}_h^{n+1}$. We can conclude that the restriction actually corrects in a global sense the physical quantities, but it is clear that this global correction is not enough to improve the primitive formulation accuracy in the case of inviscid flows with supersonic shocks.

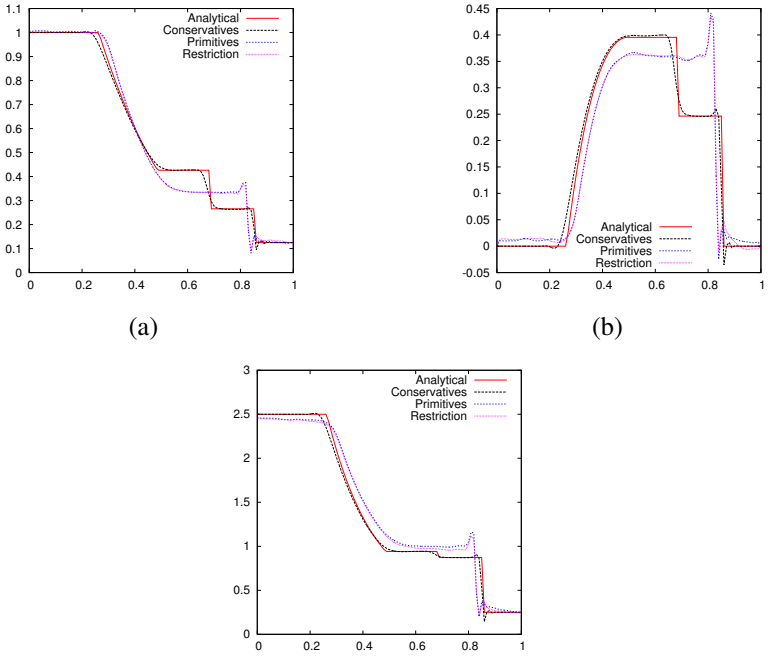


Figure 5.1: One-dimensional shock tube results: (a) density, (b) momentum, and (c) total energy at 0.2 s.

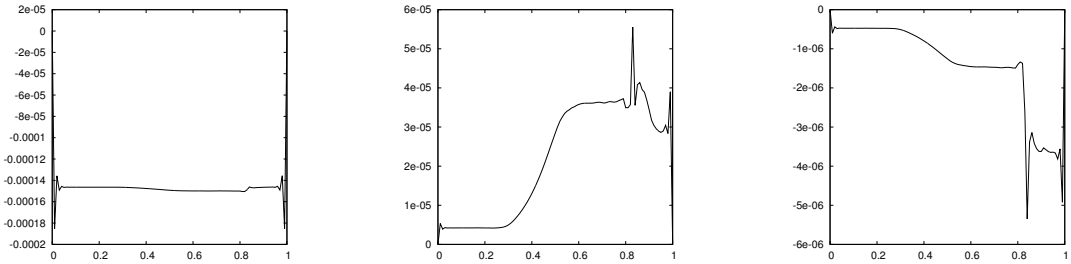


Figure 5.2: One-dimensional shock tube results. Correction of pressure (left), velocity (middle), and temperature (right) at $0,2$ s.

Table 5.1: One-dimensional shock tube results.

Component	$\int_\Omega \boldsymbol{U}_h^n \, d\Omega$	$\int_\Omega \boldsymbol{U}_h^{n+1} \, d\Omega$	$\int_\Omega \widehat{\boldsymbol{U}}_h^{n+1} \, d\Omega$
ρ	5.590647×10^{-2}	5.590786×10^{-2}	5.590647×10^{-2}
m	1.802556×10^{-2}	1.802478×10^{-2}	1.802556×10^{-2}
e_{tot}	0.141177	0.141245	0.141177

5.3.2 Supersonic inviscid shock reflection

The supersonic reflected shock problem is another inviscid supersonic problem in which the conservation restrictions formulation can be tested. The problem domain is $[0, L] \times$

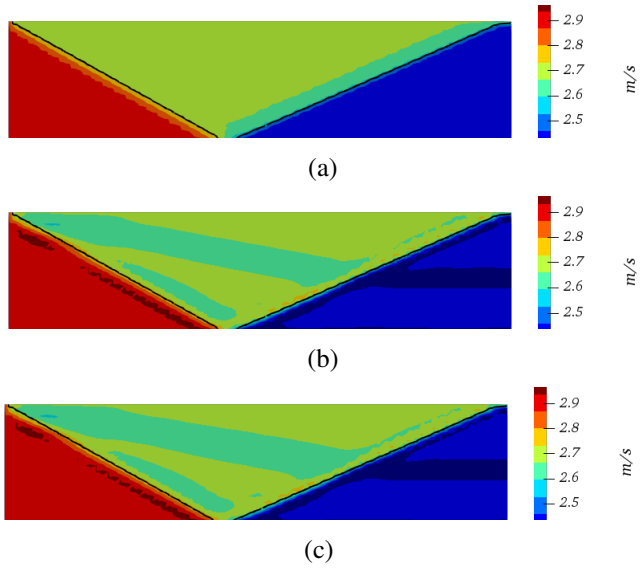


Figure 5.3: Inviscid shock reflection results. Velocity magnitude contour solved with the (a) conservative variables formulation, (b) primitive variables formulation, and (d) conservation restrictions of the primitive variables formulation.

$[-H/2, H/2]$, with $L = 4.1$ m and $H = 0.5$ m. The flow is injected from the left and upper walls. Over the inlet left wall the Mach number is set to M $= 3$, hence, fixed values of velocity $(2.9, 0)$ m/s, pressure 0.7212 Pa and temperature 0.00247 K are set. Over the upper inlet wall the Mach number is M $= 2.378$, thus, fixed values of velocity $(2.6193, -0.5063)$ m/s, pressure 1.543 Pa and temperature 0.00311 K are also set. A slip condition for the velocity and an adiabatic condition for energy are set to the lower wall. Over the outflow wall, no conditions need to be imposed because the flow is supersonic. The values of viscosity and conductivity are zero. All simulations are run in a structured mesh composed of 3280 P_1 elements until the steady state is reached.

Figure 5.3 displays the steady velocity magnitude solution obtained with the conservative variables formulation, which has been also benchmarked against the reference solutions in [73, 79, 82, 83]. In that figure, we also present the velocity magnitude solution obtained with the primitive variables formulation, and with the conservation restrictions method. We can observe a correct description of the main shock given by the primitive variables formulation, with a slight deterioration in the solution produced by an unrealistic transverse shock. In this sense, we ratify for this two-dimensional inviscid case, that the conservation restrictions method is not enough to improve the ability of the primitive variables formulation to solve supersonic shocks.

The conservation enforcement for each primitive variable can be appreciated in Fig. 5.4, in which the contours of each correction are plotted. Since no diffusive fluxes occur for this inviscid case, the correction is only related to the convective fluxes. We can appreciate in this numerical example that, although the correction is acting mostly at the inlet and outlet boundaries and not over the solid bottom boundary (corresponding to the convective fluxes), a very small quantity of correction is attained in general.

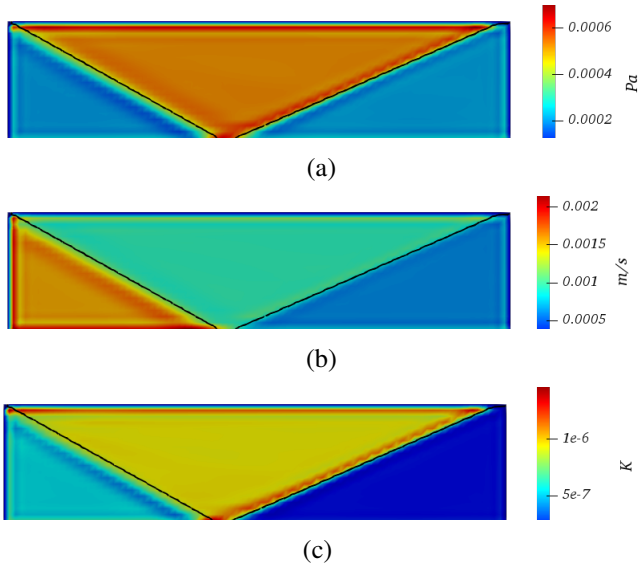


Figure 5.4: Inviscid shock reflection results. Correction for (a) pressure, (b) velocity magnitude, and (c) temperature.

5.3.3 Supersonic flow over a compression corner

As a final numerical example, we solve the $10°$ compression corner at M $= 3$ and Re $= 16800$, being a supersonic viscous flow problem that also involves the formation of a boundary layer. The compression corner problem is described as follows. The leading edge of the corner is placed at the origin and the vertex is located at $(1, 0)$ m. The left inflow boundary is located at $(-0.2, x_2)$ m, of the leading edge of the plate. Symmetry conditions are imposed for all variables over the lower upstream wall $(x_1 < 0, x_2 = 0)$ m. The upper boundary is located at $(x_1, 0.575)$ m, and the exit boundary at $(1.8, x_2)$ m. All the flow variables are prescribed at the inlet and upper boundaries. Those are: velocity $(3, 0)$ m/s, density 1 kg/m^3 and temperature 0.0024 K. On the corner edges, no-slip condition for velocity, a stagnation temperature for energy and a zero flux condition for density are set. The stagnation temperature $\theta_0/\theta_\infty = 1 + (\gamma - 1)\text{M}^2/2$, is calculated to be 0.007 K. At the outflow boundaries, free conditions are set. Viscosity and conductivity are 1.7×10^{-4} kg/(m s) and 0.254 kJ/(m s K), respectively. The finite element mesh consists of a structured non-symmetric mesh composed by 22914 P_1 elements. All simulations are run until the steady state is reached.

Figure 5.5 shows the steady state results; we present the pressure, velocity magnitude, and temperature contours obtained with the primitive variables formulation and with the conservation restrictions formulation. We identify the viscous boundary layer near the solid walls, the supersonic shock forming at the upstream, and the compression shock, all of these described by both formulations. In this case, we contrast the resulting contours against the conservative solution by suggesting the conservative position for the supersonic shock. We observe that the conservation restrictions method is able to slightly improve the solution, especially for the temperature variable.

The amount of correction in this viscous case can be appreciated in Fig. 5.6, where it is more evident that the correction is located near the non-slip walls. The correction for the

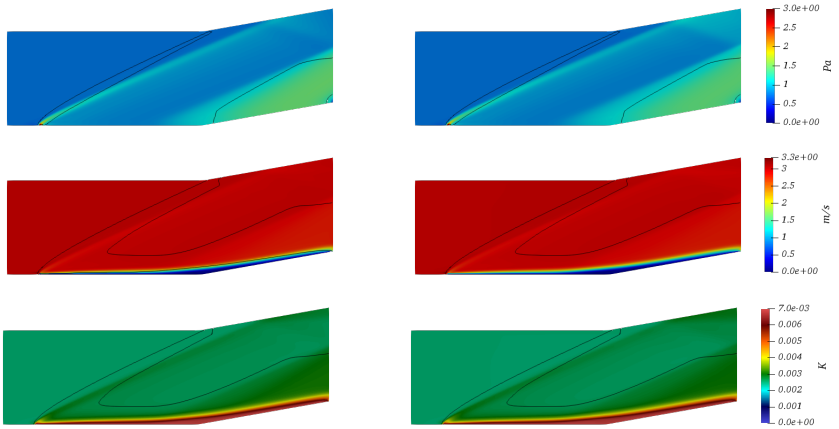


Figure 5.5: Viscid compression corner results. Top: pressure contour. Middle: velocity magnitude contour. Bottom: temperature contour. Solution is obtained using the primitives variables formulation at the left, and the conservation restrictions for the primitive variables formulation at the right.

pressure is carried out mainly near the flow boundaries. Instead, for velocity and temperature variables the correction is mostly acting at the boundary layer, and near the supersonic outflow. This is also the reason for which the temperature solution is slightly more accurate with the conservation restrictions method.

Finally, for this numerical example, we also quantify the global correction and present it in Table 5.2. Again, we observe that, even though the method is not able to improve the accuracy of the primitive variables formulation, the conservation restrictions are indeed able to globally impose the conservation of physical quantities; globally, the corrected solution $\widehat{\boldsymbol{U}}_h^{n+1}$ satisfies exactly the amount of physical quantities in the conservation solution $\widehat{\boldsymbol{U}}_h^n$.

Table 5.2: Viscid compression corner results.

Component	$\int_\Omega \boldsymbol{U}_h^n \, d\Omega$	$\int_\Omega \boldsymbol{U}_h^{n+1} \, d\Omega$	$\int_\Omega \widehat{\boldsymbol{U}}_h^{n+1} \, d\Omega$
ρ	1.253810	1.253811	1.2538101
m_1	3.619579	3.619564	3.619579
m_2	0.14571269	0.14571267	0.14571269
e_{tot}	7.814908	7.814960	7.814908

5.4 Conclusions

In this chapter, we have applied global conservation restrictions to the compressible flow formulation based on primitive variables. We have imposed the global conservation of mass, momentum, and total energy, so that, a small optimization problem involving the primitive solution and a conserved given solution must be solved. The main objective of this correction

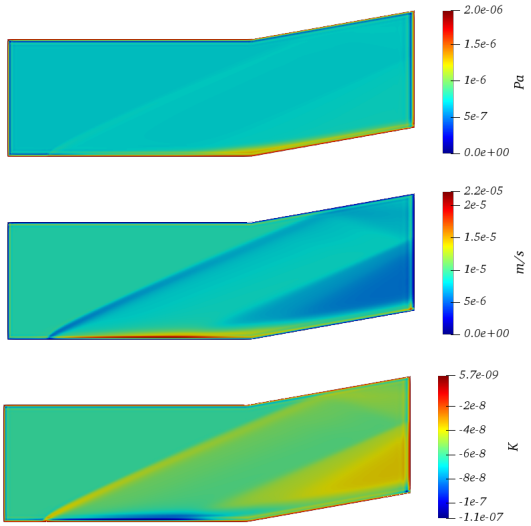


Figure 5.6: Viscid compression corner results. Correction for pressure (top), velocity magnitude (middle), and temperature (bottom). The position of the conservative shock is depicted with a solid line.

has been to allow the primitive variables formulation to accurately solve jump discontinuities in the solution arising from supersonic regimes, but also to avoid a significant increment in the computational cost accomplishing this objective.

Several numerical tests lead us to the conclusion that the present methodology actually makes the global correction of the physical quantities, but that this global correction is not enough to improve the primitive variables formulation accuracy in the case of supersonic shocks.

Consequently, as a future work, we plan to extend this formulation in order to be able to overcome the problems that we have encountered in the numerical tests. For this goal, we first plan to test the introduction of a scaling matrix $\boldsymbol{S}$, that may lead to dimensionally consistent measurements, so that the functional can be written in terms of a scaled L^2-norm of the type $\|\boldsymbol{U}\|_{\boldsymbol{S}}^2 = \int_\Omega (\boldsymbol{U}^\top \boldsymbol{S}\boldsymbol{U})\, d\Omega$. This leads to the possibility of coupling the conservative variables inside the minimization functional, so that, the Lagrangian functional may be given by

$$
\begin{aligned}
L\left(\widehat{\boldsymbol{U}}_h^{n+1}, \lambda_i\right) =& \frac{1}{2}\left\| \sum_a N^a \left(\widehat{\mathbf{U}}^{a,n+1} - \mathbf{U}^{a,n+1}\right)\right\|_{\boldsymbol{S}}^2 \\
&- \sum_{i=1}^{d+2} \lambda_i \int_\Omega \left(\sum_a N^a \left(\widehat{\mathsf{U}}_i^{a,n+1} - \mathsf{U}_i^{a,n}\right)\right) d\Omega \\
&+ \sum_{i=1}^{d+2}\sum_{s=0}^{k-1} \lambda_i \delta t \xi_{ks} \int_{\partial\Omega} \left[\sum_{j=1}^{d} n_j \boldsymbol{Q}_j\left(\boldsymbol{U}_h^{n-s}\right)\right]_i d\Gamma \\
&- \sum_{i=1}^{d+2}\sum_{s=0}^{k-1} \lambda_i \delta t \xi_{ks} \int_\Omega F_i^{n-s}\, d\Omega,
\end{aligned}
\tag{5.19}
$$

for all $\lambda_i = 1, ..., d+2$.

Another possibility is to solve the coupled optimization problem (5.19) by directly using the primitive variables. This can be calculated with the inclusion of the transient matrix $\boldsymbol{A}_0$, which may lead to the Lagrangian functional of the form:

$$\begin{aligned} L\left(\widehat{\boldsymbol{Y}}_h^{n+1}, \lambda_i\right) =& \frac{1}{2}\left\| \sum_a N^a \boldsymbol{A}_0 \left(\widehat{\mathbf{Y}}^{a,n+1} - \mathbf{Y}^{a,n+1}\right)\right\|_{\boldsymbol{S}}^2 \\ & - \sum_{i=1}^{d+2} \lambda_i \int_\Omega \left(\sum_a N^a \left[\boldsymbol{A}_0\right]_{ij} \left(\widehat{\mathsf{Y}}_j^{a,n+1} - \mathsf{Y}_j^{a,n}\right)\right) d\Omega \\ & + \sum_{i=1}^{d+2}\sum_{s=0}^{k-1} \lambda_i \delta t \xi_{ks} \int_{\partial\Omega} \left[\sum_{j=1}^{d} n_j \boldsymbol{Q}_j\left(\boldsymbol{Y}_h^{n-s}\right)\right]_i d\Gamma \\ & - \sum_{i=1}^{d+2}\sum_{s=0}^{k-1} \lambda_i \delta t \xi_{ks} \int_\Omega F_i^{n-s}\, d\Omega, \end{aligned} \tag{5.20}$$

for all $\lambda_i = 1, ..., d+2$, and denoting by Y_i^a the corresponding nodal values of the $i-$th primitive variable in the standard Lagrangian interpolation.

We hope that the coupled functional formulation may lead to an increased correction in the conservative properties of the method, and thus, to the solution of the problems that we have encountered in the present work.

Part II

Chapter 6

`RefficientLib`: An efficient load-rebalanced adaptive mesh refinement algorithm for high performance computational physics meshes

In this chapter, a novel algorithm for adaptive mesh refinement in computational physics meshes in a distributed memory parallel setting is presented. The proposed method is developed for nodally based parallel domain partitions where the nodes of the mesh belong to a single processor, whereas the elements can belong to multiple processors.

Some of the main features of the algorithm are the capability to handle multiple types of elements in two and three dimensions (triangular, quadrilateral, tetrahedral and hexahedral), the small amount of required memory per processor and the parallel scalability up to thousands of processors. The presented algorithm is also capable of dealing with non-balanced hierarchical refinement, where multi refinement level jumps are possible between neighbor elements.

An algorithm for dealing with load-rebalancing is also presented, which allows moving the hierarchical data structure between processors so that load unbalancing is kept below an acceptable level at all times during the simulation. A particular feature of the proposed algorithm is that arbitrary renumbering algorithms can be used in the load rebalancing step, including both graph partitioning and space filling renumbering algorithms.

The presented algorithm is packed in the Fortran 2003 object-oriented library `RefficientLib`, whose interface calls which allow it to be used from any computational physics code are summarized.

6.1 Introduction

Discretized partial differential equations are used to solve many types of practical problems in engineering and physics. In some of these problems, the solution leads to a wide range of spatial scales which spread over the computational domain. In these cases, the numerical solution obtained with coarse meshes is often too inaccurate, but performing computations using fine meshes is impractical considering the required computational effort. Adaptive Mesh

Refinement (AMR) methods deal with this issue by producing efficient meshes that are capable of resolving a wide range of scales. These methods locally adjust the mesh to both improve the solution and minimize the computational effort.

Development of parallel AMR methods is justified in order to solve problems that contain a large number of unknowns, and which typically require the use of a huge amount of computational resources. Parallelizing the refinement methods allows exploiting the calculation capabilities provided by rapidly-evolving parallel computer clusters. However, parallelized refinement methods lead to a distributed mesh structure, which is complex because frequent data access is necessary, and memory consumption is high. In addition, the dynamical evolution of information during the adaptive mesh refinement constitutes another major challenge: it requires a growing number of collective communication operations, and therefore it is not easily scalable in massively parallel computers. Including the possibility to redistribute the workload between processors in order to maximize the utilization of computational resources increases even more the communications demand. Hence, efficient algorithms and data structures have become the backbone of parallel AMR methods, and distributed collection of structures that can be dynamically modified without requiring several global communications have been the preferred designs.

The first approach to parallel AMR methods was block-structured methods. These methods refine parallel meshes by using a single sequential mapping and therefore are not suitable for complex geometries and non-structured meshes. Tree-based methods were an alternative to the regularity imposed by the block-structured methods. Tree data structures, namely quadtrees and octrees, are hierarchical data structures constructed with axis-aligned lines and planes. These data structures are used for searching procedures because their hierarchical structure reduces the complexity of the search. The first application of tree data structures algorithms was in parallel domain decomposition and efficient partitioning of meshes (see for example Campbell et. at.[130]). Later, data structures, balancing algorithms, and adaptive refinement algorithms over distributed octree meshes were developed in [20, 21]. The *etree* library [131] collected algorithms that addressed operations over an octree-based mesh in a database-oriented framework. The code demonstrated good scalability and parallel efficiency. Furthermore, octree developments were implemented into *Octor* parallel meshing tool [132], which could generate statical unstructured meshes on the processors, but also performed dynamically refining during execution time. Scalability tests were addressed up to 62000 processors using hexahedra and giving an overall good performance. Some multigrid solvers exploited the balancing and meshing algorithms for octree-based meshes, and were implemented in *Dendro* software [133]. The code was scaled up to thousands of processors. Other applications and multiple implementations based on octree data structures were developed by [134, 135], and possessed good adaptivity and performance.

Instead of the quadrilateral and cubed shaped domains that were described by tree data structures, a wider variety of geometries were described by forest-of-octrees based meshes. This approach was first introduced into AMR methods with the *deal.II* software[24], but the code replicated the global mesh into all processors, hence it limited the scalability to a few processors. Fully distributed algorithms handling forest-of-octrees meshes were the following step. Burstedde et. al. [136] worked in a dynamically AMR based on distributed forest-of-octrees geometries. This was the first work that supported high-order discretizations and non-Cartesian geometries, and lead to the encapsulation of algorithms into *p4est* library [25].

Good strong and weak scaling results over 224000 cores were obtained for `p4est` working as a parallel adaptive refinement library on meshes composed by quadrilateral and hexahedral elements [137]. Later, Burstedde et. al. [22] focused on the balance structure, and proposed a subtree balancing algorithm. Weak scaling times improved and required less memory than previous balance algorithms in `p4est`.

In this chapter, we describe a general adaptive finite element framework for unstructured meshes that has demonstrated suitable performance for large-scale parallel computations. The algorithm currently focuses on $h-$refinement, the extension of the algorithm to $h-p$ refinement will be a matter of future work. Contrary to other parallel refinement algorithms, the method we present here is developed for nodally based parallel domain partitions, that is, the nodes of the mesh belong to a single processor, whereas elements can belong to multiple processors if they own nodes belonging to different subdomains. These remote nodes over the set of overlapping elements are called "ghost" points. This poses some challenges in the parallel communications since neighboring parallel domains need to be kept updated. Hence, local elements, points, edges, faces, and connectivities are stored in data structures that can be easily accessed and modified. Refinement operations and load balancing procedures are handled over these structures.

To our knowledge, `LibMesh` [23] was a similar approximation. However, because completely unstructured methods work at the cost of having to store explicitly the connectivities of the mesh, the parallel partitioning scheme of `LibMesh` stored the whole mesh information in each processor, and the associated overhead limited the scalability to a hundred processors. Janson et. al. [138] also implemented a general adaptive finite element framework for unstructured tetrahedral meshes without hanging nodes, which has been suitable for large-scale parallel computations. These last-mentioned authors presented strong scaling results linear up to a thousand processors for an incompressible flow solver. In contrast, the main contributions of the proposed refinement framework are:

1. A hierarchical adaptive refinement algorithm for nodally-based partitions in distributed memory machines is presented. The algorithm allows to successively refine and unrefine computational meshes in order to adapt to the requirements of the simulation.

2. Our distributed structure handles two and three-dimensional unstructured meshes composed of triangular, quadrilateral, tetrahedral and hexahedral elements. This approach is capable of describing complex geometries and doing non-uniform refinements.

3. We propose a distributed scheme in which each processor stores only the local information of the partitioned distributed mesh. This reduces the memory consumption and allows scaling up to thousands of processors.

4. Our parallel refinement procedure is based on a hierarchical data structure for the refined elements of the mesh, that we use to efficiently search neighboring elements at the interprocessor level. A data structure containing parent and children pointers is used, where new refinement levels are successively added to or subtracted from the computational mesh.

5. Resulting meshes are non conforming with *hanging nodes* on sides where two levels of refinement meet. Contrary to other adaptive refinement methods, the algorithm proposed

here does not enforce a *balancing* restriction in the refinement level of adjacent elements: the jump in the refinement level between neighbor elements can be arbitrarily large.

6. For the parallel refinement process, the proposed algorithm deals with element and node identification across processors by using a global element and global point identifier structure. This ensures that the global numbering structure and general nodal and elemental information can be transferred to all the neighboring processors in an efficient manner.

7. To balance the processors' load a dynamical parallel repartitioning framework that changes the ownership of the mesh nodes when load unbalances reach a certain threshold is used, and then it transfers the associated elements to the corresponding processors. Contrary to other algorithms for load rebalancing in hierarchical adaptive mesh refinement, the algorithm we propose is independent from the renumbering strategy of the load rebalancing process. In particular, graph partitioning schemes and space-filling methods for load rebalancing can both be used with the proposed algorithm.

The proposed algorithms are packed in an adaptive refinement library, which we call RefficientLib. The calls to the library have been made as simple as possible so that it can be easily coupled with existing finite element, volumes or differences codes.

Several numerical tests are carried out in order to assess the performance of the proposed methods. The first group of tests corresponds to simulation driven experiments which illustrate the capability of the method to generate computational meshes for different physical problems. A Poisson heat transfer problem is solved, both for bidimensional and three-dimensional elements. The incompressible flow past a cylinder is also tested in order to apply the AMR to the incompressible Navier-Stokes equations. In the second group of experiments, weak scalability tests for uniform refinement and load balancing cases in a high-performance computing environment are presented.

The chapter is organized as follows. In Section 6.2 the distributed refinement structure with the mesh partition strategy, the distributed data structures, and the initialization of the refinement procedure are described. In Section 6.3 the refinement step is described. Classification, local refinement, hanging nodes, and exportation to the external flat mesh algorithms are presented. Load rebalancing and global renumbering procedures are included in Section 6.4. The external calls and the user interface to the RefficientLib library from an external computational physics solver are presented in Section 6.5. Numerical experiments are presented in Section 6.6, together with the scalability tests. Finally, in Section 6.7 some conclusions are stated.

6.2 Distributed refinement structure

In this section, we describe the distributed structure of the adaptive mesh refinement method. The domain partition strategy is explained first, over which the parallelization is developed. Then, we introduce the main data structures and the initialization steps of the parallelized AMR method.

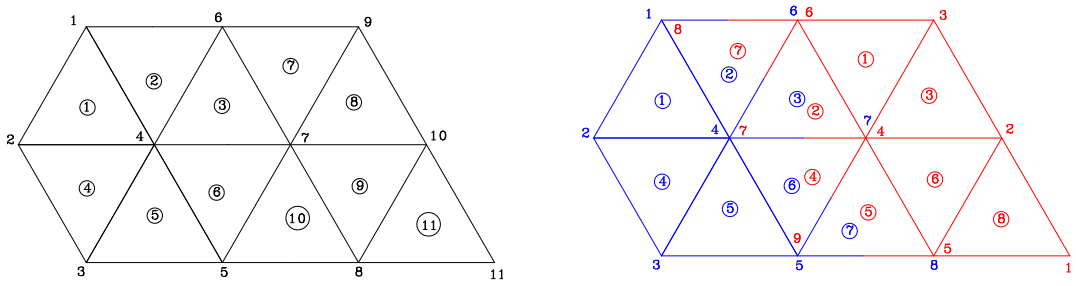


Figure 6.1: Initial mesh information. In the left, the global mesh. Points and elements (circled) are numbered globally. In the right, the mesh, partitioned into two subdomains. Colors denote the processor to which the information belongs. Points and elements (circled) are numbered locally for each domain. Note that the elements numbered as (2), (3), (6), and (10) in the global mesh, are shared by the two subdomains. This distributed structure will be used to calculate remote neighboring contributions for the shared elements.

6.2.1 Mesh partition

The algorithm described in this chapter is designed to work in distributed memory parallel machines. The idea is to have a library which takes care of all the steps necessary for the refinement, while the external driver (for instance a finite element solver) sees the resulting mesh as a non-hierarchical or flat grid. The domain partition strategy for the mesh is nodal based, which means that each node is assigned to an unique processor, but elements can belong to multiple processors if they own nodes from more than one subdomain. The concepts *point* and *node* both refer to nodes of the mesh, although *node* will generally be used when dealing with points in an element, and *point* will be used when treating them as independent entities. Points belonging to a given subdomain are denoted as *local points*. Before refinement, nodes are assigned to a single processor, but the first layer of nodes belonging to a neighbor processor is also stored in the current processor. These neighboring points are called *ghost* points. Elements can belong to multiple processors if they have nodes from multiple subdomains. We define the *processor responsible* for *an element* as the processor which owns the node of the element with the lowest global node number.

Fig. 6.1 shows an initial domain partitioned mesh as seen from different processors. The advantage of this strategy is that each processor stores only the local information of its subdomain. The processor stores the local numeration of the subdomain points, but the global numbering of these points must also be saved in order to locate and communicate points for other processors.

The parallel refinement method is constructed over the partitioned mesh. Since the mesh needs to be seen as a flat mesh by the external driver, a node and element renumbering strategy is needed in order to be able to move from the *external*, flat mesh, to the *internal* (refiner) hierarchical mesh and vice versa. Fig. 6.2 presents an example of a hierarchically refined mesh. Two levels of refinement are displayed as seen internally in the refiner from one of the

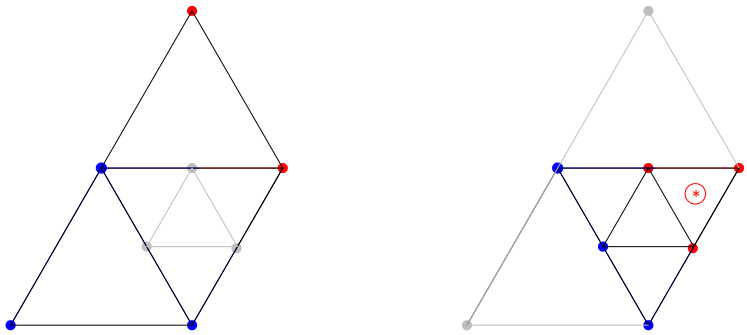


Figure 6.2: Internal hierarchical mesh as seen from one of the processors. In order to illustrate this refinement example, a shared element of the mesh of Fig 6.1 is refined. Left: initial level 0 mesh. Right: refined level 1 mesh.

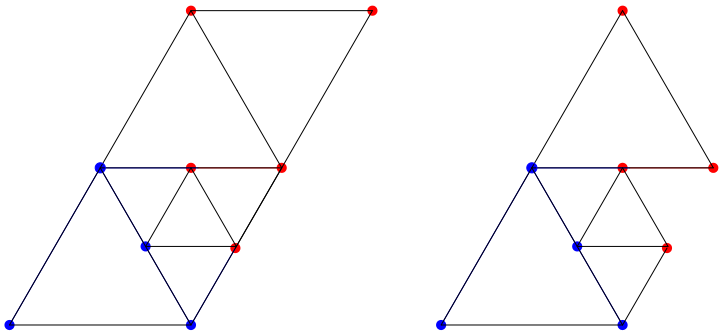


Figure 6.3: Refined mesh as seen from the external driver. Left: External flat mesh. Right: External flat mesh as seen from the processor denoted by the blue color.

processors. The same mesh is depicted in Fig. 6.3 as seen from the external driver. Note that the element marked with an $(*)$ in Fig. 6.2 does not appear in the flat mesh for the considered processor: in the external flat mesh, only the first layer of flat node neighbors is considered, while in the internal hierarchical mesh also the element hierarchical neighbors are considered. Element hierarchical neighbors are elements which share a parent with an element that belongs to a processor. The element marked with an $(*)$ in Fig. 6.2 is a hierarchical neighbor because although none of its nodes is assigned to the processor, the element shares its parent with the rest of level 1 elements in the processor. We call the elements which share a parent *sibling elements*.

6.2.2 Distributed data structures

The algorithms to be described in the next section are implemented on top of a collection of data structures which allow to efficiently access and modify the information that defines the

mesh. The implementation is generally done in an object-oriented manner, but for efficiency reasons, the actual storage in memory sometimes uses a *flattened* storage where parts of the objects are stored in an array list. We will denote this storage structure as *CSR*, in reference to the *Compact Sparse Row* storage used in many computational physics applications. All the data structures are encapsulated in a class, which we call the **Refiner**. Refinement procedures and communications are performed by this class. The data structures which are part of the Refiner class are briefly explained below:

- **gnpoin** is the total number of points of the internal mesh
- **gnelem** is the total number of elements of the internal mesh
- **npoin** is the number of points of the internal mesh in the local processor. This includes both the local (belonging to the current processor) points (**npoinLocal**) and the first layer and hierarchical neighbors of the local points (**npoinGhost**).
- **npoinLocal** is the number of points of the internal global mesh belonging to the current processor.
- **npoinGhost** is the number of points of the internal global mesh which are the first layer of hierarchical neighbors of the local nodes. Their data is required in the current processor.
- **nelem** is the number of elements of the internal mesh which are required in the current processor. This includes all elements having local nodes, but also elements which do not have a local node but are relevant in the hierarchical refinement process.
- **ElementList** is the list of elements in local numbering, of size **nelem**. For each local element we need to store it contains:
  - **ElementType**: this item identifies the element type (triangles, quadrilaterals, tetrahedra, hexahedra), and subtype or variation according to the subdivision process. The subtype is relevant for instance in the case of tetrahedral elements, where there are multiple possibilities for hierarchically subdividing an element. An array of 1-byte integers of dimension 2.
  - Data structure of type **GlobalElementIdentifier**: Similarly to the strategy followed in [137], the global element identifier allows to uniquely identify an element from the mesh (described in the following subsection 6.2.3).
  - **ParentIdentifier**: the local element numbering of the parent element.
  - **ChildrenIdentifierList**: the list of local element numbering of the children elements. (Stored in *CSR* format).
  - **NodeList**: contains the nodes which constitute the element in local numbering. An array of 4-byte integers. (Stored in *CSR* format).
  - **FaceList**: for each face in the element, it stores the neighbor (opposite) element and the corresponding face (or edge in two dimensions) of the neighbor. If this is a hanging face, it contains the neighbor of the parent element and the corresponding face. (Stored in *CSR* format).

- **PointList** is the list of nodes in local numbering, of size **npoin**. The first **npoinLocal** components of the list correspond to local points, the last **npoinGhost** components of the list correspond to ghost points. For each point, we store:
  - **GlobalPointNumbering**: this data structure stores the global (parallel) point numbering of the point. An **InverseGlobalPointNumberingList** is also created which allows getting the local point number of global points for a given processor. In order to implement the inverse global point numbering list, a hash table type structure is used (this is described in subsection 6.2.4).
  - **ProcessorNumber:** for ghost points, it stores the processor number to which the point belongs.
  - **Level:** the level (in the refinement structure) of the point. Initial points are classified as level 0.
  - **EdgeRefinementList**: for each point i, it stores a list of neighbor points j to which point i is connected only if a refinement node k between i and j exists, it also stores the local numbering of j and k. (Stored in *CSR* format).
  - **HangingNodeList**: for hanging nodes, it contains the list of recursive parent nodes and their linear combination coefficients. (Stored in *CSR* format).

Most of the lists involving multiple types of data (i.e. **ElementList** or **PointList)** are implemented as separated list arrays for convenience and memory performance, although they could also be stored as objects containing data structures. When the size of each component of the list is not constant for all elements, these separated list arrays are generally stored in a *Compact Sparse Row* storage format *(CSR)*.

6.2.3 The GlobalElementIdentifier data structure

The hierarchical refinement element information is encapsulated into a tree data structure composed of **GlobalElementIdentifier** objects. This data structure allows to uniquely identify an element in any processor, at any level of refinement. When used together with the **ParentIdentifier** and **ChildrenIdentifierList**, it allows to communicate element information between processors. This information can be used to identify the local numbering of an element when interprocessor information communication is required. The **GlobalElementIdentifier** structure is composed of the following information:

- **GlobalTopLevelElement.** This is the original element number of the top level element prior to any refinement or load rebalancing process. This is stored as a 4-byte integer.
- **Level**: The level of the element in the refinement structure. Initial elements are classified as level 0. This is stored as a 1 byte integer.
- **PositionInParentElement.** For each level, 3 bits are dedicated to storing the refinement branch (or child) of the element. This allows identifying a maximum of 8 children per level. A maximum of 21 refinement levels is allowed in the current implementation, totaling 63 bits. This is stored as an 8-byte integer.

The total amount of memory required to store the **GlobalElementIdentifier** for an element is 13 bytes, which round up to 16 bytes in memory.

6.2.4 The InverseGlobalPointNumberingList

In order to perform parallel communications, we need to be able to recover the local point number from a global point identifier at any time in the refinement process. This could be done in a straightforward manner by allocating an array of dimension **gnpoin** and then storing for each local point, in the global position of the array, the local number associated with the corresponding global point. However, **gnpoin** depends on the size of the global problem and can in general be very large. This would result in the allocation of this array becoming very time consuming and when using thousands of processors, affecting the performance and scalability of the parallel refinement algorithm.

Thus, an alternative implementation of the global to local mapping is required. In our implementation, we have opted by a hash filtering and storage in a table, followed by a binary search if collisions are found:

- Firstly, the hash function is defined as the modulus of the division by a primer number **primeNumber,** which is close to and larger than the local number of points **npoinLocal**. A hash table data structure of dimension **primeNumber** is allocated.
- Secondly, for each local point - global point pair, the hash function of the global point identifier is computed and the point is stored in the corresponding hash table data structure slot. If there are collisions, the points are stored in ascending order according to its global point identifier.
- Finally, in order to recover the local numbering of a global point, the hash function of the global point number is computed. If a single point is stored in the corresponding hash table slot, the local numbering is recovered directly. In case there are collisions, a binary search is performed on the sorted global point numbering array of the hash table slot in order to find the corresponding local point number.

This process allows reducing the average computational cost of finding the local point number associated with a global point to $\mathcal{O}(1)$ while keeping the storage requirements to $\mathcal{O}(\mathbf{npoinLocal})$. Although a possibly more efficient implementation could be found by differentiating on the behavior of the algorithm for local and ghost points, we have chosen the described implementation for its compromise between efficiency and reusability.

6.2.5 Initialization

Our parallel refinement method establishes the input initial mesh as a zero level mesh that cannot be coarsened. In the initialization step, the flat, top-level mesh is passed to the **Refiner**. From this mesh **ElementList** and **PointList** are built**.** All elements and points are assigned the zero level. The processor number for each point and a global element number for each element and point needs to be passed to the **Refiner**, from which the **GlobalElementIdentifier** for each element and the **GlobalPointNumbering** (and its inverse) for each point can be built. The

initial **FaceList** for each element is built by looping through neighbor elements and checking for faces with coincident nodes. Neighbor elements are identified in a two-step process as elements with which at least one point is shared. The remaining arrays, which refer to the refinement structure, are started as empty or null since no refinement step has been performed yet.

6.3 Refinement step

6.3.1 Amending the element refinement classification

The first stage of the refinement step consists in passing to the library an array of size **nelem** which contains the information about the element refinement. In our implementation, this is achieved by passing a 1 valued integer for elements which need to be refined, a -1 valued integer for elements which need to be unrefined and a 0 valued integer for elements which need to be neither refined nor unrefined. Elements can only be unrefined if all of their sibling elements are also unrefined. If an element is marked as to be unrefined but one of its siblings is not, then the element is reclassified as not to be unrefined.

An important point is that the refinement criteria must be the same on all processors, that is, if an element belongs to both processors i and j, then the decision on whether to refine the element must coincide in processor i and processor j. Even taking this into account, there are unrefinement cases in which the information available in a certain processor is not enough to decide if an element will effectively be unrefined. This is, for instance, the situation for a unrefinement step of the level one elements in Fig. 6.2 and Fig. 6.3. Suppose that the external driver has marked the four-level one elements which are exported to the external mesh (Fig. 6.3) as to be unrefined. However, the blue processor (Fig. 6.3, right) has no information on the classification of the hierarchical neighbor (marked with an $*$ in Fig. 6.2). As a consequence, and only in the case when all the local element siblings are marked as to be unrefined, a communication step with the neighbor processor is required in order to classify hierarchical neighbor elements.

This communication step consists in asking the processor responsible for the element to communicate its refinement classification. Elements in different processors are identified through their **GlobalElementIdentifier**. This step is not required when refining because sibling elements can be refined independently.

As explained previously, the proposed algorithm can deal with unbalanced meshes in the sense that the refinement level jump between neighbor elements can be arbitrarily large. However, this might not be convenient for some applications. For this, an optional flag which limits the level jump between neighbor elements has been added to the algorithm. If this flag is enabled, the algorithm adds the following to the previous reclassification of elements to be refined: for each node, it notes down the maximum and minimum level of the elements to which it belongs. Then, if the difference between the maximum and minimum level is 1 (the algorithm should not allow this difference to be larger than 1), it does not allow the maximum level elements to further refine, and it does not allow the minimum level elements to unrefine. This ensures that a balanced mesh is obtained in the case this flag is enabled.

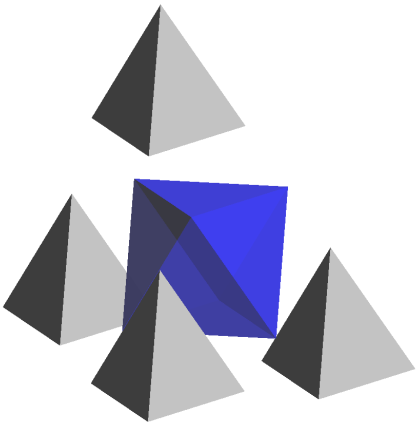


Figure 6.4: Subdivision of a tetrahedron into 4 tetrahedrons and 1 octahedron. The octahedron is then further subdivided into 4 tetrahedrons.

6.3.2 Local Refinement

Once all elements are properly classified following the refinement criteria, a local refinement step starts in each processor. The implemented subdivision process for triangles, quadrilaterals, tetrahedrons and hexahedrons refines a given element into 2^d subelements of the same type, where d is the number of dimensions, although the implementation is left open to refining into other element types in the future. A first loop through the elements allows computing the dimensions of the arrays after the refinement stage, which are then allocated. Elements which are unrefined are removed from the lists, new elements are added at the end of the **ElementList**. At the same time, the **ParentIdentifier** and the **ChildrenIdentifierList** for each element are filled. Also the **GlobalElementIdentifier** for each new element is computed from the **GlobalElementIdentifier** of its parent element (adding one level to the parent level, assigning a child number for the new level), as well as the element type (for h-refinement the element type of children elements is the parent element type). In the case of tetrahedrons, what we call an element subtype also needs to be stored: each refined tetrahedron is subdivided into eight subelements. However, there are three different ways of subdividing a tetrahedron into eight subelements, each one corresponding to the main plain of subdivision of the internal octaedron obtained by joining the midpoints of tetrahedron edges. The election of the subelement type is done so that the distortion of the resulting child elements is minimized. This is illustrated in Fig. 6.4.

For updating the **FaceList** of each element, the neighbors of each element are checked. If in the previous refinement step the face was connected to an element which has been unrefined in the current refinement step, then the element is connected to the parent element. On the contrary, if the element is connected to a higher level element which has now been refined, then the element becomes connected with the corresponding children. These face connections can be done in an efficient manner thanks to the **ParentIdentifier** and **ChildrenIdentifierList** structures, which allow moving through the different refinement levels. Some examples of face matching for elements in different levels are shown in Fig. 6.5. Faces which connect to faces in elements of a higher level are denoted as *hanging faces*.

At this point, the element refinement structure has been updated to the new refinement

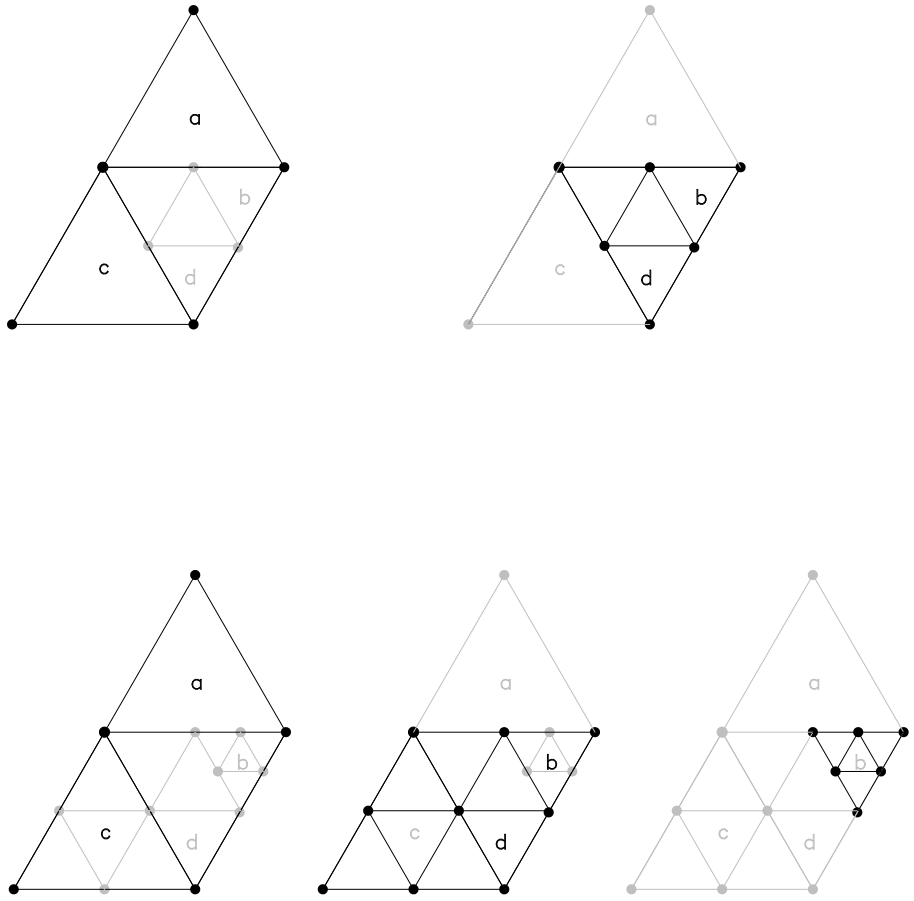


Figure 6.5: Face matching in the refinement process. In the mesh at the start of the refinement process (top), element b has a face connected to element a, which is one level higher. Similarly, element d has a face connected to one of the faces of the element c. After a refinement step, both elements c and b are refined (bottom). Some of the children of b have a face which is connected to a face in element a, while the faces in element d which were connected to element c are now connected to faces in the children of the element c.

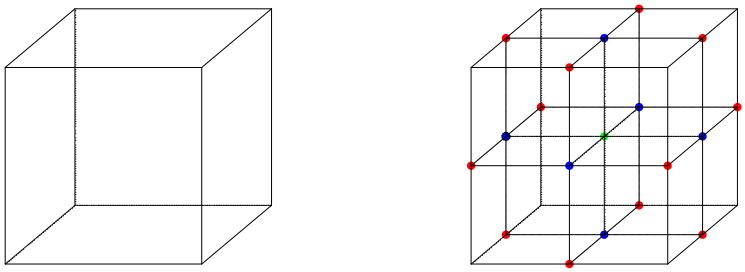


Figure 6.6: Edge (red), face (blue) and interior (green) new nodes in a refined element.

stage. However, the new nodes still need to be added to the new elements. There are three types of nodes in the new elements: nodes which are added to the edges of the parent element, nodes which are added to the faces of the parent element, and nodes which are added to the interior of the parent element. This is best illustrated in three-dimensional hexahedral elements, as shown in Fig. 6.6.

For nodes added in the interior of the parent element, we are sure that the node is new and a new point number can be assigned to the node. However, for nodes added in the faces or in the edges of a parent element, we need to check that the node does not already exist in another element.

In the case of nodes added in the face of elements, the node will only preexist if the face of the new child element is connected to an element of the same level. In this case, the new node in the new children element is assigned the point number of the node in the neighbor face.

For nodes added in the edges of elements, we make use of the **EdgeRefinementList** structure, in which for each edge connection between two points of the mesh we store the point numbering of the refined point added in-between them. This keeps track of the already existing refined points in the edges, allowing to add new points when an element is refined or, on the contrary, to match existing points with the refined point in the edge. The addition of new nodes to the **EdgeRefinementList** needs to be done in a two-step process (first loop for counting sizes and allocating, the second loop for filling the data structures) and making use of linked lists in order to obtain a proper performance of the algorithm.

6.3.3 Parallel numbering

The algorithmic steps in the previous section allow us to advance to the new refined mesh. However, no communications have been done between processors, so at this stage, new points have been assigned a local numbering, but the parallel numbering of points is still pending. Elements, on the other hand, are already identified by their **GlobalElementIdentifier**. In order to construct the new global point numbering, we start by classifying points as local or ghost in each processor. Points which already existed in the previous refinement stage keep the

same local/ghost status. New points are classified as local or ghost following different criteria depending on whether they are new interior points, new face points or new edge points.

The new interior points are classified as local for a given processor if, in the previous refinement step, the processor was the owner of the node in the parent element with the lowest global numbering (the processor was the *responsible processor for the element)*.

The new face points are classified as local for a given processor if, in the previous refinement step, the processor was the owner of the node in the parent face with the lowest global numbering (the processor was the *responsible processor for the face)*.

The new edge points are classified as local for a given processor if, in the previous refinement step, the processor was the owner of the node in the parent edge with the lowest global numbering (the processor was the *responsible processor for the edge)*.

Once all the points of each processor have been classified as local or ghost, a gather operation of the number of local points of each processor, followed by a scatter operation with the first global point number of each processor allows to set the parallel global numbering for all the local points in each processor (since the global numbering of the local points of each processor will be consecutive). However, the parallel numbering of the ghost points of each processor is still unknown to the processor. For points which were already ghosts in the previous refinement step, each processor simply asks the owner of the point to communicate its new global number. When asking for the point, this point is identified by its global number in the previous refinement step.

For new ghost points, their global numbering is obtained as follows:

- For new interior and face points, the **GlobalElementIdentifier** of the parent element is sent to the responsible processor together with the interior/face node number, which in turn returns the global point number for the interior point.

- For new edge points, the old (previous refinement step) global numbering of the edge parent points is sent to the processor responsible for the edge, which in turn seeks for the refined edge point through its **EdgeRefinementList** and returns the new point global number for the new edge point.

At this point, all the internal refinement structures are already defined and updated to the new refinement stage. The steps for exporting this information to the external mesh are detailed in the next sections.

6.3.4 Hanging nodes

Although in most refinement algorithms using tree data structures a conforming restriction over adjacent elements must be satisfied, this is not the case in our algorithm. This so-called *balance condition*, which enforces that there is only one *hanging node* on sides where two levels of refinement meet, does not need to be satisfied by the algorithm presented in this work, and an arbitrary jump in the refinement level of adjacent elements is possible. We believe that this is one of the main features of the present algorithm.

Hanging nodes are classified as nodes which belong to a face or edge that is connected to an element in a higher level (hanging face or edge) and which do not belong to the higher level element. There are several possibilities for treating hanging nodes in computational physics: one

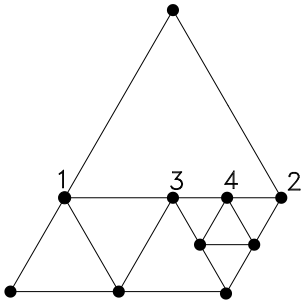


Figure 6.7: Hanging nodes. Node 4 is a hanging node, its hanging parents being nodes 2 and 3. Node 3 is in turn a hanging node, its hanging parents being nodes 1 and 2. The hanging node value for the unknown at node 4, u_4 is: $u_4 = \frac{1}{2}u_3 + \frac{1}{2}u_2 = \frac{1}{4}u_1 + \frac{3}{4}u_2$.

of the possible approximation for hanging nodes consists in fixing the value for the unknowns in the hanging node as the mean of the value of the unknowns of its hanging parents (this is the approach we have followed in the numerical examples). Other possibilities include the use of Discontinuous Galerkin methods [139], or hybrid Continuous-Discontinuous Galerkin methods [140].

In any case, it is necessary to know which are the hanging parents of a certain hanging node. Hanging parents are defined as nodes in the parent element in the case of interior refined nodes, nodes in the parent face in the case of face refined nodes, and nodes in the parent edge in the case of edge refined nodes. If the hanging parents are in turn hanging nodes, this establishes a recursive dependency of the unknown values in the hanging nodes with the values of the unknown in higher level nodes.

In our implementation, the list of hanging nodes and their relation with respect to their hanging parents is obtained by looping through the elements and checking the node matching of faces and edges which connect to higher level elements. The nodes which are not present in both sides of the face, or in all the elements which concur at the edge, are considered as hanging nodes.

Once the hanging node list is built, the recursive dependency structure with respect to their hanging parents is obtained by traversing the hanging node list from the top level to the low level hanging nodes and annotating the contribution of hanging parents to the averaged value of each hanging node. As the number of parents contributing to the averaged value of a hanging node can be arbitrarily large if no balance condition is applied, this recursive dependency structure is stored in *CSR* format. If a parent node of a hanging node is also a hanging node, then its contribution to the value of the hanging node is transferred recursively to the hanging parents. This is illustrated in Fig. 6.7.

Note also that in successive refinement steps, normal nodes can become hanging nodes depending on the refinement behavior of neighbor elements and vice-versa.

6.3.5 Exporting the external mesh

As explained in previous sections, the hierarchical mesh with which the algorithm works internally does not coincide with the flat mesh which is exported and used by the external driver. The main criteria for choosing elements and nodes which are passed to the external mesh is the following:

- Elements are only exported if they are last level elements which own at least a local node.
- Nodes are only exported if they belong to an exported element.

These two criteria are in general sufficient to decide which elements need to be exported. However, there are some specific cases (after load rebalancing has occurred and children elements are not necessarily in the same processor as their parent elements) where additional elements need to be exported. The first case is the case of hanging nodes whose hanging parent nodes are assigned to a different processor, which is illustrated in Fig. 6.8. In this case, the elements owning the recursive hanging parents need also to be exported in order to ensure that the assembly to the parent nodes can be performed and that if the high-level element is refined all the involved processors will be aware of the refinement step.

The second case is the case of elements which are *hanging opposites* of elements with nodes assigned to the current processor. Even if none of the nodes of the element belongs to the current processor, these elements need to be exported so that their refinement criteria can be known by the current processor and the new elements and face matching can be created. Again, an example of this particular case can be seen in Fig. 6.8.

In both cases, information on elements belonging to neighbor processors but not present in the current processor will have to be communicated between processors. This is the case of element *b* in Fig. 6.8.

6.4 Load Rebalancing

After several refinement steps, computational load in the processors may become unbalanced, causing the most loaded processor to damage the global efficiency. In order to avoid this issue, load rebalancing is required. Load rebalancing consists in changing the processor owning each group of nodes in such a way that the computational load is approximately equal in all processors. At the same time, it has to be ensured that the communications required in the load rebalancing process and in the external computations to be performed by the driver are minimized.

6.4.1 Rebalance Renumbering

The first step of the load rebalancing process consists in computing the new processor and new global number for each node in the mesh. Contrary to other adaptive refinement schemes in which the new node numbering is linked to the adaptive refinement algorithm, in the algorithm presented in this chapter any node renumbering strategy is possible. In fact, the new node numbering is computed externally by the driver and then passed to the refinement library. In

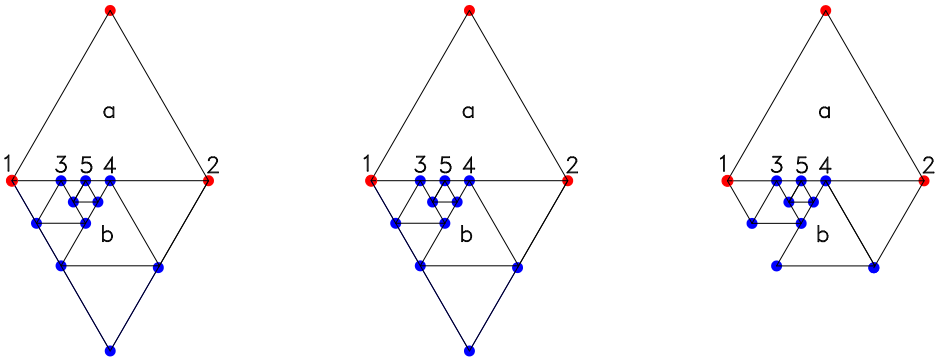


Figure 6.8: Left: Global exported mesh. Center: Local exported mesh for processor 0 (blue nodes). Right: Local exported mesh for processor 1 (red nodes). Element a would in principle not be exported in processor 0 since it has no local node for this processor, but it is exported because it is the hanging opposite of elements belonging to processor 0. Element b would in principle not be exported in processor 1, since it has no local node for this processor, but it is exported because it is the hanging opposite of element a, belonging to processor 1.

the numerical examples presented in Section 6.6, node renumbering is computed externally by using the ParMetis [141] and Zoltan [142] specialized software, which are based on nested bisection, graph partitioning, and space-filling methods. Both of these packages are accessed through an interface provided by the PETSc library [114], which we also use as a linear system solver for the numerical examples in Section 6.6. All of these remapping methods produce new partitions from an already partitioned mesh.

With all the nodes in the mesh already assigned a processor number to which they belong, it remains to decide which elements and nodes need to be sent to each processor. The algorithmic rules of this step are similar to the rules for exporting elements and nodes in Section 6.3.5, although taking into account the hierarchical nature of the internal mesh:

- An element needs to be sent to a processor if it owns a node which belongs to the processor.
- An element needs to be sent to a processor if one of its children or sibling elements is sent to the processor.
- A node needs to be sent to a processor if it belongs to at least one element which is sent to the processor.

Again, an additional rule applies in the case of elements with hanging nodes: any element which is the high-level hanging (edge or face) opposite of an element which needs to be sent to a given processor, will also be sent to the processor. The situation is similar to the one depicted in Fig. 6.8.

6.4.2 Rebuilding the refinement structure

The final step before the refiner is ready to continue with the following step of the adaptive mesh refinement strategy consists in rebuilding all the required data structures. The **FaceList** structure and the **ParentIdentifier** and **ChildrenIdentifierList** arrays can be rebuilt in a two-step process from the information in the **NodeList** and by traversing the element levels using the **GlobalElementIdentifier** of each element. Once this is done, rebuilding the **HangingNodeList** and the **EdgeRefinementList** is also straightforward**,** although some care needs to be taken so that the resulting implementation is efficient.

6.5 External calls to the `RefficientLib` library

In this section, we present the interface and calls to the `RefficientLib` object-oriented library. The implementation allows the refinement algorithm to be used from distributed memory computational physics codes. The library has been developed following the object-oriented Fortran 2003 standard. In the following, we illustrate a typical call to the adaptive refinement library.

The first step consists in the initialization of Refiner object. This is done by passing it an **IniData** object from which the initial mesh information can be retrieved (elements, nodes, and connectivities). The **IniData** object is defined as abstract in the library and needs to be implemented and extended by the user. This object is an object from which the initial mesh information can be extracted:

```
call Refiner%Initialize(IniData)
```

The required calls for this object are the following:

```
!Number of points in the processor
call IniData%GetNpoin(npoin)
!Number of local points in the processor
call IniData%GetNpoinLocal(npoinLocal)
!Number of elements in the processor
call IniData%GetNelem(nelem)
!Global Number of Points
call IniData%GetGlobalNpoin(gnpoin)

!Element connectivity list
call IniData%GetConnectivity(ielem,nnode,connectivity)

!Get the local to global mapping for a set of nnode nodes
!Output is GlobalNumbering
call IniData%GetLocal2Global(nnode,LocalNumbering,GlobalNumbering)

!Get the processor to which a set of nnode nodes are assigned
!Output is ProcessorList
call IniData%GetProcessorNumber(nnode,LocalNumbering,ProcessorList)
```

Also, the coordinates of the nodal points of the mesh are passed to the refiner. These are only used in the tetrahedral elements case in order to choose the subelement type which causes the lesser distortion, as explained previously. **Coord** contains the array of coordinates, necessary for the subelement type choice in tetrahedral elements:

```
call Refiner%SetCoordArray(coord)
```

An optional call for setting the balancing flag (and forcing the resulting mesh to be balanced) can be done in the following manner:

```
call Refiner%Set_2_1_Balancing(.false.)
```

Once the refiner is initialized, the array **RefinerMarkel** containing the refinement criteria is passed to the refiner, which performs all the required refinement steps:

```
if (refining) then
 call Refiner%Refine(RefinerMarkel)
endif
```

If instead of a refinement step, we are carrying out a load rebalancing step, an external call to the renumbering library needs to be done, which needs to retrieve the **PointGlobNumber** and the **PointProcNumber** arrays. Once these are available, the refiner communicates and rebuilds the refinement structure: form:

```
if (rebalancing) then
 call ExternalLibraryRenumbering(PointGlobNumber,
                                 PointProcNumber)
 call Refiner%SetRebalancingNumbering(PointGlobNumber,
                                      PointProcNumber)
 call Refiner%LoadRebalance
endif
```

Once the refinement or load rebalancing step is done by the refiner, information can be retrieved using several calls, and from it the new external flat mesh can be built:

```
!New Dimensions
!number of local points, local + ghost points, global points
call Refiner%GetPointDimensions(newnpoinLocal,newnpoin,newgnpoin)
!number of elements, size of the connectivity array
call Refiner%GetElementDimensions(newnelem,newLnodsSize)

!After allocation of external arrays, the information can
!be retrieved from the Refiner
!Element Connectivity list in CSR format
call Refiner%GetLnods(pnods,lnods)
!Local to Global and processor list for the new local nodes
call Refiner%GetLocalOrdering(LocalToGlobal,ProcessorList)
!A list of hanging nodes (pHangingList, lHangingList)
!with the averaging coefficients (rHangingList)
!in CSR format can be retrieved from the Refiner
call Refiner%GetHangingListDimensions(HangingListSize)
```

```
call Refiner%GetHangingList(pHangingList,lHangingList,
                            rHangingList)
!A list of hanging faces (pHangingList, lHangingList)
!in CSR format can be retrieved from the Refiner
call Refiner%GetHangingFacesList(pHangingFacesList,
                                 lHangingFacesList)
```

The hanging nodes list contains, for each node, the list of their hanging parents (*lHangingList)* and the corresponding averaging coefficients (*rHangingList)*, stored in *CSR* format.

For all nodal arrays defined in the old mesh, a call to the refiner allows to transform them (by interpolation and restriction) to arrays in the new mesh:

```
call Refiner%UpdateVariable(ndime,coord,newcoord)
```

This summarizes the interaction with the adaptive refinement library from a user point of view. Some additional optional calls that allow passing values in the boundaries of the old mesh to the boundaries of the new mesh exist. These can be convenient for instance for enforcing Neumann boundary conditions in finite element analysis, but we have not included them here for legibility and conciseness.

6.6 Numerical examples

In this section, we illustrate the behavior of the proposed algorithm through several numerical examples. In these examples, the algorithm is tested for various types of linear, bilinear and trilinear elements, both in two and three dimensions.

6.6.1 Bidimensional elements

The first numerical example consists of a square bi-dimensional domain $[0,1] \times [0,1]$ meshed using triangular linear elements. The refinement process is arbitrary in order to show the capability of the algorithm to deal with multilevel jumps across hanging faces, and in this case, it concentrates most of the elements in the right-top corner.

The number of processors in this simulation is 25, and the load rebalancing criteria is the following:

$$\frac{\max\left(\mathbf{npoinLocal}\right) \cdot \mathbf{nproc}}{\mathbf{gnpoin}} \geq \mathrm{tol}_{\mathrm{rebalance}}$$

Over this mesh, a Poisson heat transfer problem is solved using finite elements. The heat transfer problem consists of finding consist of finding $u : \Omega \longrightarrow \mathbb{R}^d$ such that:

$$\begin{aligned} -k\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{in } \Gamma_D, \\ k\boldsymbol{n} \cdot \nabla u &= h \quad \text{in } \Gamma_N. \end{aligned}$$

where $k > 0$, f is a given forcing function, h is the normal heat flux, Ω denotes the computational domain and $\partial\Omega = \Gamma = \Gamma_D \cap \Gamma_N$ its boundary, with $\Gamma_D \cap \Gamma_N = \emptyset$. In this example

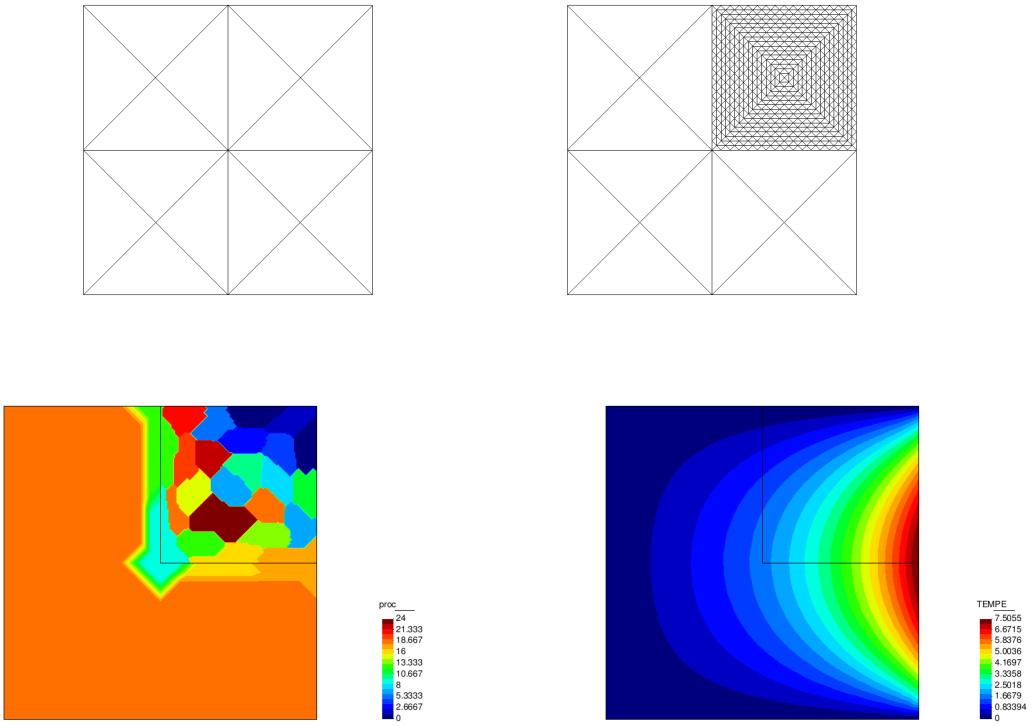


Figure 6.9: Poisson problem example on a triangular mesh. Top-left: original mesh. Top-right: mesh after some refinement steps. Bottom-left: node distribution across processors. Bottom-right: temperature field at the end of the simulation. Note that the jump of refinement level across hanging faces can be arbitrarily large.

the forcing term is uniform with value $f = 1$, Γ_D is composed by the upper, lower and left boundaries, while Γ_N is composed of the right boundary, with $h = -20$.

Fig. 6.9 shows the behavior of the method for this case. After several refinement/unrefinement steps, the mesh is much more heavily refined in the right-top corner than in the rest of the computational domain. Note that in this process, in some of the steps normal nodes become hanging nodes and vice-versa, The load rebalancing acts by rearranging the ownership of the nodes in such a way that the computational load in all the processors is approximately the same. This results in most of the processors dealing with nodes in the top-right corner. The algorithm is capable of dealing with the multilevel jump in hanging faces and providing an accurate result for the temperature field.

6.6.2 Multiple types of elements in a single bidimensional simulation

In this numerical example, we solve again the example presented in Section 6.6.1, but this time we use two types of elements: in the left half of the domain we use triangular finite elements, while in the right half quadrilateral finite elements are used. This example illustrates

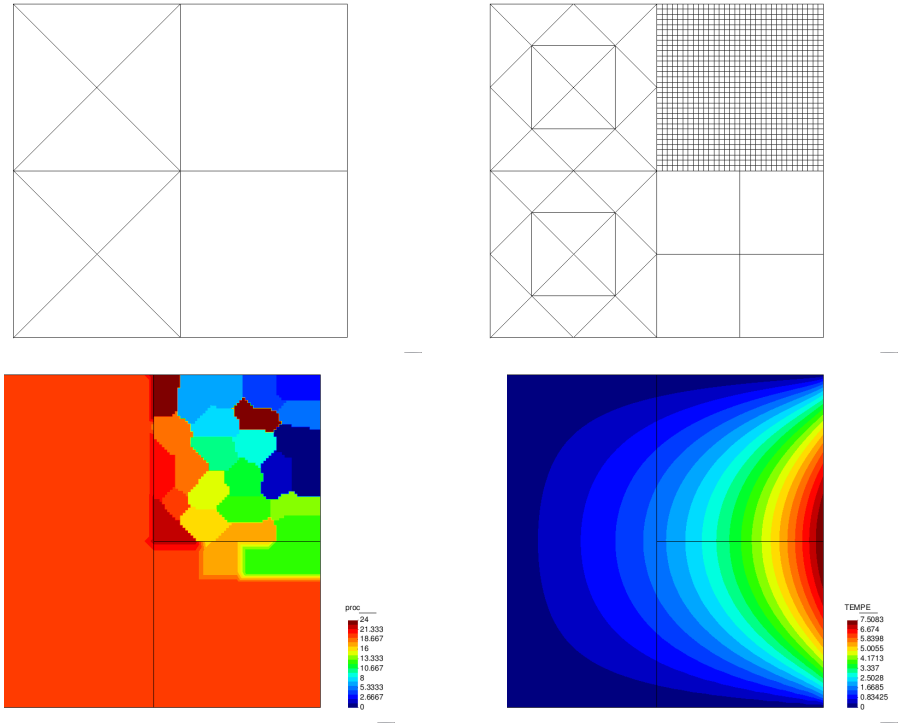


Figure 6.10: Adaptive simulation in a finite element mesh with several types of elements. Top-left: original mesh. Top-right: mesh after some refinement steps. Bottom-left: node distribution across processors. Bottom-right: temperature field at the end of the simulation.

the capability of the algorithm to deal simultaneously with several types of finite elements. Fig. 6.10 shows the numerical results. Note that some of the hanging faces of the mesh belong to the interface between triangular and quadrilateral elements.

6.6.3 Tetrahedral and Hexahedral elements

In this numerical example, a heat transfer problem is solved in a unit cube domain. The boundary conditions are adiabatic in all walls except in the lower one, where the temperature is fixed to zero. The source term is $f = 1$. The selection of the elements to refine is again arbitrary, and several refinement/unrefinement steps are performed before arriving at the final configuration. The number of processors in this numerical example is 6.

Fig. 6.11 shows the behavior of the method for this case. After several refinement/unrefinement steps, the mesh is much more heavily refined in the right-top quarter than in the rest of the computational domain and the load rebalancing algorithm acts by rearranging the nodes in each processor so that the computational load is similar in all processors. The algorithm is capable of dealing with the multilevel jump in hanging faces and providing an accurate, smooth result for the temperature field.

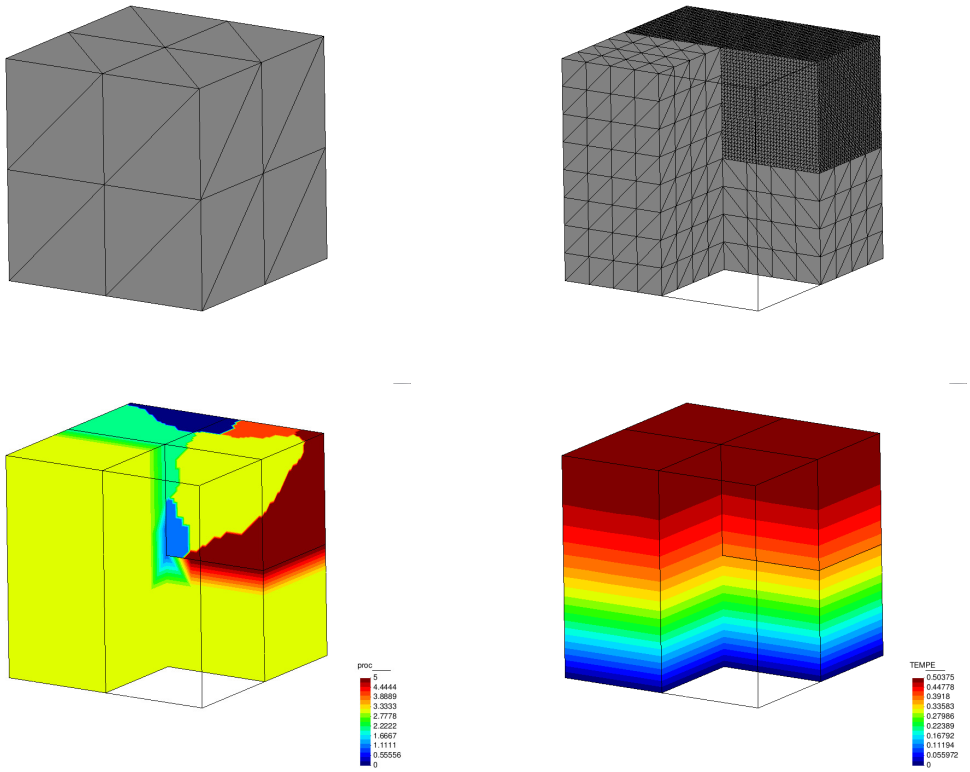


Figure 6.11: Mesh refinement for tetrahedral elements. Top-left: original mesh. Top-right: mesh after some refinement steps. Bottom-left: node distribution across processors. Bottom-right: temperature field at the end of the simulation.

Fig. 6.12 shows the same example using hexahedral elements and 25 processors, the load rebalancing criteria being the same as in the previous cases. In the three-dimensional case no simulation is done using multiple types of elements since the faces of the elements and the finite element shape functions for tetrahedral and hexahedral elements do not match at the interface.

6.6.4 An application to the incompressible Navier-Stokes equations

In this numerical example we solve the incompressible Navier-Stokes equations, which consist of finding $\boldsymbol{u} : \Omega \times (0, t_f) \longrightarrow \mathbb{R}^d$ and $p : \Omega \times (0, t_f) \longrightarrow \mathbb{R}$ such that:

$$\begin{aligned} \partial_t \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{in } \Omega, \\ \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma. \end{aligned}$$

for $t > 0$, where $\partial_t \boldsymbol{u}$ is the local time derivative of the velocity field. $\Omega \subset \mathbb{R}^d$ is a bounded domain, with $d = 2, 3$, ν is the viscosity, and $\boldsymbol{f}$ the given source term. Appropriate initial conditions have to be appended to this problem. The numerical solution of these equations through finite elements is done using a stabilized formulation [57] which allows to deal with the convective term and use equal interpolation spaces for the velocity and the pressure.

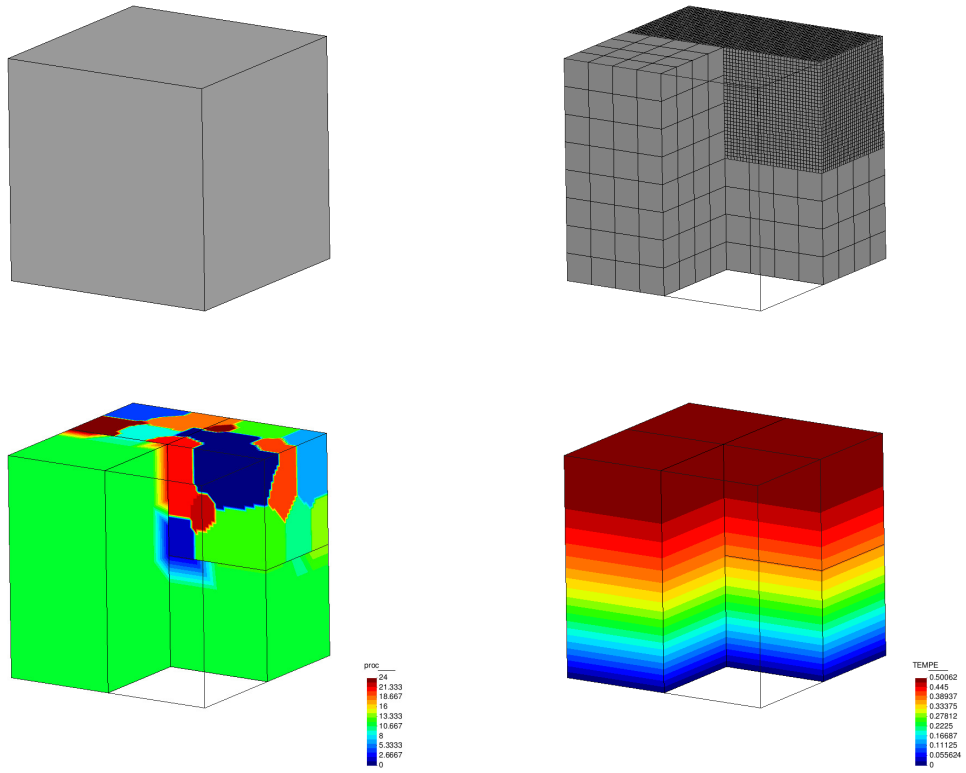


Figure 6.12: Mesh refinement for hexahedral elements. Top-left: original mesh. Top-right: mesh after some refinement steps. Bottom-left: node distribution across processors. Bottom-right: temperature field at the end of the simulation.

This numerical example deals with the flow around a cylinder at $\mathrm{Re} = 100$. The computational domain consists of a 16×8 rectangle with a unit-diameter cylinder centered at $(4, 4)$. The horizontal inflow velocity is set to 1 at $x = 0$. Slip boundary conditions are set at $y = 0$ and $y = 8$, and velocity is set to $\mathbf{0}$ at the cylinder surface. The viscosity has been set to $\nu = 0.01$, and the Reynolds number is $\mathrm{Re} = 100$ based on the diameter of the cylinder and the inflow velocity. A third-order backward differences scheme has been used for the time integration with a time step size $\delta t = 1 \cdot 10^{-3}$. As an error estimator, the Zienkiewicz-Zhu error estimator [143] for the velocity gradient has been used, which results in a refinement strategy near the boundary layer and in the regions surrounding the vortexes behind the cylinder:

$$e_K = \int_K \Pi_h^{\perp} \left(\nabla \boldsymbol{u}_h\right) = \int_K \left(\nabla \boldsymbol{u}_h - \Pi_h \left(\nabla \boldsymbol{u}_h\right)\right),$$

where K denotes each element of the mesh, and Π_h denotes the projection onto the finite element space and $\Pi_h^{\perp}$ denotes the projection onto the space orthogonal to the finite element space. Fig. 6.13 shows the results and mesh evolution obtained for this example. Note that in the refinement process, some of the normal nodes become hanging nodes and vice-versa at each step.

6.6.5 An application to a non-smooth solution

Here we present a non-smooth solution example for the steady Stokes problem, which consist of finding $\boldsymbol{u} : \Omega \times (0, t_f) \longrightarrow \mathbb{R}^d$ and $p : \Omega \times (0, t_f) \longrightarrow \mathbb{R}$ such that:

$$\begin{aligned} -\nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{in } \Omega, \\ \boldsymbol{u} &= \bar{\boldsymbol{u}} \quad \text{on } \Gamma, \end{aligned}$$

where $\Omega \subset \mathbb{R}^2$ is a two-dimensional bounded domain, ν is the viscosity, and $\boldsymbol{f}$ a given source term. A divergence free non-smooth manufactured solution is considered:

$$\begin{aligned} \boldsymbol{u}\left(r, \phi\right) &= r^{\alpha} \begin{bmatrix} \cos\left(\phi\right) \psi'\left(\phi\right) + \left(1+\alpha\right) \sin\left(\phi\right) \psi\left(\phi\right) \\ \sin\left(\phi\right) \psi'\left(\phi\right) - \left(1+\alpha\right) \cos\left(\phi\right) \psi\left(\phi\right) \end{bmatrix}, \\ p\left(r, \phi\right) &= -r^{(\alpha-1)} \frac{\left(1+\alpha\right)^2 \psi'\left(\phi\right) + \psi'''\left(\phi\right)}{1-\alpha}, \end{aligned}$$

with

$$\begin{aligned} \psi\left(r, \phi\right) = &\frac{\sin\left(\left(1+\alpha\right)\phi\right) \cos\left(\alpha\omega\right)}{1+\alpha} - \cos\left(\left(1+\alpha\right)\phi\right) \\ &+ \frac{\sin\left(\left(\alpha-1\right)\phi\right) \cos\left(\alpha\omega\right)}{1-\alpha} + \cos\left(\left(\alpha-1\right)\phi\right). \end{aligned}$$

Here ω and α are taken as $\omega = 3\pi/2$ and $\alpha \approx 0.5444837$, which is an approximation to the root of the nonlinear equation

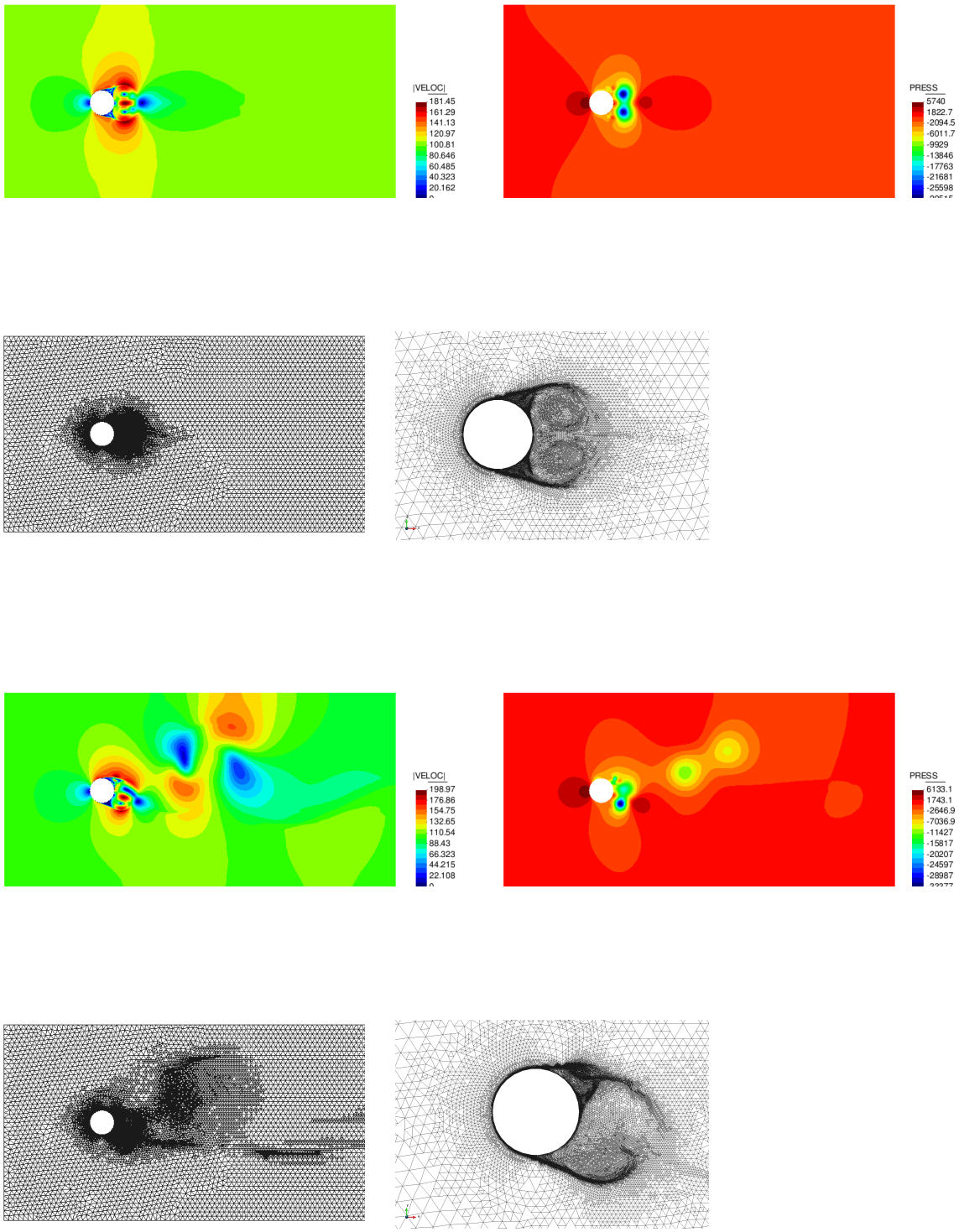


Figure 6.13: Adaptive refinement finite element solution of the flow past a cylinder. Contour results for the velocity and pressure fields and the refined mesh details are presented for two separated configurations of the wake oscillation.

$$\frac{\sin^2(\alpha\omega) - \alpha^2 \sin^2(\omega)}{\alpha^2} = 0.$$

We depart from a six linear element triangular mesh and refine it by using the Zienkiewicz-Zhu error estimator and refinement criteria explained in the previous example. Fig. 6.14 shows the obtained velocity and pressure solutions. The original and the refined mesh after several refinement steps are also displayed. Several refinement levels are developed for the refined mesh. The mesh refinement deals with the singularity appearing in the corner, and provides an accurate solution capable of better representing the pressure field close to the singularity.

6.6.6 Scalability tests

In this section, we test the scalability of the proposed refinement method when a large number of processors is used. We consider two cases: in the first case we solve an adaptive refinement case where the problem is balanced, and no load rebalancing is required. In the second case, we test the algorithm for a case where load rebalancing is required at almost every step of the refinement process. For each step, we refine the elements contained inside the domain, following either a uniform refinement criteria or a refinement criteria which aggressively forces load rebalancing. The weak scalability tests are run up to 1849 processors for bidimensional elements, and 1728 processors for three-dimensional elements. Both scalability cases are limited to 2000 million total elements for the largest mesh created at the last refinement level since our current implementation uses 4 byte integers for the nodal and elemental counters. The tests presented in this section were run at the Marenostrum supercomputer at the Barcelona Supercomputing Center, and at the Beskow supercomputer at KTH Sweden. The Marenostrum supercomputer is equipped with Intel SandyBridge-EP E5–2670 cores at 2.6 GHz (3,056 compute nodes), and 103.5 TB of main memory. The Beskow supercomputer is a Cray XC40 system based on Intel Xeon E5-2698v3 cores at 2.3 GHz (1676 compute nodes), and 104.7 TB of main memory.

The first weak scalability test corresponds to a uniform refinement problem. We depart from a structured uniform mesh with an initial number of elements and points per processor, and successively refine it. For each time step, we refine the elements in the entire spatial domain. The CPU runtime invested in the refinement procedures is presented in Fig 6.15. The objective of this weak scalability test is to measure the communications between processors. Because no communications are needed for load balancing, the measured time is only due to the refinement procedure. The results show an increase in runtime between 1 and 100 processors. After 100 processors a flat tendency is observed. The scaling results are good both for bidimensional and three-dimensional cases, assuring the correct behavior of the adaptive refinement algorithm. The runtime fraction of the refinement procedure with respect to the linear system solution is presented in Fig 6.16 for a single time step. Results show that the runtime is dominated by the linear system solution and that the runtime fraction of the refinement procedure decreases with the number of processors, ensuring a good behavior in large-scale computing.

The second weak scalability test intends to evaluate the overall performance when load rebalancing is necessary. For this case we depart from a structured uniform mesh and refine it successively using the following refinement criteria: for each step we refine only the elements contained inside the domain $\Omega_r = \left[1 - \frac{1}{2^i}, 1\right] \times \left[1 - \frac{1}{2^i}, 1\right] \times [0, 1]$, where i denotes

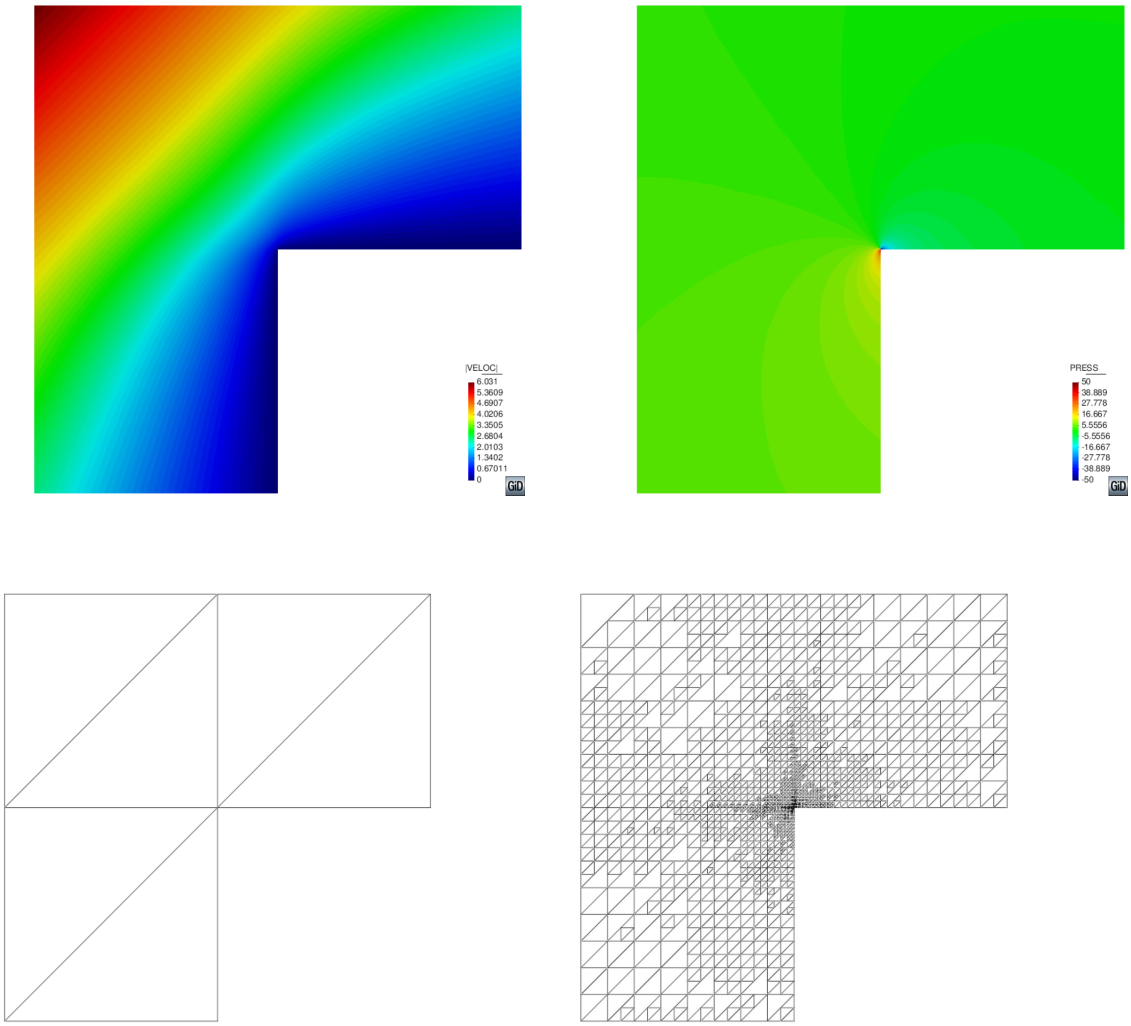


Figure 6.14: Adaptive refinement finite element solution of the Stokes problem, non-smooth solution. Contour results for the velocity (top left) and pressure (top right) fields, and the original (bottom left) and refined (bottom right) meshes.

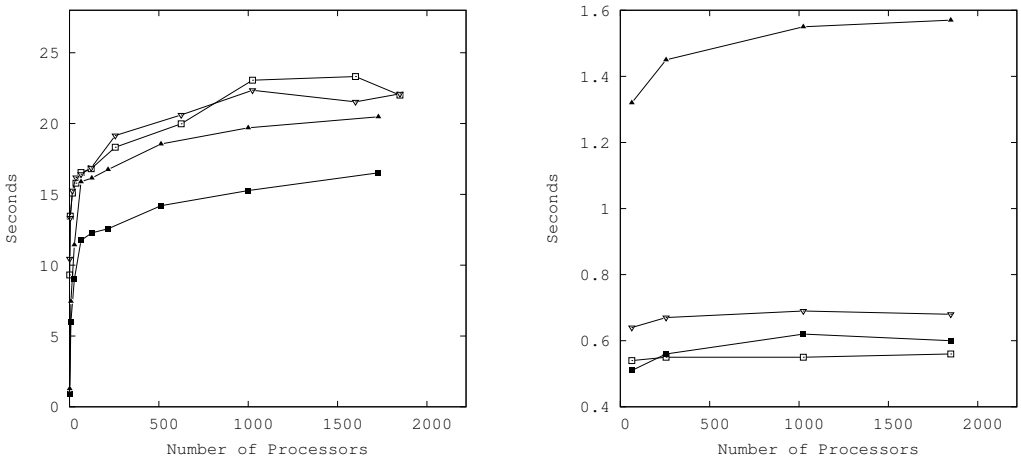


Figure 6.15: Uniform refinement weak scalability results from up to 1849 processors. The notation for the element type is used as follows: Triangles(△), squares (□), tetrahedra(▲), hexahedra(■). Left: in this case, we depart from an initial number of points per processor (100 for bi-dimensional elements and 8 for three-dimensional elements), and perform 6 levels of uniform refinement. Thereafter, we include several uniform unrefinement/refinement steps after the last refinement level. This type of run is performed in order to be able to measure the overall refinement performance of several time steps and is calculated using the Marenostrum supercomputer. Right: in this case, we depart from an initial number of points per processor (90000 for bi-dimensional elements and 270000 for three-dimensional elements), and perform a single level of uniform refinement. We calculate this case using the Beskow supercomputer.

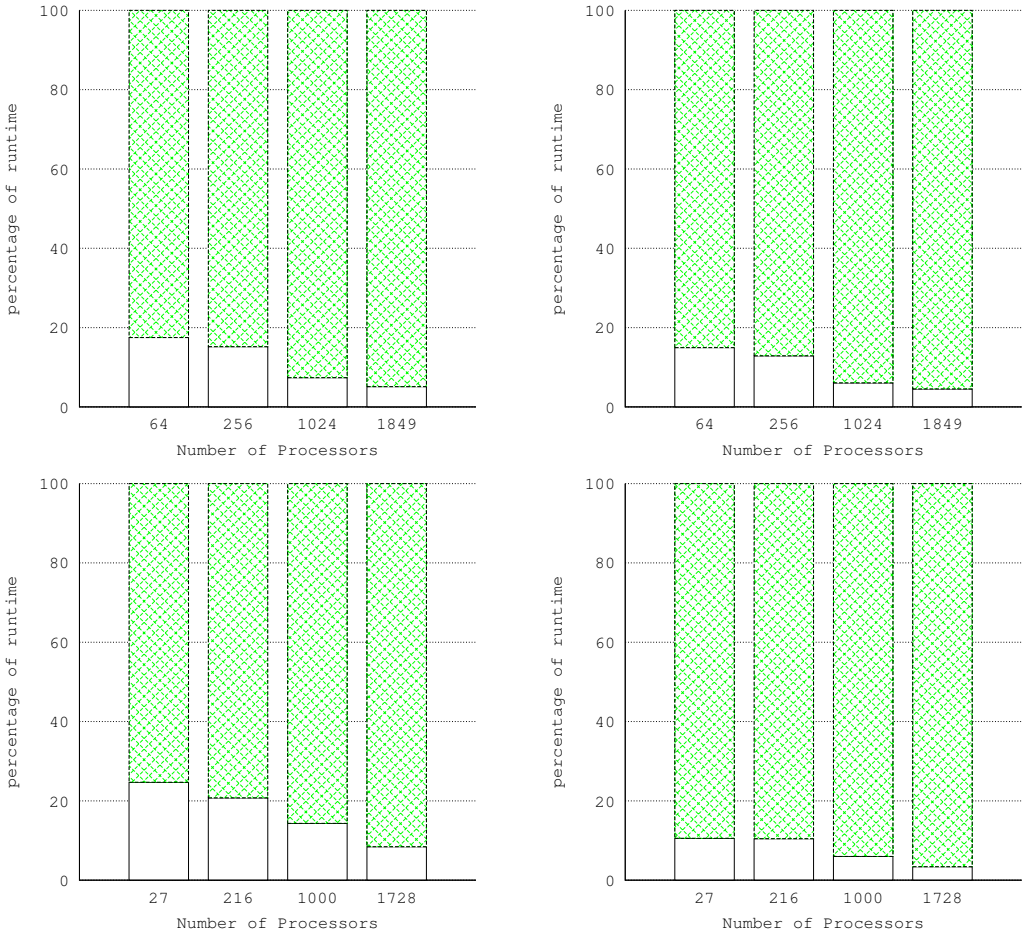


Figure 6.16: Runtime fraction results for the uniform refinement weak scaling tests up to 1849 processors on Beskow supercomputer. Triangles(top left), squares (top right), tetrahedra(bottom left), hexahedra(bottom right). The description of this case is given in Fig. 6.15. We perform a single level of uniform refinement and solve the stationary heat transfer problem with the resulting refined mesh. The refinement procedure runtime fraction is plotted with a solid fill, and the linear system runtime fraction is plotted using the pattern fill. We addressed the fraction of the time required by a single refinement procedure in comparison with the time spent by the linear solver in order to solve the refined mesh linear system. We compare with the best runtime among several linear solvers, which was given by the biconjugate gradient method implemented in the PETSc parallel solver library [114], together with the ML preconditioning package implemented in the TRILINOS library.

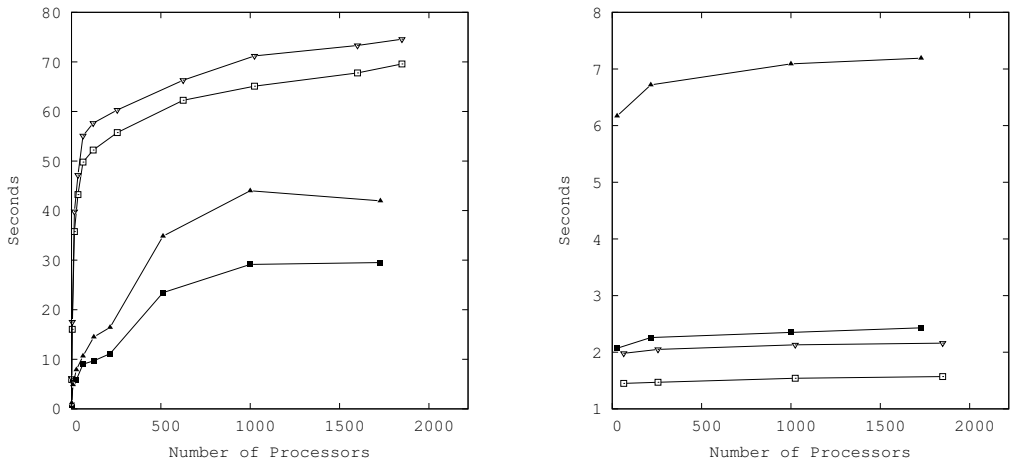


Figure 6.17: Load balancing refinement weak scalability results from up to 1849 processors. The notation for the element type is used as follows: Triangles(△), squares (□), tetrahedra(▲), hexahedra(■). Left: in this case, we depart from an initial number of points per processor (4900 for bi-dimensional elements and 8 for three-dimensional elements), and perform 15 levels of load rebalancing refinement. This run is calculated using the Marenostrum supercomputer. Right: in this case, we depart from an initial number of points per processor (90000 for bi-dimensional elements and 270000 for three-dimensional elements), and perform a single level of forced load rebalancing refinement. We calculate this case using the Beskow supercomputer.

the step number in this case. This criterion defines a spatial distribution which partially refines elements of the domain. Figure 6.17 shows the runtime required for the refinement and load-balancing procedures. The interprocessor communications, in this case, are due to the refinement step, but also to the load rebalancing procedure. An asymptotic tendency is reached for both bidimensional and three-dimensional elements as the number of processors increases. The three-dimensional elements weak scaling results are good up to the range of thousand processors. The runtime fraction spent by the refinement and load balancing procedures with respect to the linear system solution is presented in Fig 6.18. In all the cases the refinement and load-balancing computational cost was small to moderate with respect to the linear system solve. The linear system solve was tested for one degree of freedom heat-transfer problem. In the case of an incompressible Navier-Stokes problem, non-linearities and and a larger number of degrees of freedom per node would increase considerably the cost of solving the linear system, but the refinement procedure cost would remain the same, leading to an even smaller runtime fraction of the refinement step. Therefore, we conclude that the implementation of the proposed algorithm is efficient with respect to the cost of solving linear systems on adaptively refined meshes, and the algorithm is suitable for large-scale problems in high performance computing environments.

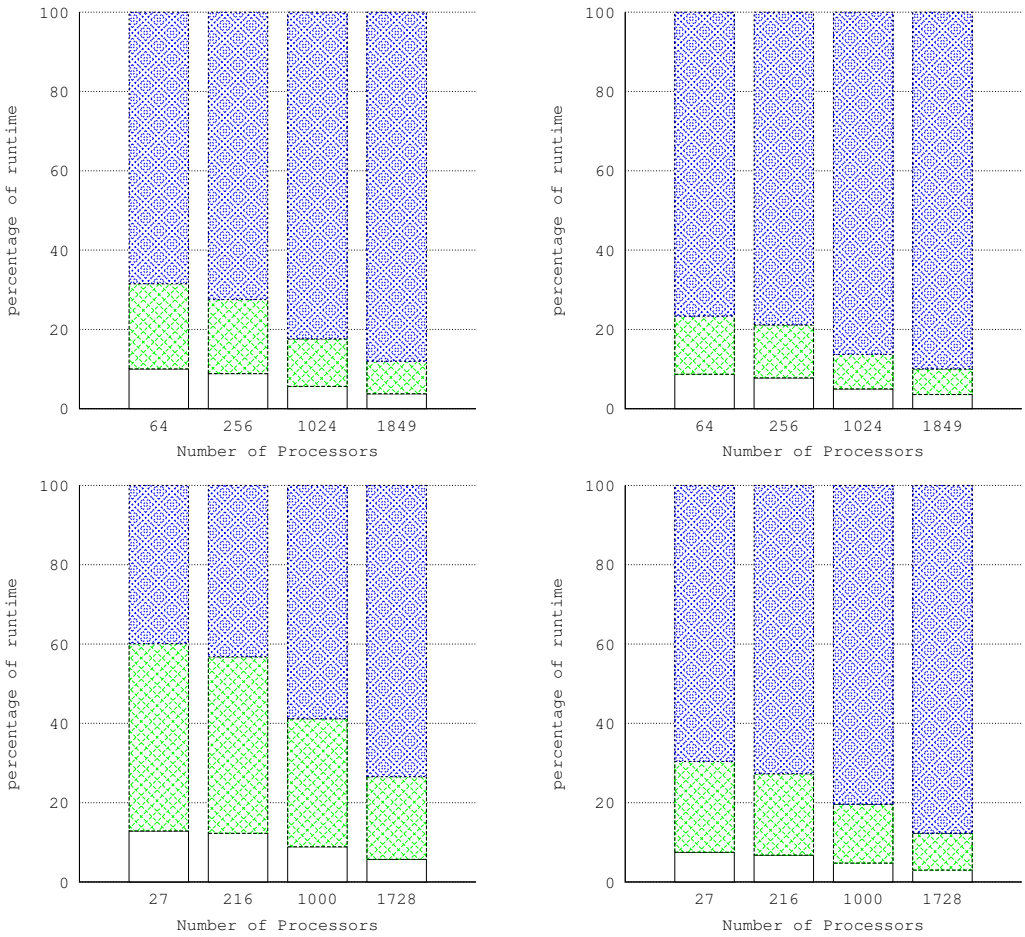


Figure 6.18: Runtime fraction results for the load rebalancing refinement weak scaling tests up to 1849 processors on Beskow supercomputer. Triangles(top left), squares (top right), tetrahedra(bottom left), hexahedra(bottom right). The full description of this case is given in Fig. 6.17. It performs a single level of forced load rebalancing refinement, and solves the stationary heat transfer problem with the resulting refined mesh. The linear solver is set to be the bi-conjugate gradient with the ML preconditioner. The refinement runtime fraction is plotted with a solid fill, load balancing runtime fraction is plotted with a gross pattern fill, and the linear system runtime fraction is plotted using the fine pattern fill.

6.7 Conclusions

In this chapter a novel parallel, hierarchical and load re-balanced algorithm for adaptive mesh refinement and coarsening of unstructured bidimensional and three-dimensional meshes has been presented.

The main features of the proposed algorithm are its suitability for nodally based partitions in distributed memory frameworks, together with the capability to successively refine and unrefine the mesh in an efficient manner in clusters of up to thousands of processors. Several different types of meshes can be dealt with by the algorithm, including triangular, quadrilateral, tetrahedral and hexahedral elements, and also meshes with several types of elements.

The memory requirements of the algorithm are reduced at the local level, because only the information corresponding to the part of the mesh in the current processor subdomain needs to be stored locally. The resulting refined meshes are non conforming with hanging nodes, but no balancing restriction needs to be enforced between adjacent elements, which allows having jumps of multiple refinement levels in neighbor elements.

A load rebalancing scheme has been presented which is independent of the rebalancing renumbering strategy, which can be chosen by the user. Both graph partitioning schemes and space-filling methods (or any other renumbering strategy for load rebalancing) can be used with the proposed algorithm.

Several numerical tests have been presented to illustrate the performance of the proposed algorithm. The first set of tests deals with simulation driven problems like the Poisson heat transfer problem and the incompressible Navier-Stokes equations in adaptive meshes, for which the expected behavior of the refinement algorithm is obtained. The second set of tests study the scalability of the algorithm in up to 2000 CPUs clusters where good weak scalability results are obtained for meshes of up to two thousand million elements. Also, the runtime fraction for the refinement process is reduced when compared to the runtime for solving linear systems of equations on the generated meshes, which ensures the suitability of the proposed algorithm for large computational physics problems in high-performance computing environments.

The proposed algorithm is packed in the `RefficientLib` Fortran 2003 library for which the user interface has been presented. This allows the easy integration (with no more than 10 calls to the adaptive refinement library) of the proposed algorithm with existing computational physics codes.

Chapter 7

Variational multi-scale error estimators for the adaptive mesh refinement of compressible flow simulations

This chapter describes an explicit a-posteriori error estimator for the finite element approximation of the compressible Navier-Stokes equations. The proposed methodology employes the variational multi-scale framework, and specifically, the idea is to use the variational subscales to estimate the error. These subscales are defined to be orthogonal to the finite element space, dynamic and non-linear, and both the subscales in the interior of the element and on the element boundaries are considered. Another particularity of the model is that we define some norms that lead to a dimensionally consistent measure of the compressible flow solution error inside each element; a scaled L^2-norm, and the calculation of a physical entropy measure, are both studied in this work. The estimation of the error is used to drive the adaptive mesh refinement of several compressible flow simulations.

7.1 Introduction

The compressible Navier-Stokes equations, namely the conservation of mass, momentum, and energy, together with constitutive and thermodynamical relations, constitute a physical model that describes the compressible fluid flow phenomena. This model is able to represent the wide range of spatial and temporal flow scales typically encountered in engineering cases of interest. When numerically approximating these equations, the smallest flow scales (in turbulence or aeroacoustics, for example) must be modeled with a high level of accuracy by numerical means, but one major source of error is the discretization error: the solution obtained with coarse meshes is often too inaccurate, and calculating over fine meshes is impractical considering the amount of computational effort. Adaptive Mesh Refinement (AMR) methods deal with this issue by dynamically re-configuring the initial mesh and changing its structure employing some type of criteria. The AMR involves two main steps: first, the decision of which elements to modify (mainly the ones contributing the most to the global solution error), and then, the adaptation of those selected elements. The focus of the present work is to investigate a local estimate of the approximation error that can be suitable for driving the AMR of compressible

flow simulations. To this end, we exploit the Variational Multi-Scale (VMS) framework introduced in [17], that is typically used to stabilize the fluid flow equations (see for instance the review of the VMS applied to fluid problems in [15]).

The VMS method decomposes the solution space into a resolved component, that is captured by the finite discretization, and a sub-grid part, which is the remainder that cannot be represented by the finite grid. The original discrete problem is therefore equivalent to two subproblems: one for each scale. Although the variational sub-grid scales (or subscales) have been also adopted for turbulence modeling (for example in Chapter 2 and references therein), essentially, the role of the subscales is related to the error of the finite approximation. Indeed, the variational subscales vanish consistently as the discrete solution tends to the exact solution, so that, they have been identified with *residual-based* error estimators (e.g in [144]). The recent literature on variational subscales error estimators is too vast to survey here, but we mention the early works of variational subscales as *a-posteriori* error estimates in [145, 146], where a patch of the elements in the mesh was used to calculate the subscales by decoupling a global residual equation with a localization function, and the *element-residual* methods in [147, 148], which were able to provide implicit estimations with the subscales inside each element of the mesh (and in some cases on the element boundaries). The application of the subscales as *explicit* a-posteriori estimators has been developed in [144, 149, 150] from the subscales at the interior and at the boundaries of the element using element Green's functions. In the case of incompressible flow simulations, the subscales have been used as error estimators for AMR in [151] by using a non-dimensional norm of the velocity subscales inside the element. However, the explicit a-posteriori error estimation given by the variational subscales has been less accurate than implicit goal-oriented methods for compressible flow problems in [18, 27–30], or than energy norm estimates using implicit residual methods in [152, 153], but cheaper, since the solution of additional differential equations is not required. We refer to the review article by [154] for a deeper understanding of the variational subscales as explicit a-posteriori error estimators.

The present approach is similar to the one in [150], where an *explicit* a-posteriori estimation of the error for the Euler and Compressible Navier-Stokes equations was constructed. We also derive an explicit a-posteriori error estimator from the VMS framework, but our approach offers three different and novel ingredients. First, we define the space where the variational subscales live as the space orthogonal to the finite element space, contrary to the most common choice to define it equal to the space of the finite element residuals. This is the so-called *Orthogonal Sub-Grid Scales* (OSGS) method, first introduced in [44], and which has been recently applied as an explicit a-posteriori error estimator for the elastic problem of solids in [155]. We also include the temporal tracking of the subscales, to what is referred in [46] as the *dynamic* subscales, and account for including the subscales into all of the *non-linear* terms of the problem. In consequence, we refer to the subscales as to be orthogonal, dynamic and non-linear. Second, we model the effect of the subscales inside each element from the perspective of a Fourier analysis, as developed in [85] or in Chapter 3, instead of using Green's functions in [17, 144, 149, 150]. Third, we model the subscales at the boundaries of the elements, as first presented in [109], and calculate them as part of the error estimator.

In this chapter, the error estimation given by the variational multiscale is intended to drive the adaptive mesh refinement process of compressible flow simulations. The error estimate must be well constructed in a dimensional sense, so that, the contribution of the subscales of

the different equations (i.e. the subscales of the mass, momentum and energy equations) into the estimation must be consistent. Several norms can be used in this regard. The L^1 and L^2 norms of the variational subscales of velocity in [151] and of the subscales of the separated compressible equations in [150], and the energy norm of the variational compressible problem in [153], have been previously applied to estimate the error inside each element. Since the error estimation given by each separated variational subscales (of the compressible equations) can vary greatly, and therefore, each refined mesh driven by the estimation of a single subscale may lead to unbalanced errors between the multiple variables of the problem, we propose to calculate the error estimation accounting for all the variational subscales. To this end, we propose two different approaches that provide dimensionally consistent measures of the solution error inside each element. The first approach is to calculate a scaled L^2-norm of the variational subscales inside the element, and on the element boundaries, as well. This approach goes in line to the one presented in [155]. The second approach is to compute the relative L^2 error norm of the physical entropy (for ideal gases) calculated using the variational subscales inside the element. The error estimation given by these two norms is used by the adaptive algorithm to drive the addition or removal of elements where the error is outside a given threshold. The *h-adaptive* method adopted here, including the refinement strategy and the data structures needed, has been presented in Chapter 6 and implemented in the `RefficientLib` software. Nevertheless, the proposed strategy is applicable to p, r, and hp adaptive mesh refinement techniques, although these are not explored in the present thesis. The AMR simulations demonstrate the ability of the variational subscales to lead an accurate approximation, provided the error estimation is kept always below the threshold inside the computational domain.

The chapter is divided into the following parts. In Section 7.2, the compressible flow problem is presented. Next, the VMS finite element formulation of the problem is described in Section 7.3. The design of the variational subscales error estimator is presented in Section 7.4. Some numerical examples, including subsonic and supersonic problems in two and three dimensions, are demonstrated in Section 7.5. Finally, in Section 7.6 some conclusions close the chapter.

7.2 Problem definition

In this section, we present the governing equations for the compressible fluid flow problem, i.e., the compressible Navier-Stokes equations. For doing so, the strong form of the problem is described in the first part of the section. Then, we transform the strong form of the problem into a quasi-linear form that allows us to deal with the non-linearities of the problem. The weak form of the compressible problem is introduced at the last part of the section.

7.2.1 Compressible Navier-Stokes equations in strong form

Consider a spatial domain $\Omega \subset \mathbb{R}^d$, being d the number of space dimensions ($d = 2$ or 3), and the time interval $(0, t_f)$. Let $t \in (0, t_f)$ be a given time instant in the temporal domain, and $\boldsymbol{x} \in \Omega$ a given point in the spatial domain. Let Γ be the boundary of the domain Ω, and $\boldsymbol{n}$ the geometric unit outward normal vector on Γ. We split Γ into two sets: the *Dirichlet* boundary denoted as Γ_G, and the *Neumann* boundary denoted as Γ_N. The strong form of the initial and

boundary-value problem consists of finding the solution vector $\boldsymbol{U} : \Omega \times (0, t_f) \rightarrow \mathbb{R}^{d+2}$, where $d+2$ is the number of unknowns (the same as equations of the system), such that for the given Dirichlet boundary conditions $\boldsymbol{U}_G : \Gamma_G \times (0, t_f) \rightarrow \mathbb{R}^{d+2}$ and the Neumann boundary conditions $\boldsymbol{H} : \Gamma_N \times (0, t_f) \rightarrow \mathbb{R}^{d+2}$, the following compressible Navier-Stokes equations are satisfied:

$$\begin{aligned}
\partial_t \boldsymbol{U} + \partial_j \boldsymbol{E}_j(\boldsymbol{U}) + \partial_j \boldsymbol{G}_j(\boldsymbol{U}) &= \boldsymbol{F} && \text{in} \quad \Omega \subset \mathbb{R}^d, \quad t \in (0, t_f), && (7.1)\\
\boldsymbol{D}(\boldsymbol{U}) &= \boldsymbol{D}(\boldsymbol{U}_G) && \text{on} \quad \Gamma_G, \quad t \in (0, t_f), && (7.2)\\
\boldsymbol{B}(\boldsymbol{U}) &= \boldsymbol{H} && \text{on} \quad \Gamma_N, \quad t \in (0, t_f), && (7.3)\\
\boldsymbol{U} &= \boldsymbol{U}^0(\boldsymbol{x}) && \text{in} \quad \Omega \subset \mathbb{R}^d, \quad t = 0. && (7.4)
\end{aligned}$$

The Eulerian time derivative and $\partial/\partial x_j$ are indicated in short notations ∂_t and ∂_j, respectively, and the usual summation convention is used over repeated indices. The Dirichlet boundary operator $\boldsymbol{D}(\cdot)$ is used to impose the components of $\boldsymbol{U}$ on different parts of Γ. The Neumann boundary conditions are given by the operator $\boldsymbol{B}(\cdot)$. Note that with our notation Γ_G and Γ_N may overlap.

The vector $\boldsymbol{U} = (\rho, \boldsymbol{m}, e_{\text{tot}})^\top$ denotes a vector of conservative variables, density ρ, momentum $\boldsymbol{m} = \rho \boldsymbol{u}$, and total energy $e_{\text{tot}} = \rho(e + \boldsymbol{u} \cdot \boldsymbol{u}/2)$, where $\boldsymbol{u}$ stands for the velocity vector, and e is the internal energy. The *convective* flux in the jth-direction, $j = 1, ..., d$, is defined as

$$\boldsymbol{E}_j(\boldsymbol{U}) = \left(\rho u_j, \rho u_j u_1 + p\delta_{1j}, \rho u_j u_2 + p\delta_{2j}, \rho u_j u_3 + p\delta_{3j}, u_j(e_{\text{tot}} + p)\right)^\top, \tag{7.5}$$

where p is the pressure and $\boldsymbol{I} = [\delta_{ij}]$ is the identity or *Kronecker* tensor. The *diffusive* flux in the jth-direction, $j = 1, ..., d$, is

$$\boldsymbol{G}_j(\boldsymbol{U}) = \left(0, -\sigma_{j1}, -\sigma_{j2}, -\sigma_{j3}, -u_i\sigma_{ij} + q_j\right)^\top, \tag{7.6}$$

where $\boldsymbol{\sigma}$ is the viscous stress tensor, $\sigma_{ij} = \mu(\partial_j u_i + \partial_i u_j) - \frac{2}{3}\mu(\partial_l u_l)\delta_{ij}$, and $\boldsymbol{q}$ is the heat flux vector, $q_i = -\lambda \partial_i T$. Here μ is the viscosity, λ is the thermal conductivity and T is the temperature of the fluid. The vector of source terms is defined as

$$\boldsymbol{F}(\boldsymbol{U}) = \left(0, \rho\boldsymbol{f}, \rho\boldsymbol{f} \cdot \boldsymbol{u} + \rho r\right)^\top, \tag{7.7}$$

where $\boldsymbol{f}$ is a body force vector, and r is a heat source/sink term.

In the previous relations the caloric equation $e = c_v(T)\,T$ and the ideal law for gases $p = \rho R T$ are used to calculate the pressure and the acoustic speed $c = \sqrt{\gamma p/\rho}$, where the specific heat at constant volume $c_v(T)$ and the specific heat at constant pressure $c_p(T)$ are thermodynamic properties of the fluid, $\gamma = c_p/c_v$ is the ratio between the specific heats, and $R = c_p - c_v$ is the specific gas constant. The non-dimensional Mach number $\mathrm{M} = |\boldsymbol{u}|/c$ is used to calculate the compressibility regime.

7.2.2 Quasi-linear form of the problem

Different formulations of the Navier-Stokes equations can be used, e.g. one can take $\boldsymbol{U}$ as the conservation variables, as done before, or one can also take $\boldsymbol{U} = (p, \boldsymbol{u}, T)^\top$, leading to the

primitive variables formulation. Any of such formulations can be written in quasi-linear form as

$$\boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t\boldsymbol{U}+\boldsymbol{\mathcal{L}}\left(\boldsymbol{U};\boldsymbol{U}\right)=\boldsymbol{F}\qquad\text{in}\quad\Omega\subset\mathbb{R}^d,t\in\left(0,t_f\right),\tag{7.8}$$

together with appropriate boundary and initial conditions. Here we have introduced the non-linear operator $\boldsymbol{\mathcal{L}}$, which is defined as

$$\boldsymbol{\mathcal{L}}\left(\widehat{\boldsymbol{U}};\boldsymbol{U}\right)=\boldsymbol{A}_j\left(\widehat{\boldsymbol{U}}\right)\partial_j\boldsymbol{U}-\partial_k\left(\boldsymbol{K}_{kj}\left(\widehat{\boldsymbol{U}}\right)\partial_j\boldsymbol{U}\right).\tag{7.9}$$

Matrices $\boldsymbol{A}_0\left(\boldsymbol{U}\right)$, $\boldsymbol{A}_j\left(\boldsymbol{U}\right)$, and $\boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)$, for $k,j=1,..,d$, are $(d+2)\times(d+2)$ matrices that depend upon $\boldsymbol{U}$, and that are appropriately defined for each type of formulation, as described in Chapters 3 and 4. Both the conservative and the primitive formulations will be applied in the next sections of this chapter.

7.2.3 Weak form of the problem

Let $\boldsymbol{\mathcal{W}}$ be the space of functions where, for each $t\in(0,t_f)$, the unknowns are well defined, with the appropriate regularity that we will not analyze here. Let us also denote by $(\cdot,\cdot)_\omega$ the integral of the product of two functions (scalar or vector-valued) in a domain ω, omitting the subscript when $\omega=\Omega$. Introducing the form

$$A(\boldsymbol{U};\boldsymbol{V},\boldsymbol{W}):=\left(\boldsymbol{V},\boldsymbol{A}_j\left(\boldsymbol{U}\right)\partial_j\boldsymbol{W}\right)+\left(\partial_k\boldsymbol{V},\boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j\boldsymbol{W}\right),\tag{7.10}$$

the variational form of the problem can be written as: find $\boldsymbol{U}:(0,t_f)\rightarrow\boldsymbol{\mathcal{W}}$ such that

$$\left(\boldsymbol{V},\boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t\boldsymbol{U}\right)+A(\boldsymbol{U};\boldsymbol{V},\boldsymbol{U})=\left(\boldsymbol{V},\boldsymbol{F}\right)+\left(\boldsymbol{V},\boldsymbol{H}\right)_{\Gamma_N},\quad t\in\left(0,t_f\right),\tag{7.11}$$

$$\left(\boldsymbol{V},\boldsymbol{U}\right)=\left(\boldsymbol{V},\boldsymbol{U}^0\right),\qquad t=0,\tag{7.12}$$

for all $\boldsymbol{V}$ in the adequate test functions space. The *Neumann* boundary operator is given by the diffusive fluxes

$$\boldsymbol{B}\left(\boldsymbol{U}\right)=-n_k\boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j\boldsymbol{U},\tag{7.13}$$

although part of the convective term could also be integrated by parts and contribute to the Neumann boundary conditions, in particular, the pressure term.

7.3 Finite element formulation

In this section, the finite element formulation of the compressible problem is described. The VMS formulation of the compressible problem is described in the beginning of the section. As it will be explained, the VMS framework is used for stabilizing the finite element approximation and allows us to use arbitrary interpolation spaces for the different variables of the problem. The finite element equation is described first. Then, we focus the attention on the solution of the subscales; we address the equation for the subscales at the interior of the element, and later, the approximation for the subscales at the element boundaries is addressed. In contrast to the previous chapters of the thesis, we now include the subscales at the element boundaries as a key ingredient in the estimation of the error. Finally, we describe the time integration scheme that is used for advancing in time at the end of the section.

7.3.1 Variational multi-scale framework

Let us first consider a finite-element partition $\mathcal{T}_h = \{K\}$ of the domain Ω. The diameter of the element partition is denoted by h. We define the test functions space $\boldsymbol{\mathcal{W}}_h \subset \boldsymbol{\mathcal{W}}$ as made of continuous piecewise polynomial functions in space. The Galerkin approximation to problem (7.11)-(7.12) can be stated as follows: find $\boldsymbol{U}_h : (0, t_f) \to \boldsymbol{\mathcal{W}}_h$ such that

$$\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{U}_h\right)\partial_t \boldsymbol{U}_h\right) + A(\boldsymbol{U}_h; \boldsymbol{V}_h, \boldsymbol{U}_h) = \left(\boldsymbol{V}_h, \boldsymbol{F}\right) + \left(\boldsymbol{V}_h, \boldsymbol{H}\right)_{\Gamma_N}, \quad t \in (0, t_f), \tag{7.14}$$

$$\left(\boldsymbol{V}_h, \boldsymbol{U}_h\right) = \left(\boldsymbol{V}_h, \boldsymbol{U}^0\right), \qquad t = 0, \tag{7.15}$$

for all $\boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h^0$, the discrete space of test functions (i.e., with components vanishing where Dirichlet conditions are prescribed on the boundary).

This approximation suffers from instability problems, which vary according to the way to construct $\boldsymbol{\mathcal{W}}_h$ (e.g. in the case of equally interpolating spaces).

The VMS framework introduced in [17], has been established for overcoming this issue. The idea of the VMS framework is to decompose the space of the unknowns into a finite-dimensional space $\boldsymbol{\mathcal{W}}_h$, and an infinite-dimensional one, $\widetilde{\boldsymbol{\mathcal{W}}}$, so that $\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{W}}_h \oplus \widetilde{\boldsymbol{\mathcal{W}}}$. The unknown and the test functions are accordingly split as $\boldsymbol{U} = \boldsymbol{U}_h + \widetilde{\boldsymbol{U}}$ and $\boldsymbol{V} = \boldsymbol{V}_h + \widetilde{\boldsymbol{V}}$, respectively. We shall refer to functions in $\boldsymbol{\mathcal{W}}_h$ as the *resolved* scales and to functions in $\widetilde{\boldsymbol{\mathcal{W}}}$ as the *error* or *subscales*.

Equation (7.11) can be equivalently written as the system of equations

$$\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t \boldsymbol{U}\right) + A(\boldsymbol{U}; \boldsymbol{V}_h, \boldsymbol{U}) = \left(\boldsymbol{V}_h, \boldsymbol{F}\right) + \left(\boldsymbol{V}_h, \boldsymbol{H}\right)_{\Gamma_N}, \tag{7.16}$$

for all $\boldsymbol{V}_h \in \boldsymbol{\mathcal{W}}_h^0,\ t \in (0, t_f)$, and

$$\left(\widetilde{\boldsymbol{V}}, \boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t \boldsymbol{U}\right) + A(\boldsymbol{U}; \widetilde{\boldsymbol{V}}, \boldsymbol{U}) = \left(\widetilde{\boldsymbol{V}}, \boldsymbol{F}\right) + \left(\widetilde{\boldsymbol{V}}, \boldsymbol{H}\right)_{\Gamma_N}, \tag{7.17}$$

for all $\widetilde{\boldsymbol{V}} \in \widetilde{\boldsymbol{\mathcal{W}}}^0,\ t \in (0, t_f)$, and likewise for the initial condition, Eq. (7.12). In Eq. (7.17), $\widetilde{\boldsymbol{\mathcal{W}}}^0$ is the space of subscale test functions.

7.3.1.1 Finite element equation

We first analyze the equation for the finite element scale (7.16). The *time-dependent* terms involving the temporal derivatives in the Left-Hand-Side (LHS) of Eq. (7.16) can be split as

$$\left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t \boldsymbol{U}\right) = \left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t \boldsymbol{U}_h\right) + \left(\boldsymbol{V}_h, \boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t \widetilde{\boldsymbol{U}}\right). \tag{7.18}$$

Similarly, the second term in the LHS of Eq. (7.16) can be split as

$$A(\boldsymbol{U}; \boldsymbol{V}_h, \boldsymbol{U}) = A\left(\boldsymbol{U}; \boldsymbol{V}_h, \boldsymbol{U}_h\right) + A\left(\boldsymbol{U}; \boldsymbol{V}_h, \widetilde{\boldsymbol{U}}\right). \tag{7.19}$$

For convenience, the first terms in $A\left(\boldsymbol{U};\boldsymbol{V}_h,\widetilde{\boldsymbol{U}}\right)$ can be integrated by parts, that is,

$$\begin{aligned}\left(\boldsymbol{V}_h,\boldsymbol{A}_j\left(\boldsymbol{U}\right)\partial_j\widetilde{\boldsymbol{U}}\right) =& -\sum_K\left(\partial_j\left(\boldsymbol{A}_j^\top\left(\boldsymbol{U}\right)\boldsymbol{V}_h\right),\widetilde{\boldsymbol{U}}\right)_K \\ &+\sum_K\left(n_j\boldsymbol{A}_j^\top\left(\boldsymbol{U}\right)\boldsymbol{V}_h,\widetilde{\boldsymbol{U}}\right)_{\partial K}, \qquad (7.20)\\ \left(\partial_k\boldsymbol{V}_h,\boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j\widetilde{\boldsymbol{U}}\right) =& -\sum_K\left(\partial_j\left(\boldsymbol{K}_{kj}^\top\left(\boldsymbol{U}\right)\partial_k\boldsymbol{V}_h\right),\widetilde{\boldsymbol{U}}\right)_K \\ &+\sum_K\left(n_j\boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)^\top\partial_k\boldsymbol{V}_h,\widetilde{\boldsymbol{U}}\right)_{\partial K}. \qquad (7.21)\end{aligned}$$

Note that these terms in (7.19) involve inter-element jumps. For continuous solution finite element spaces, the convective term jump at the element boundaries in the Right-Hand-Side (RHS) of (7.20) is continuous because it is a function of the variables and, therefore, its sum across adjacent element boundaries is zero. On the contrary, the diffusive term at the element boundaries in the RHS of (7.21) contains derivatives of the variables and it is discontinuous even for continuous finite element spaces.

If we introduce the formal adjoint $\boldsymbol{\mathcal{L}}^*\left(\boldsymbol{U};\cdot\right)$ of the operator $\boldsymbol{\mathcal{L}}\left(\boldsymbol{U};\cdot\right)$, which is

$$\boldsymbol{\mathcal{L}}^*\left(\boldsymbol{U};\boldsymbol{V}_h\right) = -\partial_j\left(\boldsymbol{A}_j^\top\left(\boldsymbol{U}\right)\boldsymbol{V}_h\right) - \partial_j\left(\boldsymbol{K}_{kj}^\top\left(\boldsymbol{U}\right)\partial_k\boldsymbol{V}_h\right), \qquad (7.22)$$

the term related to the subscales in the finite-element equation (7.19) can be written as

$$A\left(\boldsymbol{U};\boldsymbol{V}_h,\widetilde{\boldsymbol{U}}\right) = \sum_K\left(\boldsymbol{\mathcal{L}}^*\left(\boldsymbol{U};\boldsymbol{V}_h\right),\widetilde{\boldsymbol{U}}\right)_K + \sum_K\left(n_j\boldsymbol{K}_{kj}^\top\left(\boldsymbol{U}\right)\partial_k\boldsymbol{V}_h,\widetilde{\boldsymbol{U}}\right)_{\partial K}. \qquad (7.23)$$

We divide the approximation of the variational subscales: either we define the subscales in the interior of the element, or we define them at the element boundaries. The first term at the RHS of (7.23) is the stabilization term that includes the subscales in the interior of the element $\widetilde{\boldsymbol{U}}$, whereas, the second term is the element boundary term that relates the subscales at the internal and boundary edges, that we call $\widetilde{\boldsymbol{U}}_E$. We introduce some approximations to calculate the variational subscales; the way we deal with those approximations will be explained in the following paragraphs. In this work we also make the following simplification: we only account for the interior subscales in the solution of the finite element problem, whereas, the subscales at the boundary are calculated as part of the error estimator.

7.3.1.2 Subscales in the interior of the element

It is readily seen that, after integrating by parts the diffusive term of the LHS of (7.17), the equation for the subscales can be written as

$$\left(\widetilde{\boldsymbol{V}},\boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t\widetilde{\boldsymbol{U}} + \boldsymbol{\mathcal{L}}\left(\boldsymbol{U};\widetilde{\boldsymbol{U}}\right)\right) = \left(\widetilde{\boldsymbol{V}},\boldsymbol{F} - \boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t\boldsymbol{U}_h - \boldsymbol{\mathcal{L}}\left(\boldsymbol{U};\boldsymbol{U}_h\right)\right). \qquad (7.24)$$

At this point we introduce the main approximation

$$\boldsymbol{\mathcal{L}}\left(\boldsymbol{U};\widetilde{\boldsymbol{U}}\right) \approx \boldsymbol{\tau}^{-1}\left(\boldsymbol{U}\right)\widetilde{\boldsymbol{U}}, \qquad (7.25)$$

so that the application of the non-linear operator to the subscales is modeled by a matrix of stabilization parameters $\boldsymbol{\tau}^{-1}$ that depends over the unknowns. If $\widetilde{\boldsymbol{P}}$ denotes the L^2 projection onto the space of subscales, the equation for the subscales in the interior of the element (7.24) can be formally written as

$$\boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t\widetilde{\boldsymbol{U}}+\boldsymbol{\tau}^{-1}\left(\boldsymbol{U}\right)\widetilde{\boldsymbol{U}}=\widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{U};\boldsymbol{U}_h\right)\right], \tag{7.26}$$

where $\boldsymbol{R}\left(\boldsymbol{U};\boldsymbol{U}_h\right)$ stands for the finite residual,

$$\boldsymbol{R}\left(\boldsymbol{U};\boldsymbol{U}_h\right)=\boldsymbol{F}-\boldsymbol{A}_0\left(\boldsymbol{U}\right)\partial_t\boldsymbol{U}_h-\boldsymbol{\mathcal{L}}\left(\boldsymbol{U};\boldsymbol{U}_h\right). \tag{7.27}$$

In this work we define the space where the subscales belong as the orthogonal space to the finite element space, $\widetilde{\boldsymbol{W}}=\boldsymbol{W}_h^{\perp}$. This is the so called *Orthogonal Sub-Grid Scales* (OSGS) method, which defines the projection as the orthogonal projection onto the finite element space $\widetilde{\boldsymbol{P}}=\boldsymbol{P}_h^{\perp}=\boldsymbol{I}-\boldsymbol{P}_h$, being $\boldsymbol{P}_h$ the L^2-projection onto the finite element space. Apart from the construction of the spaces where the subscales belong, we call the subscales *dynamic* because the temporal derivative of subscales in (7.18) and (7.26) is taken into account, and *non-linear*, as the subscales are accounted for in all the non-linear terms of both the finite scale and subscale equations. This means that at all instances where $\boldsymbol{U}$ appears, it is replaced by $\boldsymbol{U}_h+\widetilde{\boldsymbol{U}}$.

We also adopt in the present formulation a diagonal matrix of stabilization parameters $\boldsymbol{\tau}^{-1}\left(\boldsymbol{U}\right)=\mathrm{diag}\left(\tau_c^{-1}\left(\boldsymbol{U}\right),\tau_m^{-1}\left(\boldsymbol{U}\right)\boldsymbol{I},\tau_e^{-1}\left(\boldsymbol{U}\right)\right)$, such that an approximation of the nonlinear operator applied to the subscales is made in each element. The definition for the diagonal matrix of stabilization parameters arises from the perspective of a Fourier analysis, as demonstrated in Chapter 3. These components are defined for each formulation as

	Conservative variables			Primitive variables
$\tau_c^{-1}\left(\boldsymbol{U}\right)=$	$C_2(\lvert\boldsymbol{u}\rvert+c)/h$	,	$\tau_c^{-1}\left(\boldsymbol{U}\right)=$	$\rho\tau_m/h^2$,
$\tau_m^{-1}\left(\boldsymbol{U}\right)=$	$\frac{C_1\nu}{h^2}+\frac{C_2(\lvert\boldsymbol{u}\rvert+c)}{h}$	,	$\tau_m^{-1}\left(\boldsymbol{U}\right)=$	$\frac{C_1\mu}{h^2}+\frac{C_2\rho u^\star}{h}$,
$\tau_e^{-1}\left(\boldsymbol{U}\right)=$	$\frac{C_1\alpha}{h^2}+\frac{C_2(\lvert\boldsymbol{u}\rvert+c)}{h}$	,	$\tau_e^{-1}\left(\boldsymbol{U}\right)=$	$\frac{C_1\lambda}{h^2}+\frac{C_2\rho c_v u^\star}{h}$.

In these expressions C_1 and C_2 are algorithmic parameters that we take as $C_1=12p^2$ and $C_2=2p$, where p is the order of the finite element interpolation. We also define the kinematic viscosity as $\nu=\mu/\rho$, and the thermal diffusivity as $\alpha=\lambda/\rho c_p$. For the primitive variables formulation, we follow the definition in Chapter 4 for the matrix of stabilization parameters, in which $u^\star$ is a modified velocity that is calculated with the Gauss error function, $u^\star=|\boldsymbol{u}|+\mathrm{erf}(\phi)c$, where ϕ is a normalized compressibility that is defined as $\phi=2-2(\epsilon-M)/\epsilon$. In the previous expression ϵ is a parameter that determines a certain transition from the compressible to the incompressible regime, which we assume as $\epsilon=0.1$ in the numerical examples presented in this chapter. Note that for the primitive variables formulation, the stabilization parameter for the momentum equation is dimensionally as the one for the conservative variables multiplied by the density. This is also the case of the stabilization parameter for the energy equation, which is dimensionally as the one for the conservative variables multiplied by ρc_p.

7.3.1.3 Subscales at the element boundaries

On the element boundaries, the subscales are calculated as follows. The main idea, proposed originally in [109], is to use the fact that the traction is continuous across element interfaces.

The weak continuity of the total fluxes implies that

$$\sum_K \left(\boldsymbol{V}_h, n_k \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \boldsymbol{U}\right)_{\partial K} = 0. \tag{7.28}$$

Suppose two elements K_1, K_2 that share an edge (face, in $d = 3$) E. The jump operator of a scalar function g across E is defined as

$$[\![\boldsymbol{n} g]\!]_E = \boldsymbol{n}^{(1)} g|_{\partial K_1 \cap E} + \boldsymbol{n}^{(2)} g|_{\partial K_2 \cap E}, \tag{7.29}$$

where $\boldsymbol{n}^{(1)}$ is the unit external normal to element K_1, and $\boldsymbol{n}^{(2)}$ is the unit external normal to element K_2. Therefore, the continuity of the fluxes can be imposed with the jump operator on each edge as

$$0 = [\![n_k \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \boldsymbol{U}_h]\!]_E + [\![n_k \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \widetilde{\boldsymbol{U}}]\!]_E.$$

We can write the second term on the RHS of the previous expression in the following manner:

$$[\![n_k \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \widetilde{\boldsymbol{U}}]\!]_E = \boldsymbol{\mathcal{F}}^{(1)}_{\partial K_1 \cap E} + \boldsymbol{\mathcal{F}}^{(2)}_{\partial K_2 \cap E} \tag{7.30}$$

The approximation of the method is given by the supposition that the fluxes related to the subscales at the boundary $\widetilde{\boldsymbol{U}}_E$ are calculated respectively as

$$\boldsymbol{\mathcal{F}}^{(1)}_{\partial K_1 \cap E} \approx \boldsymbol{\mathcal{K}} \frac{\widetilde{\boldsymbol{U}}_E - \widetilde{\boldsymbol{U}}^{(1)}}{\delta}, \quad \text{and} \quad \boldsymbol{\mathcal{F}}^{(2)}_{\partial K_2 \cap E} \approx \boldsymbol{\mathcal{K}} \frac{\widetilde{\boldsymbol{U}}_E - \widetilde{\boldsymbol{U}}^{(2)}}{\delta}. \tag{7.31}$$

Here we approximate the subscales $\widetilde{\boldsymbol{U}}^{(i)}$ at the interior of element i, up to a distance $\delta = \delta_0 h$ to the element boundary, $0 \leq \delta_0 \leq 1/2$. We may introduce $\boldsymbol{\mathcal{K}}$ as an *approximation* for the diffusion matrix $\boldsymbol{K}_{ij}\left(\boldsymbol{U}\right)$, for $1 \leq i, j \leq d$, which we define in this work for the conservative and primitive formulations, respectively as

$$\boldsymbol{\mathcal{K}} = \begin{bmatrix} 0 & \boldsymbol{0}^\top & 0 \\ \boldsymbol{0} & \nu \boldsymbol{I} & \boldsymbol{0} \\ 0 & \boldsymbol{0}^\top & \alpha \end{bmatrix} \quad \text{and} \quad \boldsymbol{\mathcal{K}} = \begin{bmatrix} 0 & \boldsymbol{0}^\top & 0 \\ \boldsymbol{0} & \mu \boldsymbol{I} & \boldsymbol{0} \\ 0 & \boldsymbol{0}^\top & \lambda \end{bmatrix}. \tag{7.32}$$

Nevertheless, other types of definitions for the subscale fluxes could be implemented, as finite-difference-like methods.

With the previous considerations, we obtain a definition for the subscales at the internal edges of the finite element mesh

$$\widetilde{\boldsymbol{U}}_E = \{\widetilde{\boldsymbol{U}}\}_E - \boldsymbol{\mathcal{K}}^{-1} \frac{\delta}{2} [\![n_k \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \boldsymbol{U}_h]\!]_E, \tag{7.33}$$

where $\{\widetilde{\boldsymbol{U}}\}_E$ stands for $\{\widetilde{\boldsymbol{U}}\}_E = (\widetilde{\boldsymbol{U}}^{(1)} + \widetilde{\boldsymbol{U}}^{(2)})/2$. As explained in [109], we can neglect the contribution of this term, so that the calculation of the subscales at the edges is given by

$$\widetilde{\boldsymbol{U}}_E = -\boldsymbol{\tau}_E [\![n_k \boldsymbol{K}_{kj}\left(\boldsymbol{U}\right)\partial_j \boldsymbol{U}_h]\!]_E. \tag{7.34}$$

The previous equation can be seen as the usual definition for the subscales calculation, this is, to use a boundary matrix of stabilization parameters which accounts for $\boldsymbol{\tau}_E = \boldsymbol{\mathcal{K}}^{-1} \delta_0 h/2$. In this work, the algorithmic constant $\delta_0 = 1/2$ is used.

7.3.2 Time integration method

We partition the time interval $(0, t_f)$ in a sequence of discrete time steps $0 = t^0 < t^1 < ... < t^N = t_f$, with $\delta t > 0$ the time step-size defining $t^{n+1} = t^n + \delta t$ for $n = 0, 1, 2, ..., N$. We implement an implicit monolithic time integration scheme in order to integrate the time derivatives of Eqs. (7.16) and (2.14). More specifically, we use the Backward Differentiation Formula (BDF) scheme. For the time dependent function $y(t)$, the BDF approximation of order $k = 1, 2, ..,$ is given by $\delta_k y^{n+1}/\delta t$, with

$$\delta_k y^{n+1} = \frac{1}{\gamma_k}\left(y^{n+1} - \sum_{i=0}^{k-1} \phi_k^i y^{n-1}\right),$$

where γ_k and ϕ_k^i are numerical parameters. In particular, we use the first order scheme for discretizing the transient term of the subscales in Eq. (2.14). The solution of the subscales in the interior of the element at the time step $n+1$ is computed from

$$\widetilde{\boldsymbol{U}}^{n+1} = \boldsymbol{\tau}_t\left(\boldsymbol{U}^{n+1}\right)\left(\widetilde{\boldsymbol{P}}\left[\boldsymbol{R}\left(\boldsymbol{U}^{n+1}; \boldsymbol{U}_h^{n+1}\right)\right] + \boldsymbol{A}_0\left(\boldsymbol{U}^{n+1}\right)\frac{\widetilde{\boldsymbol{U}}^n}{\delta t}\right), \tag{7.35}$$

where the dynamic operator $\boldsymbol{\tau}_t\left(\boldsymbol{U}^{n+1}\right)$ is defined as

$$\boldsymbol{\tau}_t\left(\boldsymbol{U}^{n+1}\right) = \left(\frac{1}{\delta t}\boldsymbol{A}_0\left(\boldsymbol{U}^{n+1}\right) + \boldsymbol{\tau}^{-1}\left(\boldsymbol{U}^{n+1}\right)\right)^{-1}. \tag{7.36}$$

In the case of the primitive variables formulation, we avoid the off-diagonal terms that appear in matrix $\boldsymbol{A}_0\left(\boldsymbol{U}^{n+1}\right)$ by approximating (7.36) as described in Chapter 4. Hence, for the primitive variables formulation we take the following diagonal definition of the dynamic operator:

$$\boldsymbol{\tau}_t\left(\boldsymbol{U}^{n+1}\right) = \text{diag}\left(\left(\frac{\rho^{n+1}}{p^{n+1}\delta t} + \tau_c^{-1}\right)^{-1}, \left(\frac{\rho^{n+1}}{\delta t} + \tau_m^{-1}\right)^{-1}\boldsymbol{I}, \left(\frac{\rho^{n+1}c_p}{\delta t} + \tau_e^{-1}\right)^{-1}\right). \tag{7.37}$$

7.4 Variational subscales as error estimator

The stabilized variational formulation for the compressible problem has been presented in the previous section. Regarding the discrete finite element approximation, it is identified with the coarse scales, while the variational subscales are related to the solution error. In this section, the error estimation (computed from the variational subscales) is presented. First, we introduce some additional approximations for calculating the subscales at the boundaries of the element. At the end of the section, we present the norms in which the error is measured.

7.4.1 Approximation for the subscales at the element boundaries

The equation for the subscales at the element interior (2.14) is a non-linear ordinary differential equation that is typically solved at each integration point inside the element. This follows from

the fact that subscales need to be included in the finite element scales (7.23) at the integration points. Nevertheless, the subscales (as error estimation) can be calculated at any location in the interior of the element (e.g. at the center of the element and/or at the nodes). Instead, the subscales at the element boundaries (7.34) are defined by the inter-element jump operator, and therefore, the contribution of neighbor elements must be accounted for. This type of calculation is inconvenient for parallel implementations of the AMR strategy, in which neighboring elements may be located on a different partition of the computational domain. To overcome this difficulty, we follow the approach in [155] for approximating the calculation of the subscales at the element boundaries: the inter-element jump can be bounded by an orthogonal projection in the interior of the element.

Let us first explain how the boundary subscales are developed. Expanding (7.34) and writing it for each equation of the compressible problem, results in

$$\widetilde{\boldsymbol{m}}_E = -\frac{\delta_0 h}{2}\left(\nu^{-1}\boldsymbol{I}\right)[\![\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E, \qquad \widetilde{\boldsymbol{u}}_E = -\frac{\delta_0 h}{2}\left(\mu^{-1}\boldsymbol{I}\right)[\![\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E,$$
$$\widetilde{e}_{\text{tot},E} = \frac{\delta_0 h}{2}\left(\alpha^{-1}\right)[\![\boldsymbol{q}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E, \qquad \widetilde{T}_E = \frac{\delta_0 h}{2}\left(\lambda^{-1}\right)[\![\boldsymbol{q}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E,$$

for the conservative and primitive variables, respectively. Note that no subscales over the boundaries result in the case of the continuity equation.

As described before, we approximate the jump of the fluxes. It has been shown in [156] that *any jump at the boundaries of the elements posses a lower, and an upper bound, in the sense that*

$$\gamma_1 \sum_K h_K \int_{\partial K} |[\![\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E|^2 \leq \|\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right) - P_h\left(\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right)\right)\|^2 \leq \gamma_2 \sum_K h_K \int_{\partial K} |[\![\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E|^2, \tag{7.38}$$

for certain γ_1 and γ_2. This introduces the possibility of approximating the L^2-norm of the jumps at the boundaries with

$$\int_{\partial K} |[\![\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E|^2 \approx \gamma h_K^{-1} \|\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right) - P_h\left(\boldsymbol{\sigma}\left(\boldsymbol{U}_h\right)\right)\|_K^2. \tag{7.39}$$

A similar result is obtained for the jump over the heat flux:

$$\int_{\partial K} |[\![\boldsymbol{q}\left(\boldsymbol{U}_h\right)\cdot\boldsymbol{n}]\!]_E|^2 \approx \gamma h_K^{-1} \|\boldsymbol{q}\left(\boldsymbol{U}_h\right) - P_h\left(\boldsymbol{q}\left(\boldsymbol{U}_h\right)\right)\|_K^2. \tag{7.40}$$

In these approximations, γ is a constant that can be calibrated (not to be confused with the ratio of specific heats). Another way to see this approximation is that we are replacing the boundary integral terms by a Zienkiewicz-Zhu estimator [143] over stresses and heat fluxes.

7.4.2 Error estimator measures

Providing a measure of the error for the finite element solution of all variables of the compressible problem is not trivial. Because the equations are non-symmetric and positive definite, there

is not a single energy norm (which exists in other fluid flow problems) that could give an estimate of the solution error. Moreover, as we have subscales for each of the equations of the problem, we must define an appropriate norm for estimating the error that includes the contribution of every subscale. In the following, we mention two different norms that we develop to compute an error estimator that can be used in AMR. It is obvious that many other alternatives would be possible.

7.4.2.1 Scaled L_2-Norm

One possibility is to define a scaled L^2-norm of the subscales of the type $|\widetilde{\boldsymbol{U}}|^2_{\boldsymbol{S}} = \widetilde{\boldsymbol{U}}^{\top}\boldsymbol{S}\widetilde{\boldsymbol{U}}$. The scaling matrix $\boldsymbol{S}$ is intended to guarantee the dimensional consistency between the different subscales. This scaled norm can be applied considering the interior subscales and the subscales at the element boundaries, such that the error estimator at each element is computed as

$$\eta_K^2 := |\widetilde{\boldsymbol{U}}|^2_{\boldsymbol{S}} + |\widetilde{\boldsymbol{U}}_E|^2_{\boldsymbol{S}}, \tag{7.41}$$

where it is understood that $\widetilde{\boldsymbol{U}}$ is restricted to K (and $E \in \partial K$). We adopt the diagonal matrix of stabilization parameters $\boldsymbol{\tau}$ as part of the scaling matrix.

Introducing a reference velocity of the fluid flow problem $u_0 = |\boldsymbol{u}_h| + c$, we can take for the conservative variables formulation a scaling matrix $\boldsymbol{S} = \text{diag}\left(\tau_c^{-1}u_0^2, \tau_m^{-1}\boldsymbol{I}, \tau_e^{-1}u_0^{-2}\right)$, that gives the following dimensionally-consistent estimation of the error at each element:

$$\begin{aligned}\eta_K^2 := &\int_K \tau_c^{-1}(|\boldsymbol{u}_h| + c)^2|\widetilde{\rho}|^2 + \int_K \tau_m^{-1}|\widetilde{\boldsymbol{m}}|^2 + \int_K \tau_e^{-1}(|\boldsymbol{u}_h| + c)^{-2}|\widetilde{e}_{\text{tot}}|^2 \\ &+ \int_{\partial K} \tau_{m,E}^{-1}|\widetilde{\boldsymbol{m}}_E|^2 + \int_{\partial K} \tau_{e,E}^{-1}(|\boldsymbol{u}_h| + c)^{-2}|\widetilde{e}_{\text{tot},E}|^2.\end{aligned} \tag{7.42}$$

In the case of the primitive variables formulation, the scaling matrix $\boldsymbol{S}$ can be written as $\boldsymbol{S} = \text{diag}\left(\tau_c^{-1}\rho_h^{-1}, \tau_m^{-1}\boldsymbol{I}, \tau_e^{-1}T_h^{-1}\right)$, being ρ_h and T_h the finite element approximation of the density and temperature, respectively. Therefore, in the case of the primitive formulation the error estimation at each element is given by the following dimensionally-consistent norm:

$$\begin{aligned}\eta_K^2 := &\int_K \tau_c^{-1}\rho_h^{-1}|\widetilde{p}|^2 + \int_K \tau_m^{-1}|\widetilde{\boldsymbol{u}}|^2 + \int_K \tau_e^{-1}T_h^{-1}|\widetilde{T}|^2 \\ &+ \int_{\partial K} \tau_{m,E}^{-1}|\widetilde{\boldsymbol{u}}_E|^2 + \int_{\partial K} \tau_{e,E}^{-1}T_h^{-1}|\widetilde{T}_E|^2.\end{aligned} \tag{7.43}$$

7.4.2.2 Entropy measure

Another possibility is to construct the error measure based on the calculation of an entropy functional. Assuming that the fluid possesses a constant specific heat at constant volume, the fundamental equation for the perfect gas is

$$s = s_0 + c_v \ln\left(\frac{e}{e_0}\right) - R\ln\left(\frac{\rho}{\rho_0}\right), \tag{7.44}$$

where s_0, e_0, and ρ_0 are reference values of entropy, internal energy, and density, respectively. Since the perfect gas relation can be written as $p = \rho\,(\gamma - 1)\,e$, and the caloric equation relates the internal energy with temperature, the previous relation can be developed as

$$s = s_0 + c_v \ln\left(\frac{p}{\rho^\gamma}\right) + c_v \ln\left(\frac{\rho_0^{\gamma-1}}{(\gamma-1)}\right) - c_v \ln e_0. \tag{7.45}$$

Therefore, an *entropy function* can be defined for the compressible problem as

$$s = c_v \ln\left(\frac{p}{\rho^\gamma}\right) + \hat{s}_0, \tag{7.46}$$

with $\hat{s}_0$ denoting the coefficients of (7.45) which are reference quantities. Since calculating an entropy with the subscales alone may lead to unphysical results, the entropy is used as an estimator by calculating the elemental relative L^2-error between the entropy of the finite solution including the subscales $\widetilde{s}$, and the entropy of the finite solution alone s_h. This is,

$$\eta_K^2 := \int_K \frac{\left(s(\boldsymbol{U}_h) - s(\boldsymbol{U}_h + \widetilde{\boldsymbol{U}})\right)^2}{s(\boldsymbol{U}_h)^2} \tag{7.47}$$

For the conservative variables formulation, we calculate the entropy as

$$s(\boldsymbol{U}) = c_v \ln\left(\frac{(\gamma-1)\left(e_{\text{tot}} - \frac{|\boldsymbol{m}|^2}{2\rho}\right)}{\rho^\gamma}\right). \tag{7.48}$$

Similarly, for the primitive variables formulation, the entropy is calculated as

$$s(\boldsymbol{U}) = c_v \ln\left(\frac{p}{\left(\frac{p}{RT}\right)^\gamma}\right). \tag{7.49}$$

7.5 Numerical Examples

The error estimator presented before is now tested in AMR simulations of compressible flow problems. Two steady exact solution examples are solved in the first part of this section. The simulation of exact solution problems is mainly devoted to highlighting the behavior of the subscales as explicit error estimators. Then, the three-dimensional lid-driven cavity and the unsteady differentially heated cavity examples are intended to study the error estimation in subsonic flow problems. The primitive variables formulation is used in these cases, since this formulation simplifies to the incompressible equations when the incompressible constraint is included, and consequently, it is well defined in the low Mach number limit. On the contrary, the conservative variables formulation admits physically meaningful solutions when the solution develops discontinuities, as in the case of supersonic shocks, and consequently, we use that formulation to solve the last two numerical examples: the flow over a flat plate and the flow past a cylinder supersonic flow problems. In the case of those supersonic flow examples, the conservative variables formulation is enhanced with the orthogonal projection based

shock capturing and the anisotropic imposition of the added numerical diffusivity introduced in Chapter 3.

As described in Section 7.3, we implement an implicit monolithic time integration scheme in order to integrate the time derivatives of Eqs. (7.16) and (2.14). More specifically, we use the second order accurate Backward Differentiation Formula (BDF) scheme, so that, at each time step we solve the non-linearities of both the finite element and sub-grid scales by using Picard's scheme. This leads to a monotonically decreasing relative error between consecutive iterations, with the subsequent convergence of the numerical method. At most ten iterations are performed, fulfilling the maximum relative numerical tolerance for the L^2 norm iteration residual of 10^{-10}.

As commented before, the refinement process is held locally to counterbalance the error through the computational domain. The basic idea is, to begin with a given ($0-$level) coarse grid, and then to subsequently perform refinement (or coarsening). After each time step of the transient problem is solved we estimate the local error using (7.41) or (7.47). The error estimation is used to refine or coarsen the elements of the mesh depending on a given tolerance criteria (or threshold). The algorithm for refining the mesh is based on the h-adaptivity method of Chapter 6 that splits the element into sub-elements, or removes sub-elements to coarsen the mesh (being the 0-level mesh the coarsest possible mesh). We advance in time and adapt the mesh until a temporal convergence criterion is satisfied. In the case of non-transient examples, we perform the refinement step after each Picard's iteration. From now onwards the flow is considered as an ideal gas, with ratio of specific heats $\gamma = 1.4$ and physical properties $c_p = 1.010$ kJ/(kg K) and $c_v = 0.718$ kJ/(kg K).

7.5.1 Smooth exact solution

The first numerical example is intended to study the ability of the subscales to act as an explicit error estimation. In this case, we evaluate the estimation of the error given by the subscales, specifically, we consider the error measured with the scaled L^2-norm of an exact solution problem. The exact solution problem is solved by calculating an exact force, computed with the residual of the continuous problem, that is used in the discrete problem (also known as the method of manufactured solutions).

We use the primitive variables formulation to solve this numerical example. The problem is defined inside a rectangular domain $[0, L] \times [0, L]$, with $L = 1$ m, with the viscosity fixed to $\mu = 0.1$ kg/(m s), and the thermal conductivity to $\lambda = 1000$ W/(m K). The polynomial functions that we use as a steady two-dimensional manufactured solution for the compressible Navier-Stokes equations are given as follows:

$$\begin{aligned} p &= x_1^2 x_2^2 (x_1 - 1)(x_2 - 1) + 0.01, \\ u_1 &= 2x_1^2 x_2 (x_1 - 1)^2 (x_2 - 1)(2x_2 - 1), \\ u_2 &= -2x_1 x_2^2 (x_1 - 1)(x_2 - 1)^2 (2x_1 - 1), \\ T &= 2x_1^2 x_2 (x_1^2 - 1)(x_2 - 1) + 0.01. \end{aligned}$$

This type of problem is favorable to analyze the error estimation. Indeed, the difference between the discrete solution and the exact solution can be calculated, and it can be compared to the error given by the subscales. In this numerical example, we restrain the analysis to the

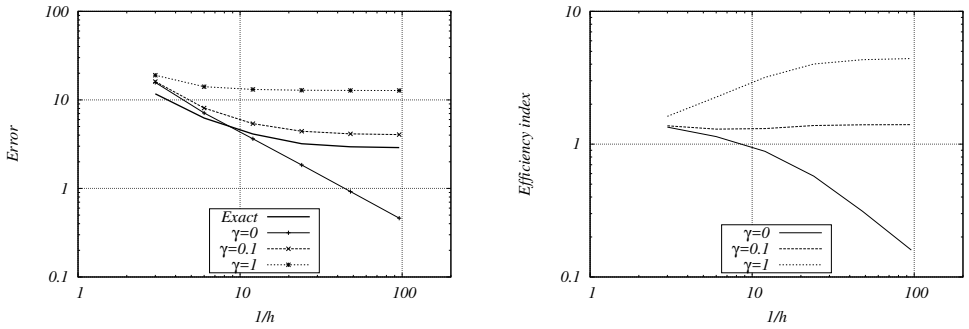


Figure 7.1: Smooth exact solution results: convergence of the exact error and the subscales error estimator at the left, and efficiency index as a function of the mesh size at the right.

scaled L^2-norm (presented in Section 7.4), as a measure of the subscales-based error, and we also use this norm to calculate the error against the exact solution. We prescribe the tolerance below the subscales-based error estimation through the simulation, so that, the algorithm refines uniformly the mesh.

This example is used to adjust the γ parameter of approximation (7.38), which is made to compute the integral of the subscales at the element boundaries. To this end, we perform different homogeneous refinement simulations varying the γ parameter and evaluate the efficiency of each subscales-based error estimation. Figure 7.1 displays the error convergence of the exact error compared with the subscales-based error estimator for the different values of γ. The efficiency index of the subscales-based estimator is also presented on the right side of this figure. In the convergence plot, we see that neglecting the subscales at the boundary ($\gamma = 0$) underestimates the error. Fixing $\gamma = 0.1$ gives the most adjusted error estimation in contrast to the exact error. It can be observed that the efficiency index of the estimator for this value is very good. The case of having $\gamma > 0.1$ overestimates the error, which is also observed in the detached result for the efficiency index. Hence, we fix $\gamma = 0.1$ to approximate the integral of the subscales over the boundary of the element in the following numerical examples.

We also show the spatial distribution of the exact error and the subscales-based error estimator in Fig. 7.2, after some refinement steps have been done. We see how it properly matches the spatial distribution of the exact error, and thus we validate the ability of the subscales-based error estimator to capture the error associated with the discretization error.

7.5.2 Singular exact solution

The second numerical example is a two-dimensional exact solution comprising a singularity in the solution. This allows us to evaluate the performance of the subscales as an error estimator when large localized gradients appear. We also use the primitive variables formulation to investigate the separated contribution of the subscales associated to each equation of the compressible problem into the estimation of the error. The exact solution is evaluated in the

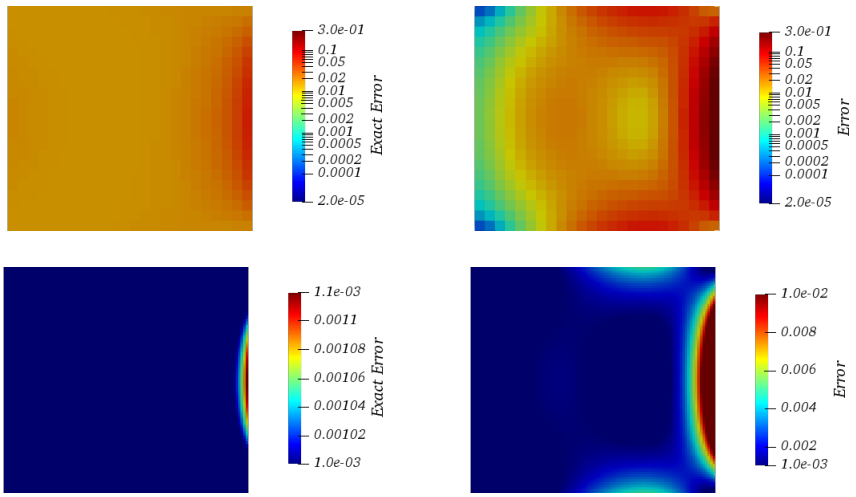


Figure 7.2: Smooth exact solution results: scaled L^2-norm of the exact error at the left, and scaled L^2-norm of the subscales at the right. Top: after three refinements. Bottom: after five refinements.

$L-$shaped domain $([-1,1]\times[-1,1])\setminus([0,1]\times[-1,0])$, with the following functions:

$$\begin{aligned} p(r,\phi) =& -r^{\alpha-1}\frac{(1+\alpha)^2\varphi'(\phi)+\varphi'''(\phi)}{1-\alpha}, \\ \boldsymbol{u}(r,\phi) =& r^{\alpha}\begin{bmatrix}\cos(\phi)\varphi'(\phi)+(1+\alpha)\sin(\phi)\varphi(\phi)\\ \sin(\phi)\varphi'(\phi)-(1+\alpha)\sin(\phi)\varphi(\phi)\end{bmatrix}, \\ T(r,\phi) =& r^{\alpha}\left(\cos(\phi)\varphi'(\phi)+(1+\alpha)\sin(\phi)\varphi(\phi)\right), \end{aligned}$$

with r and ϕ being the polar coordinates, and the function φ defined as

$$\varphi(\phi)=\frac{\sin((1+\alpha)\phi)\cos(\alpha\omega)}{1+\alpha}-\cos((1+\alpha)\phi)+\frac{\sin((1+\alpha)\phi)\cos(\alpha\omega)}{1-\alpha}+\cos((\alpha-1)\phi). \tag{7.50}$$

Here we take $\omega=3\pi/2$ and α as the (approximate) root of the following non-linear equation:

$$\frac{\sin^2(\alpha\omega)-\alpha^2\sin^2(\omega)}{\alpha^2}=0. \tag{7.51}$$

In this case the viscosity fixed to $\mu=0.1$ kg/(m s), and the thermal conductivity to $\lambda=25700$ W/(m K).

We first study the comparison between the error estimator and the exact solution error after several refinements steps. We use the scaled L^2-norm as the measure of the error, and let the mesh adaptation algorithm advance with the tolerance prescribed to $10^{-2}<\eta<10^{-1}$. Figure 7.3 shows the resulting fields, including the subscales-based error distribution over the refined mesh. The singularity in the pressure field near the corner of the domain can be appreciated at the top of this figure. It can also be seen that the refined mesh especially describes the singular point of the solution.

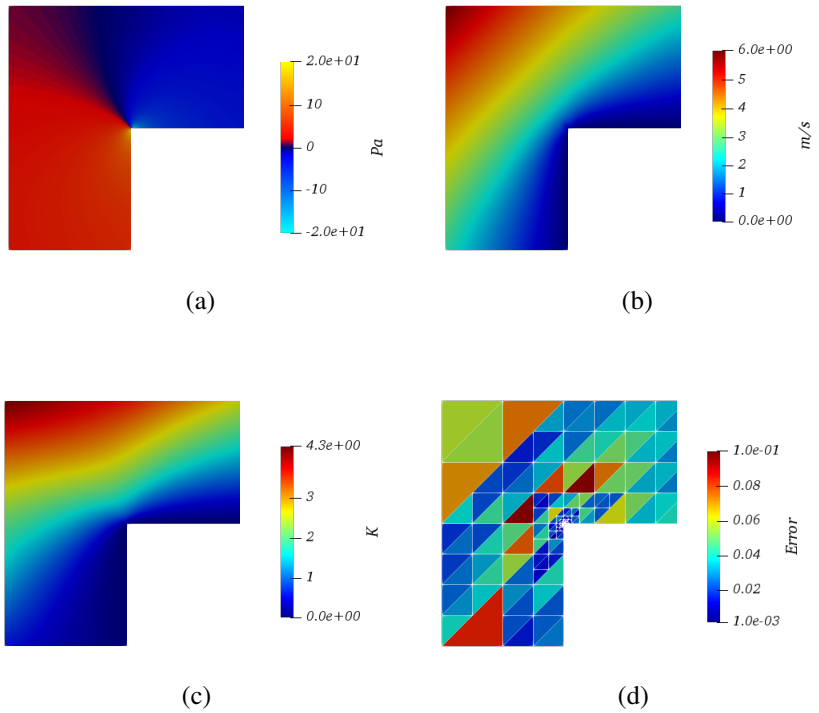
We also use this numerical example to analyze the error estimation given by the separated subscales (and the consequent type of refinement). Again, we use the scaled L^2-norm, but we account for the separated contribution of the subscales associated to each equation of the problem. The refined meshes and the subscales-based error measured with the scaled L^2-norm are displayed in Fig. 7.4. In this figure, we can see how accounting for the subscales of the mass equation refines the singular point, but the estimated error is high elsewhere. For the momentum subscales, this is not the case, it describes the singular point, but the estimated error is also low in the complete domain. In the case of the subscales of energy, there is not a description of the singularity, yet the estimated error is low.

Another analysis that we make is the comparison between a homogeneous refinement of the mesh and the one driven by the subscales-based error estimator. Figure 7.5 shows the convergence of the exact error (in the scaled L^2-norm) against the number of elements. At the left side of this figure, we plot the exact error convergence in the case of the AMR driven by subscales comparison against the homogeneously refined solution. The plot demonstrates the improved convergence of the exact error with the use of the subscales-based error (labeled as $p\boldsymbol{u}T$) in contrast to the homogeneous refinement, which is not able to accurately represent the singularity. Indeed, the exact error for the AMR converges below the homogeneous refinement at a smaller number of elements in the mesh, from which we conclude that the subscales-based error estimator is suitable for problems in which the solution presents singularities. We also present the computational cost of the subscales-based AMR and of the homogeneous refinement at the right side of Fig. 7.5. We observe that for the refinement with the subscales-based error estimator the computational effort flattens when the error converges. This is not achievable with the homogeneous refinement, as the computational effort grows with respect to the refinement.

Finally, we test the error estimation given by the separated contribution of the subscales and analyze the error convergence against the number of elements. We perform the refinement of the mesh with the error estimation given by the separated contribution of the subscales (associated with each equation of the problem). The exact error convergence of the different subscale estimations are presented in Fig. 7.6. We observe that the error diverges when the mass equation subscales (labeled p) leads the refinement. For overcoming this problem, we see that the subscales of momentum and energy equations are crucial, and therefore, the complete contribution of all subscales in the scaled L^2-norm improves the convergence of the error. In other words, the subscales of momentum (labeled $\boldsymbol{u}$ in the plot) and energy (labeled T) contribute the most to the error convergence; as long as these terms are considered both in the interior and on the boundaries of the elements.

7.5.3 Three-dimensional lid-driven cavity

The third case that we solve is the three-dimensional lid-driven cavity. We use this example to compare the scaled L^2-norm estimation of the subscales-based error with the one given by the entropy measure. The three-dimensional lid-driven cavity problem is defined as a prismatic cavity $[0, L] \times [0, L] \times [0, L]$, with $L = 1$ m. The flow is initially at rest, with a homogeneous pressure of 0.7124 Pa and a homogeneous temperature of 0.0024 K. The upper wall (x_1, x_2, L) is constantly moving at a fixed velocity of $(1, 0, 0)$ m/s. For the upper boundary, the temperature is also set to 0.0024 K. A no-slip condition for velocity, an adiabatic condition for energy,



(a) (b)

(c) (d)

Figure 7.3: Singular exact solution results: (a) pressure contour, (b) velocity magnitude contour, (c) temperature contour, and (d) refined mesh with the subscales-based error distribution.

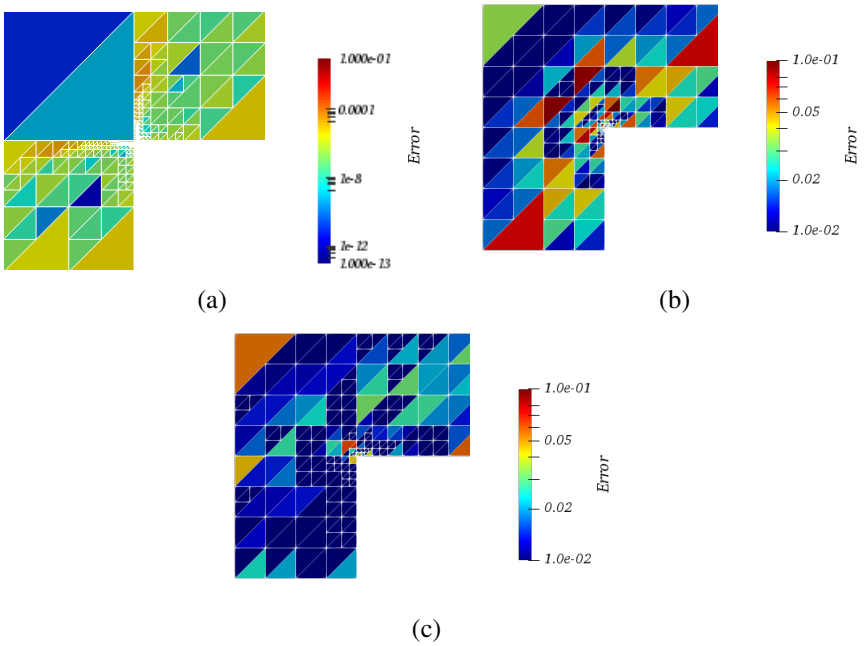


Figure 7.4: Singular exact solution results. Refined mesh and subscales-based error measured with the scaled L^2-norm considering only the subscales of (a) mass, (b) momentum, and (c) energy.

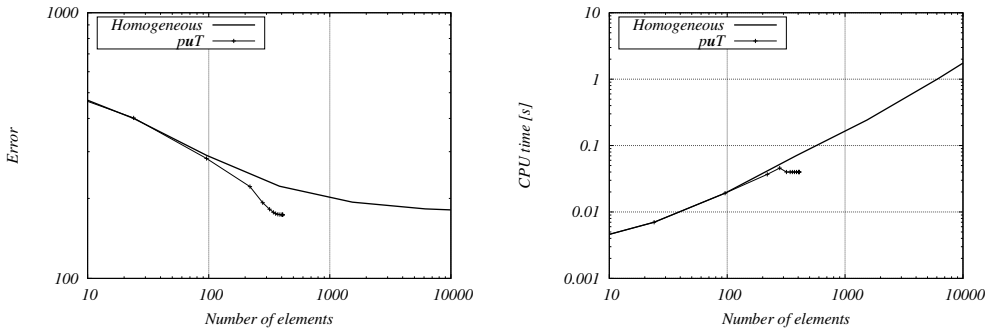


Figure 7.5: Singular exact solution results: exact error convergence measured with the scaled L^2-norm against the number of elements at the left, and computational time versus number of elements at the right.

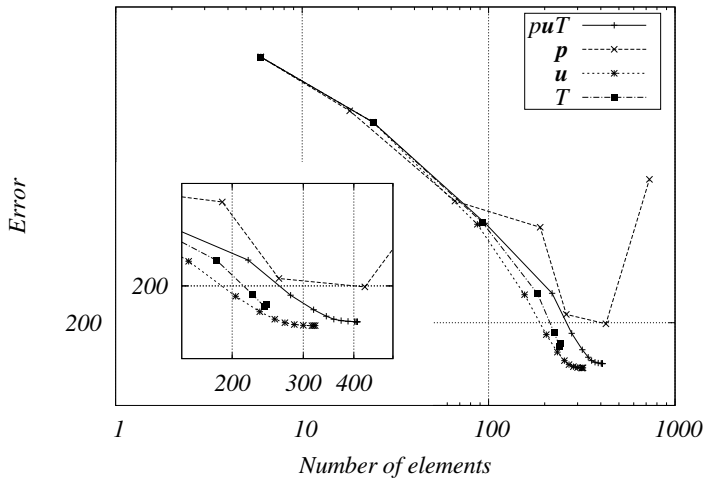


Figure 7.6: Singular exact solution results. Exact error convergence measured with the scaled L^2-norm against the number of elements for the separated subscales refinement.

and an impermeable condition for mass are set over the other walls. We set the Prandtl number to Pr $= 0.71$ and the Reynolds number to Re $= 1000$. At this conditions, the compressibility regime of the flow is M $= 1$.

The primitive variables formulation is used to solve this numerical example: the resulting flow is laminar and steady; it is obtained by running the simulation until the L^2-error norm between consecutive temporal results is below the transient converge criterion for all variables of the flow problem. The steady flow results are presented in Fig. 7.7, where the contours for pressure, velocity magnitude, and temperature, at four different cutting planes of the cavity, are presented. We observe that the laminar flow is comprised of a major vortical structure, with singular points of pressure near the top corners of the cavity.

We execute AMR simulations by fixing the error tolerance below 10^{-6} for the scaled L^2-norm, and 10^{-8} for the entropy measure; then we perform several consecutive refinements from an initial structured mesh composed of 35937 hexahedral elements. The final adapted meshes and the error estimation over these meshes are presented in Figures 7.8 and 7.9, for the scaled L^2- and entropy measures, respectively. The error estimation, which is below the prescribed tolerance in both cases, is plotted as contours at four different cutting planes of the cavity. We observe that the measured subscales-based error is smaller in the case of the entropy measure than the one with the scaled L^2-norm. Hence, the prescribed tolerance can be reduced when the error estimation is measured with the entropy measure; this may lead to a similar number of total elements, as in the case of the scaled L^2-norm measurement. Either description of the three-dimensional flow pattern given by the subscales-based error estimation is accurate: the resolution of three-dimensional flow singularities, including the description of the boundary layer near the top wall, and the singular points of pressure near the corners, are well identified. This is more evident in the case of the localized refinement that appears at the singular points near the edges and corners, and in the absence of refinement in the regions of the flow where stagnation occurs.

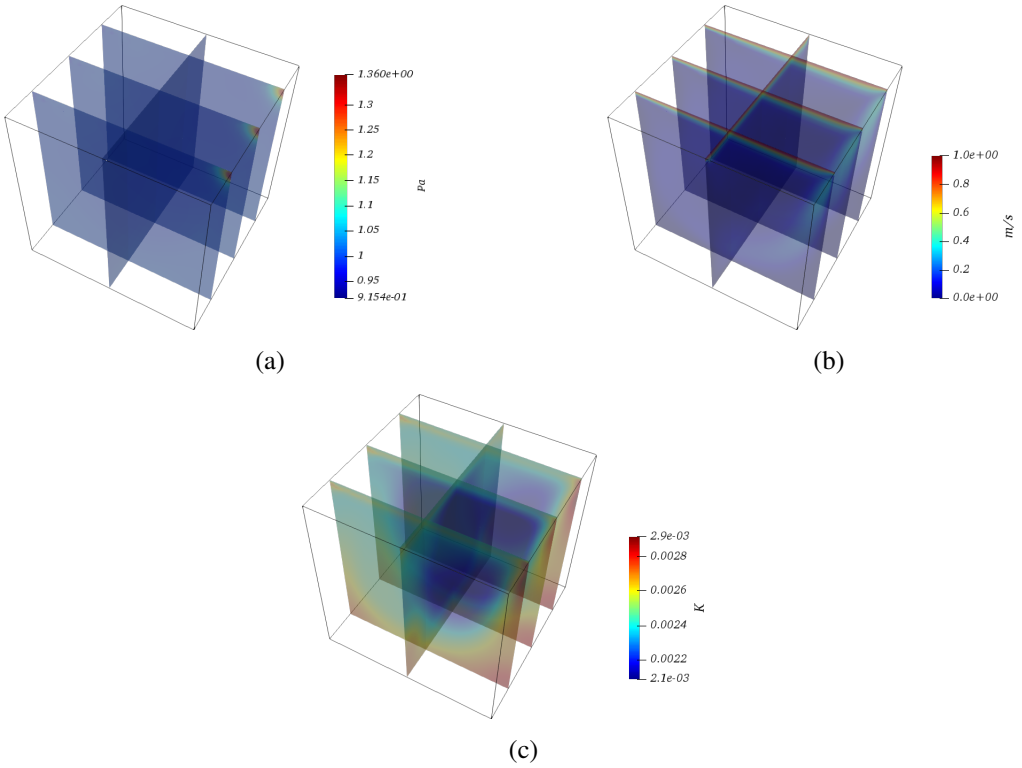


Figure 7.7: Three-dimensional lid-driven cavity results: (a) pressure, (b) velocity magnitude, and (c) temperature contours obtained using the refined mesh driven by the subscales-based error measured with the scaled L^2-norm.

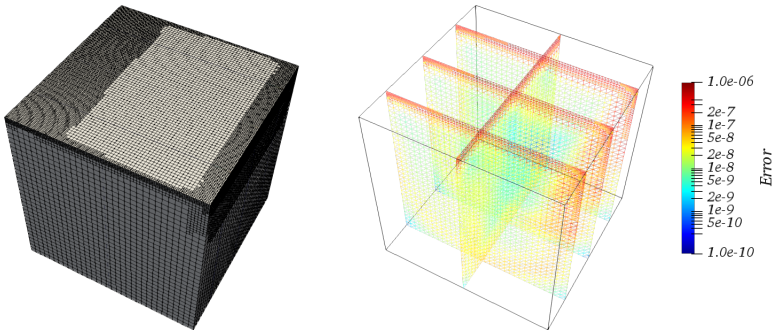


Figure 7.8: Three-dimensional lid-driven cavity results. Subscales-based error measured with the scaled L_2-norm: refined mesh composed by 86961 hexahedral elements on the left, and estimated error over four different cutting planes of the refined mesh on the right.

7.5.4 Differentially heated cavity

The fourth case that we study is the differentially heated cavity. In this problem we study the error estimation given by the particular design of the variational subscales, that is, we test the orthogonal, dynamic, and non-linear characteristics of the subscales in an unsteady

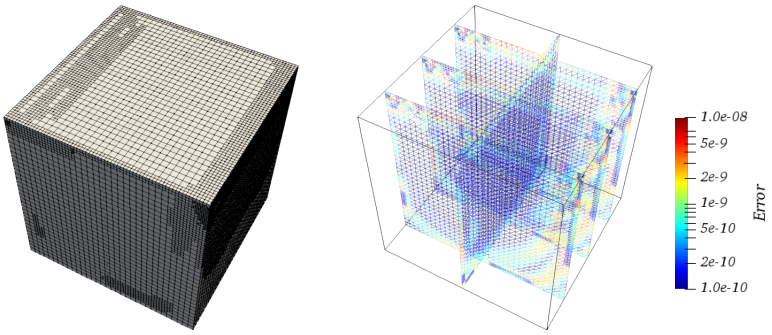


Figure 7.9: Three-dimensional lid-driven cavity results. Subscales-based error measured with the entropy measure: refined mesh composed by 83634 hexahedral elements on the left, and estimated error over four different cutting planes of the refined mesh on the right.

problem. The flow is considered as a two-dimensional flow confined inside a rectangular cavity $[0, L] \times [0, H]$ of aspect ratio $H/L = 8$, with $L = 1$ m. The temperature at the left (hot) wall is fixed to $T_H = 600$ K, and the temperature at the right (cold) wall to $T_C = 300$ K. No slip and impermeable conditions are set over the cavity walls, together with adiabatic boundary conditions for the upper and lower walls. Gravity is specified to be acting in the negative x_2 direction as $\boldsymbol{g} = (0, -9.8)$ m/s^2. The initial pressure, temperature, and density conditions for the fluid are 152525 Pa, 450 K, and 1.16 kg/m^3, respectively. The viscosity and thermal conductivity are set to $\mu = 2.5 \times 10^{-3}$ kg/(m s), and $\lambda = 3.55$ W/(m K), correspondingly. The non-dimensional Rayleigh number is $\text{Ra} = |\boldsymbol{g}|\theta\rho^2 c_p/(\mu\lambda) = 10^6$, where θ stands for the dimensionless temperature ratio $\theta = 2(T_H - T_C)/(T_H + T_C) = 0.66$. The simulation is run with a constant time step size of $\delta t = 10^{-2}$ s until the statistically stationary state (measured as the relative error between consecutive transient results of time-averaged variables) is reached.

In order to overcome the mechanical restriction of the pressure imposition for transient and variable flows at closed computational domains, an iterative penalization to the mass conservation equation, of the form $(q_h, \psi(p_h^{*i+1} - p_h^{*i}))$ at iteration $i + 1$, is included in the stabilized formulation. This penalization guarantees that p_h is solved correctly, up to a constant, when the relative value of pressure is not set at the computational boundary. The factor ψ is selected numerically as $\psi = 10^{-3}\rho/\mu$, in a way that it does not detriment neither the nonlinear scheme convergence (when ψ is large) nor the algebraic solver convergence (when $\psi \to 0$).

Moreover, we have observed that including the penalization to the mass conservation equation increases the non-linearity of the stabilized problem: it affects the dynamic subscales approximation since the transient term related to the pressure subscale in (7.37) is in fact divided by the pressure. Consequently, in this numerical example we modify this dependence by scaling the mass term of the dynamic operator in a different way, we use $(\rho^{n+1}h) \,/\, (\mu(|\boldsymbol{u}|^{n+1} + c^{n+1})\delta t)$ instead of $\rho^{n+1} / \,(p^{n+1}\delta t)$.

The transient character of the flow is firstly used for tracking the AMR simulations driven by the subscales-based error. We run the transient simulation using the primitive variables formulation and perform refinements both using the scaled L^2-norm and the entropy measure at the same two instants of time. The refinement tolerance is fixed to $0.1 < \eta_K^2 < 1$, in the case of the scaled L^2-norm, and to $10^{-14} < \eta_K^2 < 10^{-10}$, in the case of the entropy measure. The

refined meshes and the error distributions are presented in Fig. 7.10 corresponding to the same instants of time. The temperature contours, including some velocity streamlines, are shown at the left side of the figure. At the center, we display the refined meshes driven by the subscales-based error estimator measured with the scaled L^2-norm. The refined meshes driven by the subscales-based error and calculated with the entropy measure are plotted at the right side of the figure. It has been reported in Chapter 4 that the resolution of flow boundary layers and small perturbations of temperature (producing buoyancy) is enhanced with the inclusion of the dynamic subscales. We also observe that the error distribution, which is below the prescribed tolerance, and the mesh refinement are enhanced mostly at the lateral walls: the subscales-based error estimator is able to describe the boundary layer generated by the buoyancy of the flow. Moreover, the subscales-based error is able to track the relevant variations of the flow, so that, the mesh refinement is attached to the main flow structures through the simulation. It can also be seen that the coarsening of the mesh is carried out when laminar regions of the flow are found.

We can go deeper in the analysis of the transient behavior of the subscales as error estimator by performing calculations of the non-dimensional Nusselt number associated with this problem. Specifically, we investigate, in a qualitative manner, the influence of the subscales design in the estimation of the subscales-based error. For this, we calculate the Nusselt number, which relates the heat transferred from the hot to the cold wall, and which is calculated as

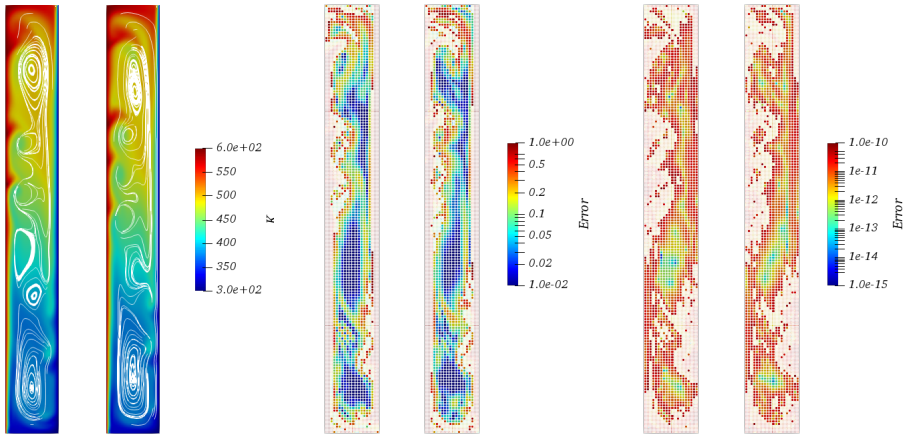
$$\text{Nusselt}(\boldsymbol{x},t) = \frac{L}{T_H - T_C} n_j \partial_j T(\boldsymbol{x},t),$$

over the hot wall of the cavity, and over a time window of 50 s after the statistically steady state is reached. In particular, the transient behavior of the Nusselt number can be evaluated by integrating the previous equation along the wall and averaging this result in time. We denote the integral result as $\overline{\text{Nusselt}}$, where $\overline{(\cdot)}$ stands for the discrete time average.

A reference value of $\overline{\text{Nusselt}}_{\text{ref}} = 52473.40$ is obtained by simulating the problem with an homogeneously refined mesh containing 2323 bi-quadratic elements and the subscales defined as quasi-static and residual-based because this method is similar to the Stream-line Upwind Petrov Galerkin (SUPG) method when linear elements are used, and the later has been widely tested in the literature in several compressible flow problems (as in [79]). In the reference simulation, we use the second order accurate BDF as the time integration scheme, and a constant time step size of $\delta t = 0.01$ s. The obtained reference Nusselt is used for qualitative comparisons: we calculate a (reference) L^2-error between the Nusselt obtained with the AMR simulations (driven by the subscales-based error) and the reference Nusselt. This reference error is calculated only for comparison reasons, and should not be confused as to be a goal-based error estimator.

We run AMR simulations using the subscales-based error estimator accounting for the different possibilities in the subscales design, and using both the scaled L^2- and entropy measures. We evaluate the inclusion of the dynamic subscales against the quasi-static subscales (the ones that neglect the temporal tracking of the subscales), and the orthogonal subscales against the residual subscales (which neglect the projection of the residual into the finite element space).

The time-averaged results of the calculated Nusselt number are presented in tables 7.1



(a) (b) (c)

Figure 7.10: Diferentially heated cavity results: (a) temperature contours with velocity streamlines, (b) scaled L^2-norm of the variational subscales: refinement tolerance is fixed $0.1 < \eta_K^2 < 1$, which led to meshes of 6795, and 6852 elements, and (c) entropy measure of the variational subscales: refinement tolerance is fixed to $10^{-14} < \eta_K^2 < 10^{-10}$, which led to meshes of 6853, and 7035 elements. Solution is obtained using the primitive formulation and it is presented for the same instants of time.

and 7.2. In those tables, we also present the error against the reference Nusselt, and the time-averaged subscales-based error estimation for the different subscales methods. In the case of the scaled L^2-norm, we find that the subscales-based error estimation matches correctly the reference error; the use of the scaled L^2-norm as a measure of the subscales-based error results in the accurate definition of the chaotic behavior, and the estimation of the error made by the subscales provides accurate approximations in comparison with the Nusselt reference error. We also observe that defining the subscales as orthogonal to the finite element space do not improves the approximation, and that, the dynamic definition of the subscales results in the most accurate description of the unsteady character of the flow made by the AMR simulations.

In the case of the subscales-based error measured with the entropy measure, the accuracy of the numerical approximation of the unsteady compressible flow problem matches the estimation of the Nusselt reference error, except when the orthogonal subscales are included. We also observe that the orthogonal subscales do not improve the numerical approximation of the AMR in contrast to the residual subscales, and that, the accuracy of the approximation is improved with the inclusion of the dynamic subscales and the residual definition of the subscales.

7.5.5 Supersonic flow over a flat plate

In this fifth case, we test a viscid supersonic example: the $M = 3$ flow over a flat plate problem. The problem domain is $[-0.1L, L] \times [-H/2, H/2]$, with $L = 1$ m, and $H = 0.25$ m. The inlet flow conditions are fixed over the left-most boundary $(-0.1, x_2)$ m as follows: a constant velocity prescription of $(3, 0)$ m/s, a constant density of 1 kg/m^3, and a constant temperature of

Table 7.1: Differentially heated cavity AMR simulations driven by the subscales-based error and measured with the scaled L^2-norm.

	Residual		Orthogonal	
	Quasi-static	Dynamic	Quasi-static	Dynamic
$\overline{\text{Nusselt}}$	52252.82	52391.22	52197.96	52249.35
$L^2-\text{error}(\overline{\text{Nusselt}})$	4.20×10^{-3}	1.56×10^{-3}	5.24×10^{-3}	4.26×10^{-3}
$\overline{\eta}$	105.17	104.87	104.54	103.43

Table 7.2: Differentially heated cavity AMR simulations driven by the subscales-based error and measured with the entropy measure.

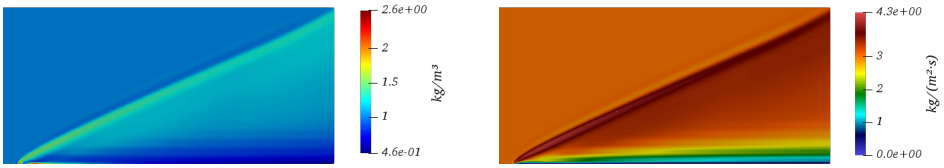
	Residual		Orthogonal	
	Quasi-static	Dynamic	Quasi-static	Dynamic
$\overline{\text{Nusselt}}$	51432.90	51501.42	51425.60	50497.68
$L^2-\text{error}(\overline{\text{Nuselt}})$	1.98×10^{-2}	1.85×10^{-2}	1.99×10^{-2}	3.76×10^{-2}
$\overline{\eta}$	9.46×10^{-4}	9.31×10^{-4}	1.047×10^{-3}	9.58×10^{-4}

0.00248 K. Zero flux conditions are imposed over the bottom boundary $(x_1 < 0, -H/2)$ m at the upstream. A no-slip condition for velocity, together with impermeable and adiabatic conditions, are specified for the plate surface $(x_1 > 0, -H/2)$ m. Over the top boundary $(x_1, H/2)$ m the following conditions are prescribed: zero normal stress, a fixed value of 3 m/s for the x_1-component of velocity, and a fixed density value of 1 kg/m^3. Lastly, free conditions are considered over the outflow wall, as the flow is supersonic. Viscosity and thermal conductivity are $\mu = 3 \times 10^{-6}$ kg/(m s) and $\lambda = 4.23 \times 10^{-3}$ W/(m K), respectively, defining a Reynolds number of Re $= 10^6$.

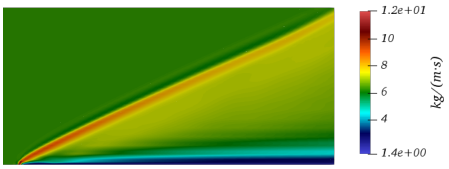
We run AMR simulations using the conservative variables formulation over a structured non-symmetric mesh composed of 11000 triangular P_1 elements, and achieve the mesh refinement with the subscales-based error until the transient convergence criterion is fulfilled. Figure 7.11 shows the density, momentum magnitude, and total energy results at the steady state. The flow is characterized by a thin boundary layer that separates from the plate, and by an oblique supersonic shock that is formed from the beginning of the plate to the outlet boundary. Results are presented for the refined mesh driven by the subscales-based error estimation measured with the scaled L^2-norm. This flow corresponds accurately to the referenced result in [82], so that, the approximation achieved by the subscales-driven AMR of the compressible flow solver is satisfactory.

In this example, we prescribe the error tolerance to $\eta_K < 10^{-8}$ for the scaled L^2-norm estimation, and to $\eta_K < 10^{-10}$ for the entropy measure estimation. The refined meshes driven by subscales-based error measured with the scaled L^2-norm and with the entropy measure, are presented in Figures 7.12 and 7.13, respectively. The estimated error, which is below the prescribed tolerance for both simulations, is also presented on the right side of these figures. We observe that the refined meshes are able to represent the characteristic flow pattern of the viscous compressible supersonic flow; the boundary layer near the plate surface, and the supersonic shock are both correctly determined by the subscales-based error estimator. Moreover,

the mesh is especially refined near the flow singularity of the initial viscous point. This ability to reproduce the local phenomena related to the supersonic flow is expressed with both types of measures of the subscales-based error estimator.



(a) (b)



(c)

Figure 7.11: Flow over a flat plate results. Density, momentum magnitude, and total energy contours obtained using the refined mesh driven by the subscales-based error measured with the scaled L^2-norm.

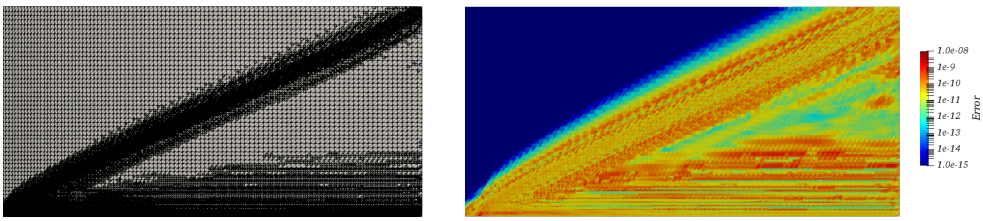


Figure 7.12: Flow over a flat plate results. Subscales-based error estimator measured with the scaled L_2-norm: refined mesh composed of 1558980 elements on the left, and estimated error over the refined mesh on the right.

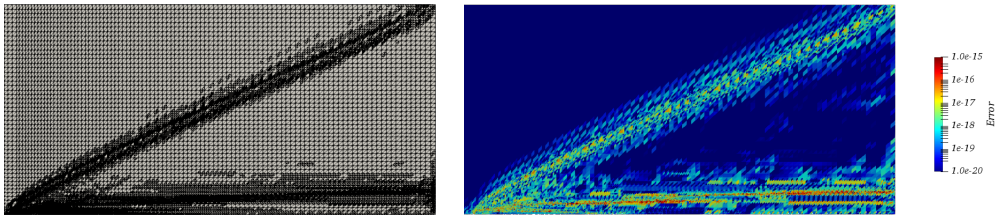


Figure 7.13: Flow over a flat plate results. Subscales-based error estimator measured with the entropy measure: refined mesh composed of 202806 elements on the left, and estimated error over the refined mesh on the right.

7.5.6 Supersonic flow past a cylinder

The last numerical example that we solve is the supersonic flow past a cylinder problem at Re $= 2000$ and M $= 2$. The cylinder is defined to be infinitely long in the axial direction and immersed in a compressible viscous flow that impinges it uniformly. The domain for this problem is typically defined as a rectangular domain. Instead, we define a curve-shaped domain with the cylinder located in the center, in which the inlet and outlet curved boundaries intersect. Boundary conditions are set as follows. The flow is injected from the left wall with a uniform and constant velocity of $(1, 0)$ m/s, a temperature of 6.14×10^{-4} K, and a density of 1 kg/m^3. On the cylinder surface, a no-slip condition for velocity and an adiabatic condition for energy is imposed. Free conditions are considered over the outflow wall (as for supersonic flows). The physical properties are set to $\mu = 0.0001$ kg/(m s) and $\lambda = 0.14338$ kJ/(m K s).

We depart from an initial unstructured mesh composed by 8141 P_1 elements and run AMR simulations using the conservative variables formulation together with the subscales-based error estimation, measured both with the L^2-norm and the entropy measure, until the convergence criteria for advancing in time is satisfied. Figure 7.14 shows the steady state results for the supersonic flow past a cylinder. The solution in this figure is the one obtained with the refined mesh driven by the subscales-based error estimator and measured with the scaled L^2-norm. We observe that the supersonic flow is composed by a strong shock at the upstream part of the cylinder, and by some oblique detached shock waves at the downstream part of the cylinder, as referenced in [73, 79]. The refinement gives an accurate resolution of the mesh at the thin shock layer of the supersonic expansion. It also gives correct results where gradients of the solution are not too sharp, such as for the weak tail shock that is formed in the wake structure.

Figures 7.15 and 7.16 display the comparison between the original unstructured mesh and the refined meshes driven by the error estimators measured with the scaled L^2-norm and with the entropy measure, respectively. In both AMR simulations the tolerance is fixed depending on the selected norm; in the case of the scaled L^2-norm we set the tolerance to $\eta_K < 10^{-5}$, and in the case of the entropy measure we fix it to $\eta_K < 10^{-8}$. The estimated error distribution, which is below the tolerance in both simulations, is also presented on the right side of those figures.

We observe that resolution of flow singularities and shocks, including the upstream supersonic shock, the description of boundary layers near the cylinder surface, and the wake structure of the flow, are described correctly by the refined mesh. In the case of the L^2-norm results, the subscales-based error estimation is taking effect mostly at the supersonic shock, and at the boundary layer near the cylinder surface. It can also be seen that the upstream supersonic shock structures (but also downstream) are greatly characterized by the use of this norm. It is worth to comment that the reference velocity of the scaled L^2-norm (7.42) designed to be of the order of $u_0 \approx |\boldsymbol{u}| + c$ gives accurate definitions of the error estimation. In the case of the subscales-based error estimator measured with the entropy measure, the refinement is acting homogeneously through the downstream part of the flow, so that, the resulting mesh is strongly refined at the wake structure behind the cylinder (even as much as for the supersonic shock).

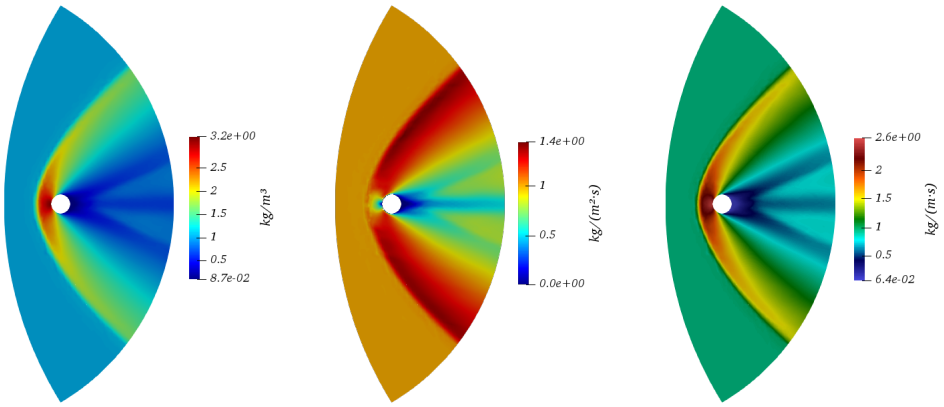


Figure 7.14: Supersonic flow past a cylinder results. Density, momentum magnitude, and total energy steady solution using the refined mesh driven by the subscales-based error estimator measured with the scaled L_2-norm.

7.6 Conclusions

In this chapter, we have used the variational subscales as an error estimator for the adaptive mesh refinement of compressible flow simulations. The estimator includes both the subscales at the element boundaries and in the interior of the elements. These subscales are defined as orthogonal, dynamic and non-linear.

Appropriate measures, namely, a scaled L^2-norm, and an entropy measure, have been used for composing the contribution of the subscales into a single error estimate.

The method has been tested in subsonic and supersonic compressible flow examples, both with the conservative variables formulation and with the primitive variables formulation. In all numerical examples, the local error has been measured using the subscales-based estimator, and the AMR has been performed leading to an equally distributed estimated error (below some prescribed tolerance). The error estimation given by the subscales has demonstrated to provide accurate information about the discretization error in an explicit fashion, this is,

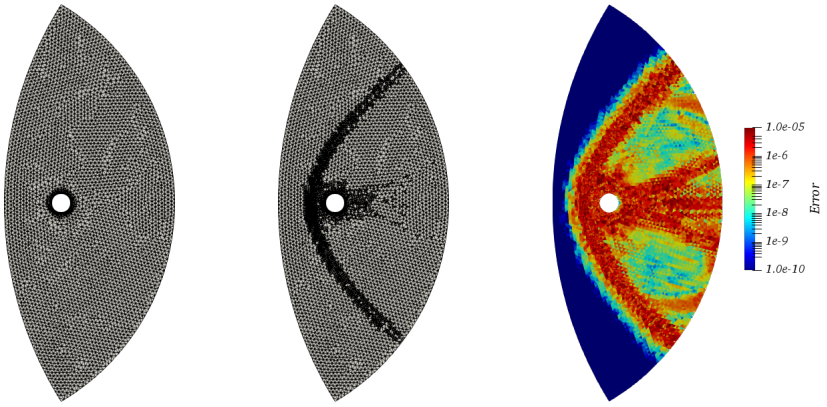


Figure 7.15: Supersonic flow past a cylinder results. Subscales-based error estimator measured with the scaled L_2-norm: initial unstructured mesh composed by 8141 P_1 elements on the left, refined mesh composed of 15613 P_1 elements on the center, and estimated error over the refined mesh on the right.

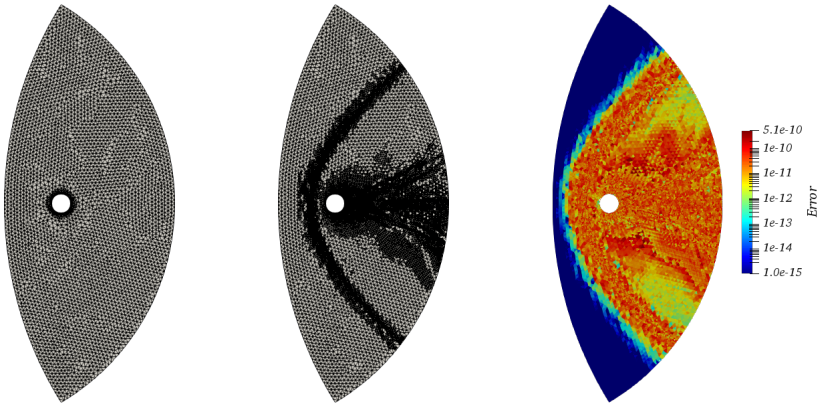


Figure 7.16: Supersonic flow past a cylinder results. Subscales-based error estimator measured with the entropy measure: initial unstructured mesh composed by 8141 P_1 elements on the left, refined mesh composed of 61020 P_1 elements on the center, and estimated error over the refined mesh on the right.

without having to estimate an overall error in the solution. This methodology has proven to give an efficient refined mesh with regard to the estimated error and the computational effort.

Chapter 8

Direct numerical simulation of the fricative [s] sound production

In this chapter, we simulate the flow and the acoustic waves generated inside the human vocal tract. In particular, we simulate the fricative [s] sound production. In order to directly solve the fluctuating scales of the compressible flow up to the sound frequencies, we approximate the full compressible Navier-Stokes equations (without the need of acoustic analogies) using a finite element stabilized formulation. In the first part of this chapter, we detail the numerical methods that we have implemented to deal with the challenges that arise in the aeroacoustic simulations. Among the most important challenges that we have encountered there are the numerical and physical instabilities that may occur due to wave reflections at the boundaries, and the need of fine meshes in order to represent the small fluctuating scales of the flow (sources of the acoustic propagation). Both are explained in detail. A realistic three-dimensional biomechanical replica of a vocal tract corresponding to the fricative sound [s] is used to simulate the sound generation. Clarifying information for such phenomena is developed based on the simulation.

8.1 Introduction

The sound production of the human voice is related to the flow of air passing through the vocal tract. The way the sound is generated in this particular configuration remains as an *open* question that still needs to be studied. One important argument for investigating this problem, for example, is the detailed illustration of the voice production mechanisms, which is required by speech therapies. The main issue on achieving an accurate description of sound is the complexity of the fluid mechanics involved: the compressible Navier-Stokes equations (together with the constitutive gas law) portray as the mathematical model that completely describes the fluid flow.

The description made by these equations possesses the advantage of representing the sound waves in the continuum limit, which is the propagation of very low amplitude displacements (in comparison to the main scales of the flow), but the main disadvantage is that the analytical solution of those equations is almost impossible for sound production applications. The only alternative is to numerically approximate the solution by means of numerical methods. Indeed,

the numerical approximation of the compressible Navier-Stokes equations represents an active research topic in computational mechanics, and the possibility to apply the numerical approximation of compressible flows onto a predictive model (that brings clues to the description of the human voice production) is emerging nowadays.

Ongoing numerical simulations intending to describe the sound production inside the vocal tract have been mainly dedicated to vowels and diphthongs, and mostly by using wave propagation equations (see for example the references [157–159]). In those simulations, the source of sound arising from the vocal chord vibration is propagated through the vocal tract, and then to the radiated field; sound mechanisms related to the fluid flow of air passing through the vocal tract are, therefore, not relevant. Instead, fricative sounds are another type of sounds which are driven in principle by the air flow impinging an obstacle (e.g. the teeth and the tongue), so that, the generation of the sound waves is tightly related to the description of the fluid flow. Actually, the production of fricative sounds arises distinctly from the pressure fluctuations of the turbulent airflow, as explained in [160].

The first descriptions of the turbulent flow related to the fricative sound production were analytic: several works have been devoted to predicting the flow vorticity, and hence, to study the propagation of the noise sources that arise from the vortical structures of the flow by applying the acoustic analogies concept introduced in [161]. The extensive literature on acoustic analogies applied to the analytical description of fricative sounds is not surveyed here, but we refer to the studies in [162, 163] for a comprehensive review of the numerical methodology involved. These works were able to match the experimental measurements of fricative sounds, and they indicated that the sound tends to be sensitive to the three-dimensional details of the vocal tract.

A more complex approach is the numerical solution of the compressible fluid equations up to the sound waves, named as the direct numerical simulation of sound. Direct numerical simulations (of the aerodynamic generation) of sound during phonation have been only attempted in [35, 164]. A two-dimensional higher order finite-difference method was used in those works to approximate the compressible flow inside a simplified human glottis and vocal tract. But the major drawback of finite-differences is the limitation to Cartesian meshes, and therefore, to simple geometries that differ from the actual mechanisms of the fricative sound production.

The preferred approach to simulate fricative sounds has been to solve the bulk scales of the flow with an incompressible flow solver and to model the acoustic propagation of sound waves through acoustic analogies. In this line, the numerical production of the fricative [s] sound has been recently studied by Ramsay and Shadle [165], Van Hirtum et. al. [166] and Cisonni et. al. [167], who tried to isolate the sound phenomena. Those authors investigated the influence of the aperture of a constriction formed by a teeth-shaped obstacle inside a channel by performing a Large Eddy Simulation (LES) of the incompressible flow field in a simplified domain. Nozaki et. al. [168] also achieved the LES simulation of the incompressible flow inside a realistic vocal tract geometry and studied the acoustic wave production. It was found that the turbulent jet description (made by the LES model) modifies the spectral characteristics of the generated sound. Recently, Pont et. al. [169] carried out a large-scale LES simulation of the incompressible flow in a realistic vocal tract geometry of fricative [s], and computed the generated sound by applying acoustic analogies. An accurate spectrum was found for locations near the mouth, and the separated contribution from the turbulent flow and the walls diffraction (e.g. the teeth) was also studied, as initially achieved in [170].

In the present work, we simulate the fricative [s] sound production by numerically approximating the compressible flow up to the sound frequencies at the morphological conditions of the air inside the human vocal tract. In this regard, we employ the term "direct" simulation in this chapter to imply that the solution of the compressible Navier-Stokes equations describes the propagation of sound waves and does not need an acoustic analogy model. This work may serve in understanding the relevant physics of sound production: due to the impossibility of *in situ* measurements of the airflow passing through the vocal tract, the direct numerical simulation of the fricative [s] voice production becomes an alternative option. As mentioned before, a direct numerical simulation of the human voice production inside the complex three-dimensional geometry of the human morphology has not previously been done: audible frequencies (related to the turbulent scales) have been difficult to reach using numerical simulation, and the process on how to design feasible sound generation simulations constitutes a necessary phase in the development of accurate models of voice production description.

This chapter is organized as follows. In Section 8.2, we describe the methodology that we use to numerically simulate the fricative sound [s]. Then, in Section 8.3 we present the numerical results. We develop a complete numerical study of the mechanisms that generate the sound inside the realistic geometry. Finally, some concluding aspects about the direct numerical simulation of fricative sound production problems are stated in Section 8.4.

8.2 Methodology

In this section, we present the process on how to achieve a feasible simulation of the fricative [s] sound generation. First, the realistic three-dimensional geometry that characterizes the vocal tract is presented. Then, the compressible flow equations and the detailed description of the strategies that we implement (to overcome the difficulties typically encountered for numerically simulating sound production) are presented. Finally, we describe how the numerical simulation is carried out.

8.2.1 Vocal tract model

An important issue of the simulation is the complex three-dimensional geometry of the human morphology. The main difficulty is to represent in a computational environment the complexity of the vocal tract shapes and tissues. The vocal tract shape that we implement in this work (and display partially in Fig. 8.1) is taken from [171]. This geometry was constructed in that previous work from medical imaging, specifically from a cone-beam CT scan (CB MercuRay, 512 slices of 512×512 pixels grid with accuracy 60.1mm), and has been previously used in [169] for the numerical simulation of fricative [s] sound using acoustic analogies. Moreover, the experiments using medical imaging that have ended in this geometry have also been designed setting the flow rates and the flow properties inside the mouth when the fricative [s] is articulated: the geometry corresponds to an adult male Japanese native speaker (in normal sitting position) uttering the phoneme [s] with a flow rate of 21 l/min.

In Fig. 8.1 we display the vocal tract from glottis to mouth, including the constricted passage between the tongue blade and the hard palate, the lower and upper incisors, and the lips. The face, and the propagation field are also displayed in the figure.

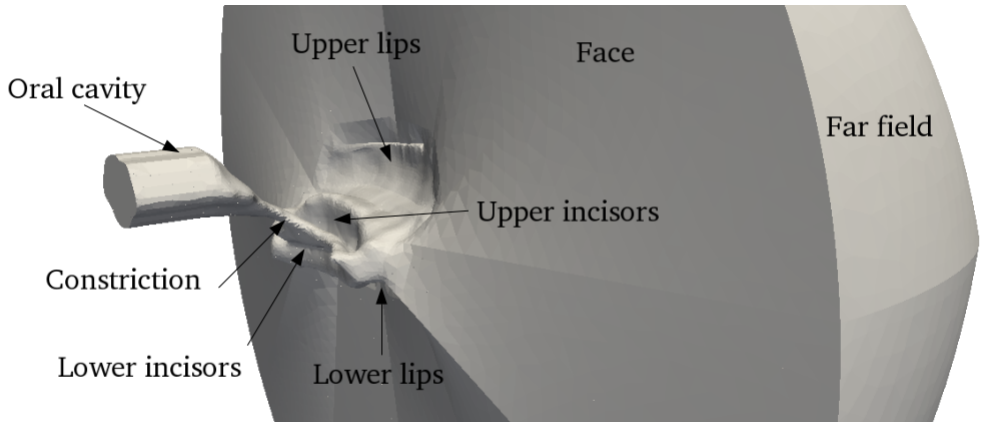


Figure 8.1: Vocal tract model.

8.2.2 Fluid flow model

The complex fluid mechanics related to sound production are completely described by the compressible Navier-Stokes equations. Considering a compressible, Newtonian, and viscous fluid, these equations are written in conservation form as:

$$\partial_t \rho + \partial_i \left(\rho u_i\right) = 0, \tag{8.1}$$

$$\partial_t \left(\rho u_i\right) + \partial_j \left(\rho u_j u_i + p\delta_{ij} - \tau_{ji}\right) = \rho f_i, \tag{8.2}$$

$$\partial_t \left(\rho \left(e + \frac{1}{2} u_i u_i\right)\right) + \partial_j \left(\rho u_j \left(h + \frac{1}{2} u_i u_i\right) - u_i \tau_{ij} + q_j\right) = \rho f_i u_i + \rho r, \tag{8.3}$$

together with appropriate boundary and initial conditions. Here ∂_t and ∂_j are short notations that indicate the Eulerian time derivative and $\partial/\partial x_j$, respectively. The usual summation convention is implied, with indices running from 1 to the number of dimensions d. In these equations ρ is the density, p is the pressure, $\boldsymbol{u}$ is the velocity, $\boldsymbol{\tau}$ is the viscous stress tensor, $\boldsymbol{f}$ is a body force vector, e is the internal energy, h is the enthalpy, $\boldsymbol{q}$ is the heat flux vector, r is a heat source/sink term and $\boldsymbol{I} = [\delta_{ij}]$ is the identity or *Kronecker* tensor. Supplementary constitutive relations are considered in order to close the mathematical model. For the viscous part of the stress tensor we use the relation for the Newtonian fluid $\tau_{ij}\left(\boldsymbol{u}\right) = \mu\left(\partial_j u_i + \partial_i u_j\right) - \frac{2\mu}{3}\left(\partial_l u_l\right)\delta_{ij}$, where μ is the viscosity. For the heat flux vector, we use Fourier's law, $q_i\left(\theta\right) = -\lambda \partial_i T$, where λ is the thermal conductivity and T is the temperature of the fluid. The caloric equation $e = c_v\left(T\right)T$ and the perfect gas state equation $p = \rho R T$ are used to calculate the pressure and the acoustic speed c. In these relations the specific heat at constant volume $c_v\left(T\right)$ and the specific heat at constant pressure $c_p\left(T\right)$ are thermodynamic properties of the fluid. We also define $R = c_p - c_v$ for the specific gas constant.

8.2.3 Non-reflecting conditions

Since the flow compressibility inside the vocal tract is typically near the incompressible range for the human voice production, classical boundary conditions for the equations (8.1) - (8.3)

cause artificial reflections. In particular, inlet and outlet boundaries cause an ingoing wave traveling through the domain. Artificial wave reflections not only interfere with the acoustic signal, but those can also produce numerical instabilities if the numerical method is not able to control the ingoing waves: the influence of spurious wave reflections over the sound source (namely, turbulent fluctuations of the flow) at nearly incompressible conditions is instantaneous.

Some methods that counteract wave reflections from the computational boundaries have emerged in the literature: the explicit damping of the equation, the addition of artificial convection velocities, and the solution of non-reflecting boundary conditions, are some of the most popular in the aeroacoustics field. Non-reflecting boundary conditions are our preferred approach to directly simulate sound at subsonic regimes. We implement the Poinsot and Lele non-reflecting conditions [105] by formulating the characteristic waves analysis traveling along the subsonic outlet boundaries introduced by Thompson in [104], and solving the local one-dimensional inviscid problem, where transverse, viscous and reaction terms of the compressible equations are neglected. A perfect an-echoic condition of the problem consists of modeling the in-going characteristic wave by using the linear relaxation method in [99]. Contrary to other approaches, we implement this condition because it guarantees that the problem is well-posed, and in particular, it implies the conservation of physical quantities (exhibited by the hyperbolic nature of the system) when the in-going wave is modeled. Particularly, we enforce the implicit solution of the non-reflecting boundary conditions by introducing some penalization terms to the compressible formulation (as done in Chapter 4).

We also enforce the total annihilation of wave reflections by damping the compressible Navier-Stokes equations at the external-most part of the computational domain. This is a robust approach that filters the solution only in the local region called buffer zone, where the extra-term $\boldsymbol{\sigma}(\boldsymbol{U} - \boldsymbol{U}_\infty)$ is added to the right-hand-side of the compressible flow equations. Here $\boldsymbol{U}$ is the vector of the problem variables, and $\boldsymbol{U}_\infty$ is the vector of unperturbed values at the far field. For convenience, let us denote as $\text{diag}(\boldsymbol{e})$ the diagonal $(d+2) \times (d+2)$ matrix with vector $\boldsymbol{e}$ on the diagonal. The damping matrix $\boldsymbol{\sigma}$ is defined as a diagonal matrix of the form $\boldsymbol{\sigma} = (\sigma_0/\delta t)\ \text{diag}\,(\rho/p, \rho\boldsymbol{I}, \rho c_p)$, where σ_0 is a damping coefficient, and δt is the discrete time step size. The computational effort that is required to solve the buffer zones is large, and consequently, we restrict the damping zone using the damping coefficient σ that is calculated to vary smoothly from a constant value at the boundaries to zero at the inner-most part of the buffer zone.

8.2.4 Numerical strategy

The compressible Navier-Stokes equations together with the non-reflecting conditions presented before, portray as the mathematical model that is used to describe the propagation of acoustic waves traveling through the air flow. These equations are so complex that it is impossible to obtain an analytical solution for the transient character of the flow involved in aeroacoustics. Numerical methods remain to be the practical approach when one aims to approximate the solution of these equations.

8.2.4.1 Finite element approximation

In this work, we adopt the Variational Multi-Scale (VMS) finite element formulation of the compressible equations that has been presented in Chapter 3 and extended to the solution of aeracoustic simulations at low compressibility regimes in Chapter 4. This stabilized finite element method combines the possibility of describing complex geometries with high order interpolations, and it is accurate in low Mach number flows. The design of the stabilization method leads to an accurate description of transient compressible flows. In particular, including the orthogonal, time-dependent, and non-linear definition of the variational sub-grid scales improves the accuracy of the finite element approximation in contrast to other stabilized formulations.

Because the acoustic scales are much smaller than the bulk scales, the simulation of the sound generated by the vocal tract is involved with the correct description of the non-linear interaction between the very different flow scales. The numerical approximation of all the relevant scales of the bulk flow is typically referred as Direct Numerical Simulation. Resolving only the largest bulk scales of the flow, and modeling the smallest scales is the object of Reynolds Average Navier Stokes (RANS) and Large Eddy Simulation (LES), which are some of the most prevailing turbulence models in the computational fluid mechanics community. Applying RANS models in the case of aeroacoustic flows may filter the aeroacoustic frequencies of the flow (as discussed in [121, 172] for aeroacoustic simulations using acoustic analogies), especially those related to the high range frequencies.

In this sense, no turbulence model is applied in the present simulation. Instead, the accurate definition of the sound waves relies only on the numerical diffusion given the VMS method, without any modification of the continuous problem, nor the inclusion of any turbulence model for the sub-grid scales. This kind of approximation has been recently related to the Implicit LES (ILES) methods (for the Burgers equation in Chapter 2, or in [46, 53] for incompressible turbulent flows), which accurately represent the underlying turbulent behavior by the addition of purely dissipative numerical terms given by the VMS method without any modification of the continuous problem, even if the mesh is not too fine to resolve the majority of the flow scales.

8.2.4.2 Aeroacoustic inefficiency

Performing the direct numerical simulation of sound by numerically approximating the compressible Navier-Stokes equations arises several challenges. One prevailing challenge is that the small spatial and temporal scales (that act as the main acoustic sources of noise) coexist with the large propagation distances of the radiated sound. Attempting to resolve the full aeroacoustic problem of sound production (with the adequate description of the radiated sound) demands a high amount of computational resources: the solution of small spatial scales inside a large computational domain leads to a discretized system of equations that contains a large number of unknowns. Solving this large linear system is computationally very expensive.

The main computational strategy that we adopt (to overcome this situation) is a nodally-based parallelization strategy that decomposes the computational domain so that, the required computational effort can be distributed into several processors working simultaneously.

Moreover, the propagation of the smallest spatial scales leads to long simulation run-times to describe the complete frequency spectrum of sound. In particular, at low Mach numbers the acoustic speed is very high, and consequently, the number of operations required by the

compressible solver to describe the propagation of sound is very demanding. The stability restriction placed by the Courant number for explicit time marching solvers limits the maximum time step size of the method. In the case of implicit solvers, the number of operations that the linear solver has to complete is large: the discrete linear system is very ill-conditioned, and therefore, the solution convergence for iterative solvers is low.

To overcome the inefficiency that arises from the aeroacoustic propagation in the low Mach number limit we include some other developments in the formulation; the use of an implicit time integration scheme and the convenient scaling of the compressible problem that allows the use of iterative linear system solvers, are two of the most crucial requirements. Concretely, we implement an implicit second order Backward Differentiation Formula (BDF) for integrating the fluid flow equations in time that allows us to define the time step size of the simulation, and a Picard's scheme for dealing with the nonlinear character of the compressible equations. Also, a scaling of the compressible problem is performed by splitting the primitive unknowns into a relative and a reference part (that is fixed).

However, the main numerical ingredient to deal with a large number of unknowns arising from the description of small scales is the adoption of an Adaptive Mesh Refinement (AMR) method. The AMR method has the objective of optimizing the computational effort by dynamically re-configuring an initial mesh during the simulation execution. The AMR involves two main steps: first, the decision of which elements to modify (mainly the ones contributing the most to the global solution error), and then, the adaptation of those selected elements.

In the case of the first issue, we adopt an explicit a-posteriori error estimator in order to choose the elements that are modified. The methodology that we implement has been presented in Chapter 7 and uses the variational sub-grid scales to estimate the error in each element; the estimator includes both the subscales at the element boundaries and in the interior of the elements, which are defined as orthogonal, time-dependent and non-linear. An scaled L^2-norm is used for composing the contribution of the subscales associated to the pressure, velocity, and temperature variables into a single estimate in each element.

With respect to the adaptation of the selected elements, the refinement process is held locally in each distributed processor. The method for refining the mesh is based on the $h-$refinement algorithm for computational physics meshes within a distributed memory parallel setting that has been described in Chapter 6 and implemented in the `RefficientLib` software. We use the AMR method as follows. The simulation begins with a given ($0-$level) coarse grid, and then it subsequently performs refinement (or coarsening). The local error is estimated with the subscales-based estimator after some time steps of the simulation, and this estimation is used to refine or coarsen the elements of the mesh depending on a given tolerance criteria. The simulation advances in time and adapts the mesh until a temporal convergence criterion is satisfied.

The suitability of the developed numerical techniques for large computational problems have been addressed in previous chapters of the thesis, including its execution in high-performance computing environments. This AMR strategy for compressible flows leads to an equally distributed estimated error in the computational domain (below some prescribed tolerance), incurring in a low runtime fraction for the refinement process compared to the solution time of each time step.

8.2.5 Numerical simulation

In the following paragraphs, we describe the application of those methods in the direct numerical simulation of the fricative [s] sound production.

8.2.5.1 Initial and boundary conditions

Let us first comment how the initial and boundary conditions are applied to the computational model. Provided the flow rates that were used to construct the fricative geometry model, we implement these as initial and boundary conditions.

The initial conditions are considered as follows: the air flow is supposed to be homogeneously at rest, with density, pressure, and temperature values of $\rho = 1.2$ kg/m^3, $p = 101300$ Pa, and $T = 283.88$ K, respectively. Moreover, the air is considered as an ideal gas, with ratio of specific heats $\gamma = 1.4$ and physical properties $c_p = 1.010$ kJ/(kg K) and $c_v = 0.718$ kJ/(kg K). The viscosity is fixed to $\mu = 1.81 \times 10^{-5}$ kg/(m s), and the thermal conductivity to $\lambda = 2.57 \times 10^{-2}$ W/(m K), mimicking the air physical properties at atmospheric conditions. These values give a sound speed of $c = 343$ m/s and a Prandtl number of Pr $= 0.71$.

A no-slip wall condition for the momentum equation, together with an impermeable condition for mass, and an adiabatic condition for total energy are specified for all the boundaries except for the inlet and outlet boundaries. The inlet flow conditions are set to mimic the *in vivo* conditions by imposing an uniform velocity profile $(2.4, 0, 0)$ m/s, together with a temperature value of 283.88 K. Since a uniform velocity profile is set at the channel entrance, the entrance channel ensures that a parabolic velocity profile is fully developed at the constriction. This inlet flow condition gives a Mach number of M ~ 0.007, which is nearly the incompressible regime. The Reynolds number measured with respect to the inlet diameter of the channel gives Re $= 8850$.

The pressure at the inlet and outlet boundaries, together with the flow variables at the outlet boundary, are calculated using the non-reflecting boundary conditions described in the first part of this section. In this sense, the weak penalty, the pressure relaxation, and the characteristic length coefficients for the non-reflecting boundary conditions (see Chapter 4 and references therein for a complete description about these parameters) are set to $\eta_0 = 0.01$, $\sigma = 100$, and $l = 0.1$ m, respectively.

8.2.5.2 Spatial discretization

The initial spatial discretization is carried out using tetrahedral four-node finite elements. We depart from an initial unstructured mesh composed by 9532418 tetrahedral elements (as the $0-$level coarsest grid) and run AMR simulations using the subscales-based error estimation until the convergence criteria for advancing in time is satisfied. The refinement tolerance is fixed to $10^{-4} < \eta_K^2 < 10^{-3}$, so that, the adaptive meshes vary from the initial number of elements to $\sim 19 \times 10^6$ total elements at most.

8.2.5.3 Temporal description

As we implement the second order implicit BDF scheme for advancing in time, the time step size can be set to the constant value of 10^{-6} seconds through the simulation. This time step

size is higher than the one of the Courant-Friederich-Lewi condition, which is less than 10^{-8} s. At least three non-linear iterations are performed by the solution method at each time step. The solution of the linear system is carried out by the iterative stabilized bi-conjugate gradient method implemented in the PETSC library [114].

Nevertheless, the very different propagation scales of the flow arise a bad condition of the linear system of equations that is very difficult to solve: the speed of sound propagation (in this almost incompressible regime) differs from the velocity scales of the flow. Due to this bad condition of the linear system of equations, most of the calculation time per step of time is consumed by the linear solver: for a 1 millisecond of simulation time (with a total number of 1000 time steps) the overall computational time gives more than 24576 computational hours.

Different types of preconditioners have been evaluated, such as the additive Schwartz method and the Point block Jacobi preconditioner, but they have not been able to improve the rate of convergence of the linear solver. We find that using an additive Schwarz method and a block ILU preconditioning [115], slightly improves the convergence and the numerical accuracy of the linear solver. The computational cost of this 1 ms simulation is therefore very large, and it becomes a must to perform the calculations in a supercomputer. But the cost of the simulation remains restrictive in order to complete a realistic description of the sound generation process in the human voice (close to 1 s of simulation time).

The numerical computation of this problem is performed at the Marconi supercomputer [1]. For this case, the resulting computing time per time step is less than 75 seconds using 1024 CPUs, and the simulation is completed in less than 24 wall clock hours.

This time integration setting has been proved to give the best performance regarding the available computational hours in the supercomputer infrastructure. As commented before, the most expensive part of the solution is related to the solution of the linear system of equations, being more than the 95% of the cost per time step. In the case of the adaptive refinement strategy, this has demonstrated to be negligible as it represents less than the 2% of the computational cost per time step.

8.3 Results

The simulation results are presented in this section. We first investigate the performance of the adaptive mesh refinement simulation. Then, we study the acoustic sources around the constriction and the sound wave propagation to the far field, including the frequency analysis of the sound.

In order to demonstrate the ability of the aeroacoustic flow solver to simulate the sources of sound, we first track the mesh adaptation, and particularly, we plot the estimated error over the refined meshes. This is presented in Fig. 8.2, where the original mesh, the refined mesh, and the estimated error over the refined mesh are presented. The refined mesh and the estimated error, which is below the prescribed tolerance, are displayed over one half of the computational domain at a certain instant of the simulation. As described in the previous section, the refinement of the mesh is enforced in the turbulent regions of the flow since the acoustic sources have to be well resolved: smaller element sizes are necessary near the channel constriction and

[1] Lenovo NeXtScale 1.512 nodes (each node has 2 $\times$ 18-cores Intel Xeon E5-2697 v4 Broadwell at 2.30 GHz, and 128 GB of main memory connected via Infiniband)

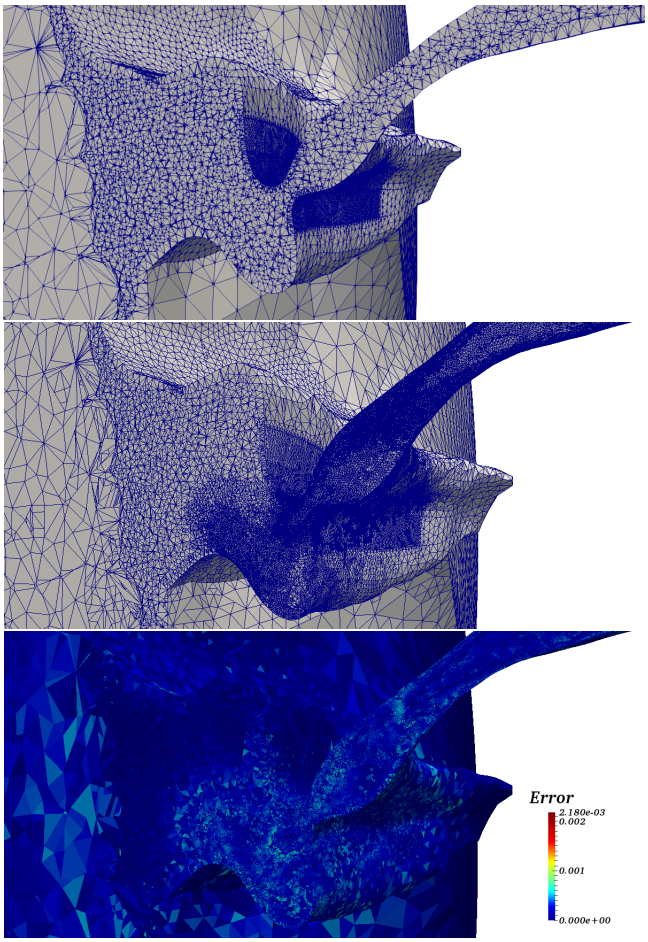


Figure 8.2: Fricative model results: original mesh at the top, refined mesh at the middle, and error estimation at the bottom.

inside the channel in order to properly describe the turbulence phenomena of the air jet. In this sense, we observe that the refinement is indeed able to localize the compressible effects inside the mouth, as well as boundary layers near the walls, and the turbulent jet downstream the constriction. As commented before, the adaptive meshes vary from the initial number of elements to $\sim 19 \times 10^6$ total elements at most. This is remarkable, since the previous description of the turbulent flow in [169] was achieved using a homogeneously refined mesh composed of $\sim 45 \times 10^6$ elements.

We evaluate the velocity field by plotting some contours in the middle-plane cut: Fig. 8.3 displays the time-averaged velocity magnitude and the root mean square velocity distribution at the $(x_1, x_2, 0)$ plane. Both the time-averaged and the root mean square distributions of velocity give insights about laminar and turbulent regions in the flow. These results demonstrate the formation of some back-flow turbulent structures within the constriction (between the tongue and the upper incisors), as well as the turbulent jet that separates from the constricted region to the lips. Apart from the jet formation, we observe that the turbulent core of the jet diminishes as it propagates downstream: the jet collides with the lower lip, and then it forms some characteristic back-flow structures which are specific to the vocal tract configuration of the teeth and

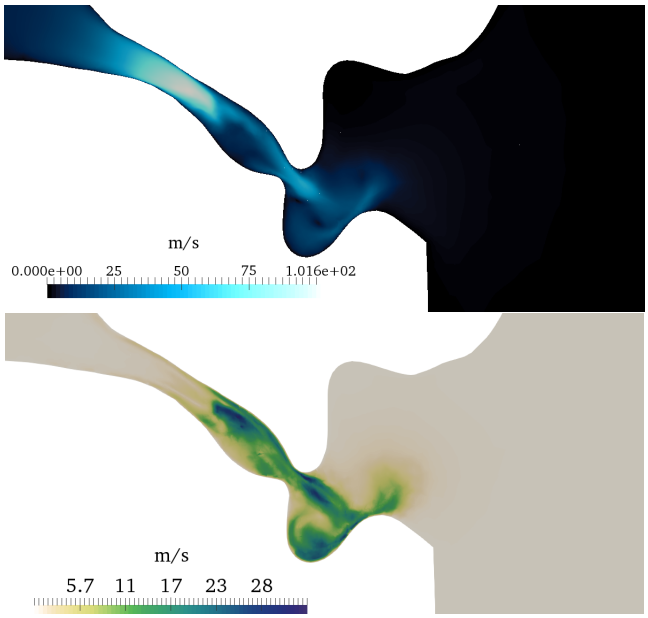


Figure 8.3: Fricative model results: time-averaged velocity magnitude (at the top), and root mean square velocity magnitude at a cutting middle plane of the constriction.

lips. This characteristic flow pattern has been also obtained by Pont et. al. [169] in the case of the incompressible flow simulation, and has been identified as the prevalent turbulent structure in this flow setting.

Another look at the constriction region that is shown in Fig. 8.4, where instantaneous results for velocity magnitude, relative pressure, and relative temperature are displayed at the middle-plane cut, demonstrates that the turbulent jet formation and its collision with the lower incisor is the main cause for the pressure fluctuations, hence, for the sound wave generation. In contrast, it can be observed that the fluctuations produced by the turbulent flow inside constriction only cause a pressure drop due to the flow acceleration at the corners of the incisors (both the upper and lower incisors), but weakly affects the propagation of waves in the downstream region. The temperature, on the other hand, slightly increases at the constriction channel and at the jet collision, but it is not relevant in the wave generation process.

The instantaneous field of pressure at the middle-plane cut, presented in Fig. 8.4, exposes the regions of the turbulent jet which are representative in the production of high fluctuations of pressure. We distinguish some particular regions of the flow related to the fluctuating pressure: upstream and constricted region, incisors to lips, and lips out to the free field. As commented before, the pressure differences along the jet and the evolution of those fluctuations downstream at the collision can be observed as a high contribution in the noise generation. Great differences in pressure are observed between the incisors and the lips, which generate a considerable scattering interaction inside the vocal tract.

Figure 8.5 plots the pressure contours for the middle-plane cut at two different instants of the developed flow. We observe that the three-dimensional features of the sound wave propagation in the far-field is correctly described in the simulation. The scattered waves inside the vocal tract are propagated apart from the teeth and lips to the surrounding downstream flow.

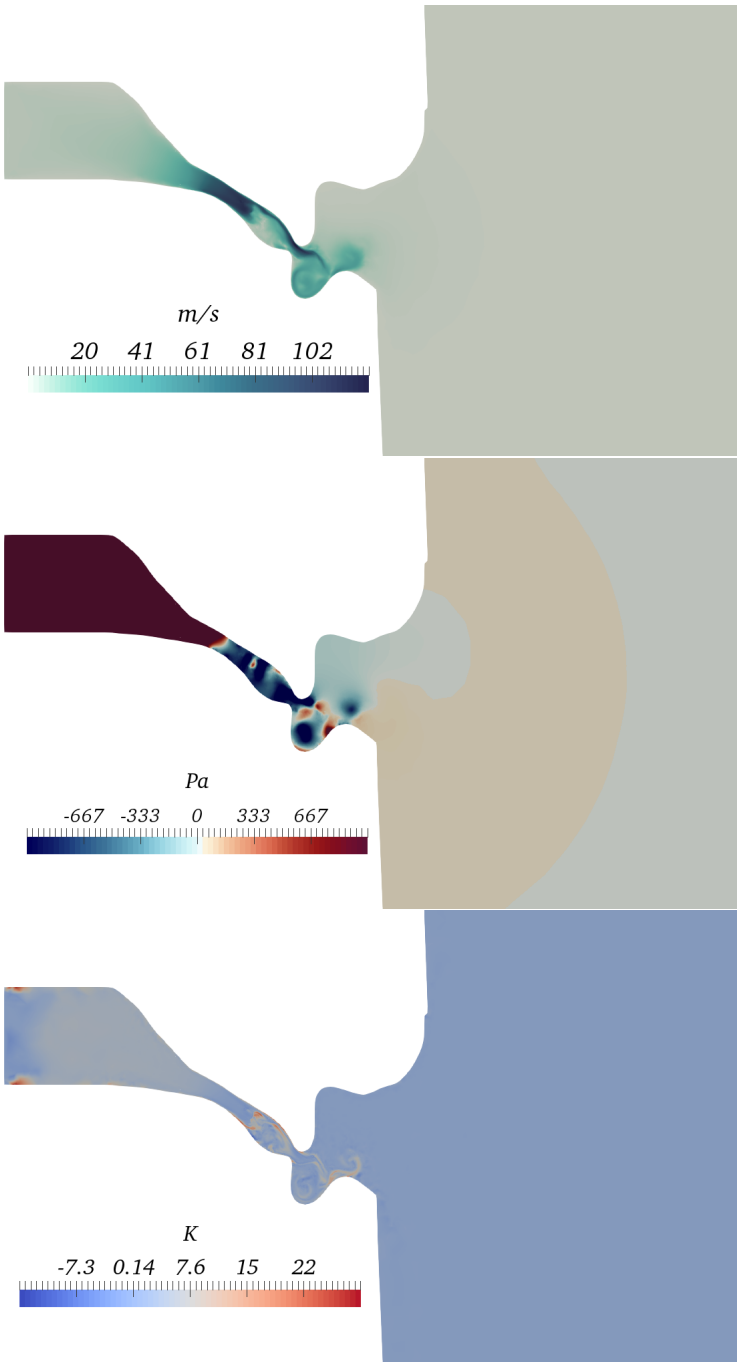


Figure 8.4: Fricative model results: velocity magnitude, relative pressure, and relative temperature closeup at a cutting middle plane of the constriction.

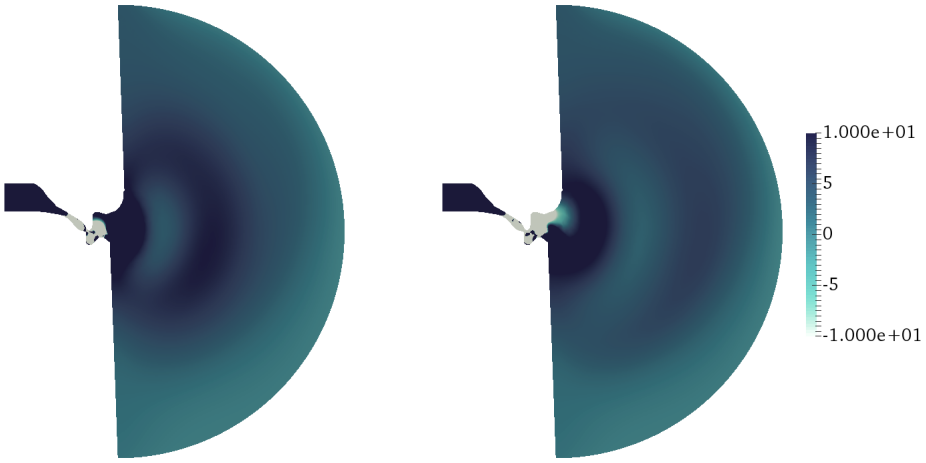


Figure 8.5: Fricative model results: relative pressure contour magnitude at the cutting middle plane for two different instants of the simulation.

No reflections that may produce both physical and numerical instabilities occur at the outlet boundaries. In this sense, the accuracy of the non reflecting schemes has been crucial to assure that waves reflections do not produce further disturbances to the sound wave.

Finally, in order to characterize the production of sound, we plot the spectrum of the time history of the velocity magnitude at $(0, 0.0025, 0)$ in Fig. 8.6. The transformation from the time domain to the frequency domain at the given point is calculated by using a Fast Fourier Transform (FFT). Spectrum results show the description of the audible frequencies that belong to the range $[20, 20000]$ Hz, but also the reproduction of higher frequencies of the flow.

8.4 Conclusions

In this work we have simulated the fricative [s] sound production. We have presented the numerical model that allows solving the propagation of sound waves through a realistic replica of the human vocal tract. Some particular features of the proposed numerical model are the finite element formulation that is able to represent the complex realistic geometry, the non-reflecting conditions that mitigate the scattering of radiated waves, and the adaptive mesh refinement during the simulation execution.

A realistic three-dimensional bio-mechanical replica of a vocal tract corresponding to the fricative sound [s] has been used to simulate the sound generation. The level of refinement near the channel constriction given by the adaptive method has demonstrated to capture the acoustic frequencies of sound, and therefore to produce computational savings in comparison with homogeneous element size meshes. The numerical model has been able to represent the mechanisms that produce the sound inside the vocal tract, specifically, the mesh refinement of the constricted region has allowed to capture the turbulent flow scales that cannot be described with coarser homogeneous meshes.

Fluctuations in pressure near the channel constriction lead to the formation of acoustic

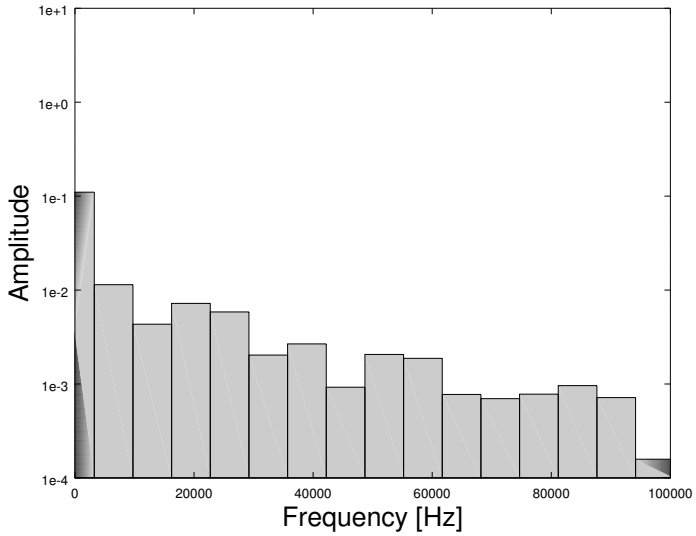


Figure 8.6: Fricative model results: single-sided amplitude spectrum of the velocity magnitude at point $(0, 0.0025, 0)$.

pressure waves which propagate to the far field. Spectrum results demonstrate the description of audible frequencies using the proposed implicit method.

Nevertheless, the cost of the simulation is high and remains restrictive in order to complete a realistic description of the sound generation process in the human voice. The present work is an initial possibility that we hope clarifies the search for affordable numerical approximations to this type of complex aeroacoustic applications. A future work can be related to the implementation of a fractional step method for compressible formulations (e.g. the one in [91]) as the temporal integration scheme of the VMS stabilized compressible formulation, which is expected to relieve the computational effort of the flow solver related to monolithic time integration schemes, especially in the case of low Mach number aeroacoustics simulations. In that case, further investigations have to be completed to evince the accurate reproduction of the [s] sound spectrum given by the simulated pressure fluctuations, specifically, additional simulations that may verify the published spectrum in [169], which has been obtained using a broader 11 ms signal, have to be completed. Another possibility can be to use *in vitro* experiments (e.g. the constructed with synthetic geometries that represent part of the vocal tract) to validate the proposed numerical model.

Chapter 9

Conclusions

9.1 Achievements

The main objective of this thesis has been to investigate a stabilized finite element formulation of the compressible Navier-Stokes equations able to represent high-frequency fluctuations of the fluid flow, and it has been addressed in several progressive parts.

The first part of the thesis has been devoted to the study of the Variational Multi-Scale (VMS) stabilized formulations for the compressible fluid flow equations, and has been opened in Chapter 2 with the numerical approximation of the one-dimensional Burgers equation. The stabilized formulation has been proposed by means of the Orthogonal Sub-Grid Scales - Variational Multi-Scale (OSGS-VMS) method in the Fourier space, which has allowed to clarify the scale dependence of the numerical diffusion introduced by the VMS method. The interaction between the subscales and the resolved scales when the former are defined to be orthogonal to the finite dimensional space has been clarified: an accurate approximation to the turbulence energy spectra has been obtained with the addition of the purely dissipative numerical terms given by the OSGS-VMS method without any modification of the continuous problem (e.g. as in the case of Large Eddy Simulation models).

The design of the VMS stabilized formulation has been the main objective of the first part of the thesis, specifically, the approximations made for the VMS formulation like the OSGS method, the non-linear tracking of the subscales, and their time evolution, have been extensively studied for the compressible Navier-Stokes stabilized formulations.

Firstly, the VMS formulation of the compressible Navier-Stokes equations written in conservative variables has been presented in Chapter 3. A systematic way to design the matrix of algorithmic parameters from the perspective of a Fourier analysis has been demonstrated, so that, the correct description of compressible flows given by the stabilized method has been addressed. Also, the solution of supersonic shocks with the inclusion of shock capturing methods has been exposed. In this sense, it has been demonstrated the improved solvability of the non-linearity introduced by the anisotropic shock capturing method. The artificial shock capturing method has been completed with the calculation of the added numerical diffusivity using the orthogonal projection onto the finite element space of the gradient of the solution, instead of the common residual definition.

Secondly, Chapter 4 has been focused on the development of the finite element solver for complex aeroacoustic flows; this chapter has presented the selected approaches for over-

coming most of the numerical challenges that arise in the aeroacoustic simulations at low compressibility regimes. The VMS approximation of the compressible Navier-Stokes equations written in primitive variables has been used in this regard. The definition of the static matrix of stabilization parameters in terms of a modified velocity that accounts for the local compressibility, together with the dynamic stabilization matrix design for the time-dependent subscales have been developed for this formulation. The possibility of directly computing the acoustic pressure waves at low Mach numbers with this formulation has been validated: accurate simulations of acoustic radiation have been obtained by solving directly the compressible Navier-Stokes equations and damping wave reflections at the computational boundaries with the inclusion of the weak imposition of implicit non-reflecting boundary conditions in the stabilized formulation.

Remarkably, numerical experiments with subsonic flows have demonstrated that including the temporal derivatives of the subscales, and defining the space where the subscales live as the orthogonal to the finite element space, improve the accuracy of the variational method both in the conservative variables and in the primitive variables formulations.

As a culmination of the first part of the thesis, Chapter 5 has been devoted to the formulation of a numerical approach that ensures the global conservation of mass, momentum, and total energy over the primitive variables solution of the compressible Navier-Stokes equations through the solution of a small optimization problem that uses Lagrange multipliers. Several numerical tests have led to the conclusion that the present methodology actually makes the global correction of the physical quantities, but that this global correction is not enough to improve the primitive variables formulation accuracy in the case of supersonic shocks.

The second part of this thesis has been dedicated to making affordable the solution of the smallest fluctuating scales of flow. To this end, the development of a novel algorithm for $h-$refinement in computational physics meshes in a distributed memory parallel setting, its implementation and the solution of some AMR examples in supercomputers, have been presented in Chapter 6. The algorithm has been capable of dealing with non-balanced hierarchical refinement, where multi refinement level jumps are possible between neighbor elements, giving good weak scalability results for meshes of up to two thousand million elements over 2000 CPUs clusters.

A further step has been to apply the developed refinement algorithm in the compressible flow problems with the development of a local error estimate of the compressible flow solver. Chapter 7 has been devoted to the definition of an explicit a-posteriori error estimator that can be used in AMR simulations of the compressible Navier-Stokes equations. The proposed methodology has employed the VMS framework, and specifically, the idea has been to use the variational subscales to estimate the error. The estimator has included both the subscales at the element boundaries and in the interior of the elements, which are defined as orthogonal, time-dependent and non-linear. Appropriate norms, namely, an scaled L^2-norm and an entropy norm, have been used for composing the contribution of the subscales into a single explicit error estimate, which has demonstrated to give an efficient refined mesh (with regard to the estimated error and the computational effort), and more importantly, without having to solve additional differential equations to compute the estimate.

At that point, the possibility to directly compute the turbulent compressible airflow and the generated acoustic waves (without the need of an analogy model) in a complex aeroacoustic flow has been granted: the AMR driven by the subscales-based error estimation has led to an

equally distributed estimated error (below some prescribed tolerance), and the runtime fraction for the refinement process has been seen to be reduced when compared to the runtime for solving linear systems of equations on the generated meshes. Therefore, Chapter 8 covered the application of the previously described developments in the direct numerical solution of the fricative [s] sound production inside a realistic bio-mechanical geometry replica of the human vocal tract. The suitability of the developed numerical techniques, together with the proposed refinement algorithm for this large computational problem has been tested in a high-performance computing environment. Numerical results have demonstrated the ability of the numerical methods in reproducing the smallest scales of the flow (related to the audible sound). Refined meshes have been able to represent the small fluctuating scales of the turbulent flow near the constriction, thus, the generated acoustic frequencies have been accurately described and propagated to the far field. Nevertheless, the computational cost of the simulation is high and remains restrictive in order to complete a realistic description of the sound generation process in the human voice at low Mach numbers.

9.2 Further research

The development of the thesis has brought several ideas that can be addressed in the future.

- One first idea is the extension of the OSGS-VMS formulation of Chapter 2 in order to solve more general problems, especially the Burgers equation in three-dimensional periodic cubes, which is expected to be straightforward. In addition, the work in that chapter serves as a first approach to the definition of the subscales as both L^2-orthogonal and H^1- orthogonal to the finite element space. Therefore, an expected future work will be to include the H^1- orthogonal subscales to the finite element space in several flow problems. For this goal, the application of this concept to the incompressible Navier-Stokes equations may be the first step, with the consequent hope that the implicit turbulence modeling will be more accurate. Likewise, the energy budget between finite element scales and sub-grid scales involves the same transfer of energy from the resolved scales to the sub-grid scales as vice versa, so that, the numerical methodology agrees exceptionally. Then, this idea can be extended to the VMS approximation of the compressible Navier-Stokes equations.

- Another task will be the culmination of the global conservation restrictions idea to the compressible flow formulation based on primitive variables. The main objective, in this case, will be to overcome the problems encountered in the numerical tests by modifying the present formulation. For this goal, the introduction of a scaling matrix $\boldsymbol{S}$ is one idea to follow. The scaling matrix may lead to dimensionally-consistent measurements, so that, the functional can be written in terms of a scaled L^2-norm of the type $\|\boldsymbol{U}\|_{\boldsymbol{S}}^2 = \int_\Omega (\boldsymbol{U}^\top \boldsymbol{S}\boldsymbol{U})\, d\Omega$. This leads to the possibility of coupling the conservative variables inside the minimization functional, so that, the Lagrangian functional may account simultaneously for the complete physical quantities. Completing this work will also lead to the submission in a peer-reviewed scientific journal of the article: C. Bayona, A. Pont, J. Baiges, and R. Codina, "Global conservation restrictions of the compressible Navier-Stokes equations written in primitive variables", arising from Chapter 5.

- An important work will be the implementation of the fractional step method for compressible formulations (e.g. the one in [91]) as the temporal integration scheme of the VMS stabilized compressible formulation, which is expected to relieve the computational effort of the flow solver, especially in the case of low Mach number aeroacoustics simulations.

- Also, the formulation of the anisotropic shock capturing method in system form may be another research to be done: the calculation of the VMS stabilization diffusion along the streamline can be achieved in tensor form as $\boldsymbol{A}_j^\top \boldsymbol{\tau} \boldsymbol{A}_k$ and then considered in the streamline tensor with (3.72).

- Non-reflecting boundary conditions for the compressible Navier-Stokes equations have to be deeply investigated, so that, a more direct approach (than the weak imposition) for dealing with spurious wave reflections in the case of implicit finite element compressible flow solvers may emerge.

- In the case of the $h-$refinement algorithm `RefficientLib`: it currently deals with $h-$refinement, but the extension of the algorithm to $h-p-$refinement will definitely be a matter of future work. Several approaches can be evaluated to locally increase the degree of the polynomial that is used to construct the elemental shape functions. One of the preferred approaches is to restrict the $p-$refinement to the use hierarchical shape functions, in which refining p to $p+1$ consists of adding extra terms to the actual functions. Nonetheless, other options may be evaluated for consistency with the current refinement algorithm, and since we will favor the integration of the proposed refinement library with existing computational physics codes.

- Several ideas follow the application of the variational subscales as error estimators. Including them as estimators for the coarsement of meshes in the hyper-reduction approach for Reduced Order Models (ROM) developed in our research group is one of the straightforward duties: in that framework, the discrete problem is solved in a coarser mesh than the one used in the Full-Order/ROM approach. The idea is to construct this coarse mesh by coarsening the original mesh with the use of the AMR method driven by the subscales-based estimator, which can be defined in terms of the variational subscales of the physical problem, but also by including the calculation of certain ROM subscales.

- Finally, the exploitation of the aeroacoustics solver that has been developed in this thesis into several other complex aeroacoustics problems emerges as an opportunity, specially if those are in the subsonic range. The objective at this point is to extend the achievements made in the fricative [s] sound production. First, to perform a broader simulation (at least 10 ms) at a feasible computational cost, so that wider comparisons against published spectra can be completed. Then, the simulation of other sounds at the actual *in-vivo* conditions of the vocal tract can be pursued. For example, the more challenging [z] fricative that exhibits important acoustic feedback inside the mouth, or the possibility of simulating syllables within an Arbitrary Lagrangian Eulerian framework, both may lead to a better study of the aeroacoustic problem of the human voice production.

Bibliography

[1] J. M. Burgers, *The nonlinear diffusion equation: asymptotic solutions and statistical problems*. Springer Science & Business Media, 2013.

[2] E. Hopf, "The partial differential equation $u_t + (uu)_x = \mu u_{xx}$," *Communications on Pure and Applied Mathematics*, vol. 3, no. 3, pp. 201–230, 1950.

[3] J. D. Cole, "On a quasi-linear parabolic equation occurring in aerodynamics," *Quarterly of applied mathematics*, vol. 9, no. 3, pp. 225–236, 1951.

[4] A. Majda, *Compressible fluid flow and systems of conservation laws in several space variables*. Springer Science & Business Media, 2012, vol. 53.

[5] J. T. Beale, T. Kato, and A. Majda, "Remarks on the breakdown of smooth solutions for the 3-D Euler equations," *Communications in Mathematical Physics*, vol. 94, no. 1, pp. 61–66, 1984.

[6] P. Constantin and C. Foias, *Navier-Stokes equations*. University of Chicago Press, 1988.

[7] S. Klainerman and A. Majda, "Singular limits of quasilinear hyperbolic systems with large parameters and the incompressible limit of compressible fluids," *Communications on pure and applied Mathematics*, vol. 34, no. 4, pp. 481–524, 1981.

[8] ——, "Compressible and incompressible fluids," *Communications on Pure and Applied Mathematics*, vol. 35, no. 5, pp. 629–651, 1982.

[9] H. B. Da Veiga, "An l^p-theory for the $n-$dimensional, stationary, compressible Navier-Stokes equations, and the incompressible limit for compressible fluids. The equilibrium solutions," *Communications in Mathematical Physics*, vol. 109, no. 2, pp. 229–248, 1987.

[10] P.-L. Lions and N. Masmoudi, "Incompressible limit for a viscous compressible fluid," *Journal de mathématiques pures et appliquées*, vol. 77, no. 6, pp. 585–627, 1998.

[11] J. Principe and R. Codina, "Mathematical models for thermally coupled low speed flows," in *Advances in Theoretical and Applied Mechanics 2009*, Citeseer.

[12] A. Matsumura and T. Nishida, "Initial boundary value problems for the equations of motion of compressible viscous and heat-conductive fluids," *Communications in Mathematical Physics*, vol. 89, no. 4, pp. 445–464, 1983.

[13] D. Hoff, "Discontinuous solutions of the Navier-Stokes equations for multidimensional flows of heat-conducting fluids," *Archive for Rational Mechanics and Analysis*, vol. 139, no. 4, pp. 303–354, 1997.

[14] B. Desjardins and C. Lin, "A survey of the compressible Navier-Stokes equations," *Taiwanese Journal of Mathematics*, vol. 3, no. 2, pp. 123–137, 1999.

[15] R. Codina, S. Badia, J. Baiges, and J. Principe, "Variational multiscale methods in computational fluid dynamics," in *Encyclopedia of computational mechanics*, E. Stein, R. de Borst, and T. J. R. Hughes, Eds., Wiley Online Library.

[16] T. J. R. Hughes and M. Mallet, "A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems," *Comput. Methods. Appl. Mech. & Eng.*, vol. 58, no. 3, pp. 305–328, 1986.

[17] T. J. R. Hughes, "Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods," *Comput. Methods. Appl. Mech. & Eng.*, vol. 127, no. 1, pp. 387–401, 1995.

[18] M. Woopen, A. Balan, G. May, and J. Schütz, "A comparison of hybridized and standard DG methods for target-based hp-adaptive simulation of compressible flow," *Computers & Fluids*, vol. 98, pp. 3–16, 2014.

[19] J. T. Erwin, W. K. Anderson, S. Kapadia, and L. Wang, "Three-dimensional stabilized finite elements for compressible Navier-Stokes," *AIAA journal*, vol. 51, no. 6, pp. 1404–1419, 2013.

[20] T. Tu and D. O'Hallaron, "Balance refinement of massive linear octrees," Technical Report CMU-CS-04-129, Carnegie Mellon School of Computer Science, Tech. Rep., 2004.

[21] H. Sundar, R. S. Sampath, and G. Biros, "Bottom-up construction and 2:1 balance refinement of linear octrees in parallel," *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2675–2708, 2008.

[22] T. Isaac, C. Burstedde, and O. Ghattas, "Low-cost parallel algorithms for 2:1 octree balance," in *Proceedings of the 26th IEEE International Parallel & Distributed Processing Symposium*, 2012.

[23] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "LibMesh: A C++ library for parallel adaptive mesh refinement/coarsening simulations," *Engineering with Computers*, vol. 22, no. 3-4, pp. 237–254, 2006.

[24] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler, "Algorithms and data structures for massively parallel generic adaptive finite element codes," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 2, p. 14, 2011.

[25] C. Burstedde, L. C. Wilcox, and O. Ghattas, "`P4EST`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011. DOI: `10.1137/100791634`.

[26] P. Devloo, J. T. Oden, and P. Pattani, "An hp adaptive finite element method for the numerical simulation of compressible flow," *Comput. Methods. Appl. Mech. & Eng.*, vol. 70, no. 2, pp. 203–235, 1988.

[27] R. Hartmann, "Adaptive discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 51, no. 9-10, pp. 1131–1156, 2006.

[28] K. J. Fidkowski and D. L. Darmofal, "A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 225, no. 2, pp. 1653–1672, 2007.

[29] M. Kouhi, E. Oñate, and D. Mavriplis, "Adjoint-based adaptive finite element method for the compressible Euler equations using finite calculus," *Aerospace Science and Technology*, vol. 46, pp. 422–435, 2015.

[30] B. R. Ahrabi, W. K. Anderson, and J. C. Newman, "An adjoint-based hp-adaptive stabilized finite-element method with shock capturing for turbulent flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 318, pp. 1030–1065, 2017.

[31] R. Glasby, N. Burgess, K. Anderson, L. Wang, S. Allmaras, and D. Mavriplis, "Comparison of SU/PG and DG finite-element techniques for the compressible Navier-Stokes equations on anisotropic unstructured meshes," in *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013, p. 691.

[32] B. Reza Ahrabi, W. K. Anderson, and J. C. Newman, "High-order finite-element method and dynamic adaptation for two-dimensional laminar and turbulent Navier-Stokes," in *32nd AIAA Applied Aerodynamics Conference*, 2014, p. 2983.

[33] P. Capon and P. Jimack, "An adaptive finite element method for the compressible Navier-Stokes equations," *Numerical methods for fluid dynamics*, vol. 5, pp. 327–334, 1995.

[34] S. Lele, "Compact finite difference schemes with spectral-like resolution," *Journal of Computational Physics*, vol. 103, no. 1, pp. 16–42, 1992.

[35] W. Zhao, C. Zhang, S. Frankel, and L. Mongeau, "Computational aeroacoustics of phonation, Part I: Computational methods and sound generation mechanisms," *The Journal of the Acoustical Society of America*, vol. 112, no. 5, pp. 2134–2146, 2002.

[36] J. M. Burgers, *The nonlinear diffusion equation*. D.Reydel publishing company, Boston, 1974.

[37] K. S. Surana, S. Allu, J. Reddy, and P. Tenpas, "Least-squares finite element processes in h, p, k mathematical and computational framework for a non-linear conservation law," *International Journal for Numerical Methods in Fluids*, vol. 57, no. 10, pp. 1545–1568, 2008.

[38] J. Smagorinsky, "General circulation experiments with the primitive equations: I. The basic experiment," *Monthly weather review*, vol. 91, no. 3, pp. 99–164, 1963.

[39] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, "A dynamic subgrid-scale eddy viscosity model," *Physics of Fluids A: Fluid Dynamics (1989-1993)*, vol. 3, no. 7, pp. 1760–1765, 1991.

[40] P. J. Mason and D. Thomson, "Stochastic backscatter in large-eddy simulations of boundary layers," *Journal of Fluid Mechanics*, vol. 242, pp. 51–78, 1992.

[41] B. J. Geurts and D. D. Holm, "Regularization modeling for large-eddy simulation," *Physics of Fluids (1994-present)*, vol. 15, no. 1, pp. L13–L16, 2003.

[42] A. Das and R. D. Moser, "Optimal large-eddy simulation of forced Burgers equation," *Physics of Fluids*, vol. 14, no. 12, pp. 4344–4351, 2002.

[43] R. Codina, "Stabilization of incompressibility and convection through orthogonal subscales in finite element methods," *Comput. Methods. Appl. Mech. & Eng.*, vol. 190, no. 13, pp. 1579–1599, 2000.

[44] R. Codina, "Stabilized finite element approximation of transient incompressible flows using orthogonal subscales," *Comput. Methods. Appl. Mech. & Eng.*, vol. 191, no. 39, pp. 4295–4321, 2002.

[45] T. J. R. Hughes, V. Calo, and G. Scovazzi, "Variational and multiscale methods in turbulence," in *Mechanics of the 21st Century*, Springer, 2005, pp. 153–163.

[46] R. Codina, J. Principe, O. Guasch, and S. Badia, "Time dependent subscales in the stabilized finite element approximation of incompressible flow problems," *Comput. Methods. Appl. Mech. & Eng.*, vol. 196, no. 21, pp. 2413–2430, 2007.

[47] O. Guasch and R. Codina, "Statistical behavior of the orthogonal subgrid scale stabilization terms in the finite element large eddy simulation of turbulent flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 261, pp. 154–166, 2013.

[48] C. E. Wasberg, T. Gjesdal, B. A. P. Reif, and Ø. Andreassen, "Variational multiscale turbulence modelling in a high order spectral element method," *Journal of Computational Physics*, vol. 228, no. 19, pp. 7333–7356, 2009.

[49] V. Gravemeier and W. A. Wall, "An algebraic variational multiscale–multigrid method for large-eddy simulation of turbulent variable-density flow at low Mach number," *Journal of Computational Physics*, vol. 229, no. 17, pp. 6047–6070, 2010.

[50] J. Hoffman and C. Johnson, "A new approach to computational turbulence modeling," *Comput. Methods. Appl. Mech. & Eng.*, vol. 195, no. 23, pp. 2865–2880, 2006.

[51] Y Bazilevs, V. Calo, J. Cottrell, T. J. R. Hughes, A Reali, and G Scovazzi, "Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 197, no. 1, pp. 173–201, 2007.

[52] J. Liu and A. Oberai, "The residual-based variational multiscale formulation for the large eddy simulation of compressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 245, pp. 176–193, 2012.

[53] O. Colomés, S. Badia, R. Codina, and J. Principe, "Assessment of variational multiscale models for the large eddy simulation of turbulent incompressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 285, pp. 32–63, 2015.

[54] D. Forti and L. Dedè, "Semi-implicit BDF time discretization of the Navier–Stokes equations with VMS-LES modeling in a high performance computing framework," *Computers & Fluids*, vol. 117, pp. 168–182, 2015.

[55] Z. Wang and A. Oberai, "Spectral analysis of the dissipation of the residual-based variational multiscale method," *Comput. Methods. Appl. Mech. & Eng.*, vol. 199, no. 13, pp. 810–818, 2010.

[56] Y. Li and Z. Wang, "A priori and a posteriori evaluations of sub-grid scale models for the Burgers' equation," *Computers & Fluids*, vol. 139, pp. 92–104, 2016.

[57] R. Codina, "A stabilized finite element method for generalized stationary incompressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 190, no. 20, pp. 2681–2706, 2001.

[58] A. K. Kuczaj, B. J. Geurts, and W. D. McComb, "Nonlocal modulation of the energy cascade in broadband-forced turbulence," *Physical Review E*, vol. 74, no. 1, p. 016 306, 2006.

[59] G. Wei and Y. Gu, "Conjugate filter approach for solving Burgers equation," *Journal of Computational and Applied Mathematics*, vol. 149, no. 2, pp. 439–456, 2002.

[60] O. Métais and M. Lesieur, "Spectral large-eddy simulation of isotropic and stably stratified turbulence," *Journal of Fluid Mechanics*, vol. 239, pp. 157–194, 1992.

[61] T. Tezduyar and T. J. R. Hughes, "Finite element formulations for convection dominated flows with particular emphasis on the compressible Euler equations," in *Proceedings of AIAA 21st aerospace sciences meeting, AIAA Paper*, vol. 83, 1983, p. 0125.

[62] A. Brooks and T. J. R. Hughes, "Streamline Upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," in *Proceedings of FENOMEC '81*, 1981.

[63] M. Polner, *Galerkin least-squares stabilization operators for the Navier-Stokes equations: a unified approach*. Enschede, Netherlands: University of Twente, 2005.

[64] M. Billaud, G. Gallice, and B. Nkonga, "Stabilized finite element method for compressible–incompressible diphasic flows," in *Numerical Mathematics and Advanced Applications 2009*, Springer, 2010, pp. 171–179.

[65] R. Sevilla, O. Hassan, and K. Morgan, "An analysis of the performance of a high-order stabilised finite element method for simulating compressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 253, pp. 15–27, 2013.

[66] F. Shakib, "Finite element analysis of the compressible Euler and Navier-Stokes equations," PhD thesis, 1989.

[67] G. Hauke and T. J. R. Hughes, "A comparative study of different sets of variables for solving compressible and incompressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 153, no. 1, pp. 1–44, 1998.

[68] G. Hauke, A. Landaberea, I. Garmendia, and J. Canales, "A segregated method for compressible flow computation. Part II: General divariant compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 49, no. 2, pp. 183–209, 2005.

[69] B. Koobus and C. Farhat, "A variational multiscale method for the large eddy simulation of compressible turbulent flows on unstructured meshes—-application to vortex shedding," *Comput. Methods. Appl. Mech. & Eng.*, vol. 193, no. 15, pp. 1367–1383, 2004.

[70] F. Van Der Bos, J. Van Der Vegt, and B. Geurts, "A multi-scale formulation for compressible turbulent flows suitable for general variational discretization techniques," *Comput. Methods. Appl. Mech. & Eng.*, vol. 196, no. 29, pp. 2863–2875, 2007.

[71] V. Levasseur, P. Sagaut, F. Chalot, and A. Davroux, "An entropy-variable-based VMS/GLS method for the simulation of compressible flows on unstructured grids," *Comput. Methods. Appl. Mech. & Eng.*, vol. 195, no. 9, pp. 1154–1179, 2006.

[72] W. Dahmen, T. Gotzen, S. Müller, and R. Schäfer, *Adaptive multiresolution Finite Volume Discretization of the Variational Multiscale Method: General Framework*. Inst. für Geometrie und Praktische Mathematik, 2011.

[73] F. Rispoli and R. Saavedra, "A stabilized finite element method based on SGS models for compressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 196, no. 1, pp. 652–664, 2006.

[74] S. Marras, M. Moragues, M. Vázquez, O. Jorba, and G. Houzeaux, "Simulations of moist convection by a variational multiscale stabilized finite element method," *Journal of Computational Physics*, vol. 252, pp. 195–218, 2013.

[75] F. Rispoli, A. Corsini, and T. Tezduyar, "Finite element computation of turbulent flows with the discontinuity-capturing directional dissipation (DCDD)," *Computers & fluids*, vol. 36, no. 1, pp. 121–126, 2007.

[76] R. Codina, "Finite element approximation of the convection-diffusion equation: Subgrid-scale spaces, local instabilities and anisotropic space-time discretizations," in *BAIL 2010-Boundary and Interior Layers, Computational and Asymptotic Methods*, Springer, 2011, pp. 85–97.

[77] T. J. R. Hughes and M. Mallet, "A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems," *Comput. Methods. Appl. Mech. & Eng.*, vol. 58, no. 3, pp. 305–328, 1986.

[78] G. Le Beau and T. Tezduyar, *Finite element computation of compressible flows with the SUPG formulation*. Army High Performance Computing Research Center, 1991.

[79] S. Mittal and T. Tezduyar, "A unified finite element formulation for compressible and incompressible flows using augmented conservation variables," *Comput. Methods. Appl. Mech. & Eng.*, vol. 161, no. 3-4, pp. 229–243, 1998.

[80] S. Mittal, "Finite element computation of unsteady viscous compressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 157, no. 1, pp. 151–175, 1998.

[81] T. J. R. Hughes, G. Scovazzi, and T. Tezduyar, "Stabilized methods for compressible flows," *Journal of Scientific Computing*, vol. 43, no. 3, pp. 343–368, 2010.

[82] V. Kotteda and S. Mittal, "Stabilized finite element computation of compressible flow with linear and quadratic interpolation functions," *International Journal for Numerical Methods in Fluids*, vol. 75, no. 4, pp. 273–294, 2014.

[83] T. Tezduyar and M. Senga, "Stabilization and shock-capturing parameters in SUPG formulation of compressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 195, no. 13, pp. 1621–1632, 2006.

[84] R. Codina, J. M. González-Ondina, G. Díaz-Hernández, and J. Principe, "Finite element approximation of the modified Boussinesq equations using a stabilized formulation," *International Journal for Numerical Methods in Fluids*, vol. 57, no. 9, pp. 1249–1268, 2008.

[85] R. Codina, "Finite element approximation of the three-field formulation of the Stokes problem using arbitrary interpolations," *SIAM Journal on Numerical Analysis*, vol. 47, no. 1, pp. 699–718, 2009.

[86] ——, "A discontinuity-capturing crosswind-dissipation for the finite element solution of the convection-diffusion equation," *Comput. Methods. Appl. Mech. & Eng.*, vol. 110, no. 3, pp. 325–342, 1993.

[87] S. Albensoeder and H. C. Kuhlmann, "Accurate three-dimensional lid-driven cavity flow," *Journal of Computational Physics*, vol. 206, no. 2, pp. 536–558, 2005.

[88] J. Wong, D. Darmofal, and J. Peraire, "The solution of the compressible Euler equations at low Mach numbers using a stabilized finite element algorithm," *Comput. Methods. Appl. Mech. & Eng.*, vol. 190, no. 43, pp. 5719–5737, 2001.

[89] O. Guasch and R. Codina, "Computational aeroacoustics of viscous low speed flows using subgrid scale finite element methods," *Journal of Computational Acoustics*, vol. 17, no. 03, pp. 309–330, 2009.

[90] T. Yabe and P.-Y. Wang, "Unified numerical procedure for compressible and incompressible fluid," *Journal of the Physical Society of Japan*, vol. 60, no. 7, pp. 2105–2108, 1991.

[91] O. C. Zienkiewicz and R. Codina, "A general algorithm for compressible and incompressible flow—Part I. the split, characteristic-based scheme," *International Journal for Numerical Methods in Fluids*, vol. 20, no. 8-9, pp. 869–885, 1995.

[92] R. Codina, M. Vázquez, and O. C. Zienkiewicz, "A general algorithm for compressible and incompressible flows. Part III: The semi-implicit form," *International Journal for Numerical Methods in Fluids*, vol. 27, no. 1-4, pp. 13–32, 1998.

[93] F. Xiao, R. Akoh, and S. Ii, "Unified formulation for compressible and incompressible flows by using multi-integrated moments II: Multi-dimensional version for compressible and incompressible flows," *Journal of Computational Physics*, vol. 213, no. 1, pp. 31–56, 2006.

[94] L. Pesch and J. J. van der Vegt, "A discontinuous Galerkin finite element discretization of the Euler equations for compressible and incompressible fluids," *Journal of Computational Physics*, vol. 227, no. 11, pp. 5426–5446, 2008.

[95] T. J. R. Hughes, L. Franca, and M. Mallet, "A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics," *Comput. Methods. Appl. Mech. & Eng.*, vol. 54, no. 2, pp. 223–234, 1986.

[96] M. Billaud, G. Gallice, and B. Nkonga, "A simple stabilized finite element method for solving two phase compressible–incompressible interface flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 200, no. 9, pp. 1272–1290, 2011.

[97] T. Colonius, S. K. Lele, and P. Moin, "Boundary conditions for direct computation of aerodynamic sound generation," *AIAA journal*, vol. 31, no. 9, pp. 1574–1582, 1993.

[98] B. Engquist and A. Majda, "Absorbing boundary conditions for numerical simulation of waves," *Proceedings of the National Academy of Sciences*, vol. 74, no. 5, pp. 1765–1766, 1977.

[99] D. H. Rudy and J. C. Strikwerda, "A nonreflecting outflow boundary condition for subsonic Navier-Stokes calculations," *Journal of Computational Physics*, vol. 36, no. 1, pp. 55–70, 1980.

[100] A. Bayliss and E. Turkel, "Radiation boundary conditions for wave-like equations," *Communications on Pure and applied Mathematics*, vol. 33, no. 6, pp. 707–725, 1980.

[101] T. Colonius, "Modeling artificial boundary conditions for compressible flow," *Annu. Rev. Fluid Mech.*, vol. 36, pp. 315–345, 2004.

[102] T. J. R. Hughes, G. Scovazzi, and T. Tezduyar, "Stabilized methods for compressible flows," *Journal of Scientific Computing*, vol. 43, no. 3, pp. 343–368, 2010.

[103] M. Polner, L. Pesch, and J. Van Der Vegt, "Construction of stabilization operators for Galerkin least-squares discretizations of compressible and incompressible flows," *Comput. Methods. Appl. Mech. & Eng.*, vol. 196, no. 21, pp. 2431–2448, 2007.

[104] K. W. Thompson, "Time dependent boundary conditions for hyperbolic systems," *Journal of Computational Physics*, vol. 68, no. 1, pp. 1–24, 1987.

[105] T. J. Poinsot and S. Lelef, "Boundary conditions for direct simulations of compressible viscous flows," *Journal of Computational Physics*, vol. 101, no. 1, pp. 104–129, 1992.

[106] G. Lodato, P. Domingo, and L. Vervisch, "Three-dimensional boundary conditions for direct and large-eddy simulation of compressible viscous flows," *Journal of Computational Physics*, vol. 227, no. 10, pp. 5105–5143, 2008.

[107] P. Fosso, H. Deniau, N. Lamarque, T. Poinsot, *et al.*, "Comparison of outflow boundary conditions for subsonic aeroacoustic simulations," *International Journal for Numerical Methods in Fluids*, vol. 68, no. 10, pp. 1207–1233, 2012.

[108] C. K. Tam, "Computational aeroacoustics-Issues and methods," *AIAA journal*, vol. 33, no. 10, pp. 1788–1796, 1995.

[109] R. Codina, J. Principe, and J. Baiges, "Subscales on the element boundaries in the variational two-scale finite element method," *Comput. Methods. Appl. Mech. & Eng.*, vol. 198, no. 5, pp. 838–852, 2009.

[110] R. Codina and J. Principe, "Dynamic subscales in the finite element approximation of thermally coupled incompressible flows," *International Journal for Numerical Methods in Fluids*, vol. 54, no. 6-8, pp. 707–730, 2007.

[111] M. Avila, J. Principe, and R. Codina, "A finite element dynamical nonlinear subscale approximation for the low Mach number flow equations," *Journal of Computational Physics*, vol. 230, no. 22, pp. 7988–8009, 2011.

[112] M. Avila, R. Codina, and J. Principe, "Large eddy simulation of low mach number flows using dynamic and orthogonal subgrid scales," *Computers & Fluids*, vol. 99, pp. 44–66, 2014.

[113] H. A. Van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 2, pp. 631–644, 1992.

[114] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang, *PETSc Web page*, `http://www.mcs.anl.gov/petsc`, 2015. [Online]. Available: `http://www.mcs.anl.gov/petsc`.

[115] T. F. Chan and H. A. Van der Vorst, "Approximate and incomplete factorizations," in *Parallel Numerical Algorithms*, Springer, 1997, pp. 167–202.

[116] R. Codina, J. Principe, and M. Ávila, "Finite element approximation of turbulent thermally coupled incompressible flows with numerical sub-grid scale modelling," *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 20, no. 5, pp. 492–516, 2010.

[117] C. W. Rowley, T. Colonius, and A. J. Basu, "On self-sustained oscillations in two-dimensional compressible flow over rectangular cavities," *Journal of Fluid Mechanics*, vol. 455, pp. 315–346, 2002.

[118] X. Gloerfelt, C. Bailly, and D. Juvé, "Direct computation of the noise radiated by a subsonic cavity flow and application of integral methods," *Journal of Sound and Vibration*, vol. 266, no. 1, pp. 119–146, 2003.

[119] G. A. Bres and T. Colonius, "Three-dimensional instabilities in compressible flow over open cavities," *Journal of Fluid Mechanics*, vol. 599, pp. 309–339, 2008.

[120] K. Krishnamurty, "Sound radiation from surface cutouts in high speed flow," PhD thesis, California Institute of Technology, 1956.

[121] L. Larchevêque, P. Sagaut, I. Mary, O. Labbé, and P. Comte, "Large-eddy simulation of a compressible flow past a deep cavity," *Physics of fluids*, vol. 15, no. 1, pp. 193–210, 2003.

[122] P. Lax and B. Wendroff, "Systems of conservation laws," *Communications on Pure and Applied mathematics*, vol. 13, no. 2, pp. 217–237, 1960.

[123] V. Maslov, "Propagation of shock waves in an isentropic, nonviscous gas," *Journal of Mathematical Sciences*, vol. 13, no. 1, pp. 119–163, 1980.

[124] B. Cockburn and C.-W. Shu, "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework," *Mathematics of computation*, vol. 52, no. 186, pp. 411–435, 1989.

[125] F. Bassi and S. Rebay, "A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations," *Journal of computational physics*, vol. 131, no. 2, pp. 267–279, 1997.

[126] H. Luo, L. Luo, R. Nourgaliev, V. A. Mousseau, and N. Dinh, "A reconstructed discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids," *Journal of Computational Physics*, vol. 229, no. 19, pp. 6961–6978, 2010.

[127] J. Van der Vegt and H Van der Ven, "Space–time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. General formulation," *Journal of Computational Physics*, vol. 182, no. 2, pp. 546–585, 2002.

[128] A. Pont, R. Codina, and J. Baiges, "Interpolation with restrictions between finite element meshes for flow problems in an ALE setting," *International Journal for Numerical Methods in Engineering*, 2016.

[129] G. Houzeaux and R. Codina, "Transmission conditions with constraints in finite element domain decomposition methods for flow problems," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 17, no. 3, pp. 179–190, 2001.

[130] P. M. Campbell, K. D. Devine, J. E. Flaherty, L. G. Gervasio, and J. D. Teresco, "Dynamic octree load balancing using space-filling curves," *Williams College Department of Computer Science, Technical Report CS-03*, vol. 1, p. 68, 2003.

[131] T. Tu, D. R. O'Hallaron, and J. C. López, "Etree: A database-oriented method for generating large octree meshes," *Engineering with Computers*, vol. 20, no. 2, pp. 117–128, 2004.

[132] T. Tu, D. R. O'Hallaron, and O. Ghattas, "Scalable parallel octree meshing for terascale applications," in *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, IEEE, 2005, pp. 4–4.

[133] R. S. Sampath and G. Biros, "A parallel geometric multigrid method for finite elements on octree meshes," *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 1361–1392, 2010.

[134] S. Popinet, "Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries," *Journal of Computational Physics*, vol. 190, no. 2, pp. 572–600, 2003.

[135] A. Langer, J. Lifflander, P. Miller, K.-C. Pan, L. V. Kale, and P. Ricker, "Scalable algorithms for distributed-memory adaptive mesh refinement," in *Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium on*, IEEE, 2012, pp. 100–107.

[136] C. Burstedde, O. Ghattas, M. Gurnis, T. Isaac, G. Stadler, T. Warburton, and L. Wilcox, "Extreme-scale AMR," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Computer Society, 2010, pp. 1–12.

[137] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler, "Algorithms and data structures for massively parallel generic adaptive finite element codes," *ACM Transactions on Mathematical Software*, vol. 38, no. 2, 14:1–14:28, 2011.

[138] N. Jansson, J. Hoffman, and J. Jansson, "Framework for massively parallel adaptive finite element computational fluid dynamics on tetrahedral meshes," *SIAM Journal on Scientific Computing*, vol. 34, no. 1, pp. C24–C41, 2012.

[139] B. Cockburn and C.-W. Shu, "The Runge–Kutta discontinuous Galerkin method for conservation laws V: Multidimensional systems," *Journal of Computational Physics*, vol. 141, no. 2, pp. 199–224, 1998.

[140] S. Badia and J. Baiges, "Adaptive finite element simulation of incompressible flows by hybrid continuous-discontinuous Galerkin formulations," *SIAM Journal on Scientific Computing*, vol. 35, no. 1, A491–A516, 2013.

[141] G. Karypis and V. Kumar, *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0*, `http://www.cs.umn.edu/~metis`, University of Minnesota, Minneapolis, MN, 2009.

[142] E. G. Boman, U. V. Catalyurek, C. Chevalier, and K. D. Devine, "The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering, and coloring," *Scientific Programming*, vol. 20, no. 2, pp. 129–150, 2012.

[143] O. C. Zienkiewicz and J. Z. Zhu, "A simple error estimator and adaptive procedure for practical engineerng analysis," *International Journal for Numerical Methods in Engineering*, vol. 24, no. 2, pp. 337–357, 1987.

[144] D. Irisarri and G. Hauke, "A posteriori pointwise error computation for 2-D transport equations based on the variational multiscale method," *Comput. Methods. Appl. Mech. & Eng.*, vol. 311, pp. 648–670, 2016.

[145] M. G. Larson and A. Målqvist, "Adaptive variational multiscale methods based on a posteriori error estimation: Energy norm estimates for elliptic problems," *Comput. Methods. Appl. Mech. & Eng.*, vol. 196, no. 21, pp. 2313–2324, 2007.

[146] A. ElSheikh, S. Chidiac, and W. Smith, "A posteriori error estimation based on numerical realization of the variational multiscale method," *Comput. Methods. Appl. Mech. & Eng.*, vol. 197, no. 45, pp. 3637–3656, 2008.

[147] P. Ladeveze and D. Leguillon, "Error estimate procedure in the finite element method and applications," *SIAM Journal on Numerical Analysis*, vol. 20, no. 3, pp. 485–509, 1983.

[148] S. Berrone, "Robustness in a posteriori error analysis for FEM flow models," *Numerische Mathematik*, vol. 91, no. 3, pp. 389–422, 2002.

[149] G. Hauke, M. H. Doweidar, and S. Fuentes, "Mesh adaptivity for the transport equation led by variational multiscale error estimators," *International Journal for Numerical Methods in Fluids*, vol. 69, no. 12, pp. 1835–1850, 2012.

[150] G. Hauke, D. Fuster, and F. Lizarraga, "Variational multiscale a posteriori error estimation for systems: The Euler and Navier-Stokes equations," *Comput. Methods. Appl. Mech. & Eng.*, vol. 283, pp. 1493–1524, 2015.

[151] R. Rossi, J. Cotela, N. M. Lafontaine, P. Dadvand, and S. R. Idelsohn, "Parallel adaptive mesh refinement for incompressible flow problems," *Computers & Fluids*, vol. 80, pp. 342–355, 2013.

[152] J. Oden, L. Demkowicz, W. Rachowicz, and T. Westermann, "A posteriori error analysis in finite elements: The element residual method for symmetrizable problems with applications to compressible Euler and Navier-Stokes equations," *Comput. Methods. Appl. Mech. & Eng.*, vol. 82, no. 1-3, pp. 183–203, 1990.

[153] W. Rachowicz, "An anisotropic h-adaptive finite element method for compressible Navier-Stokes equations," *Comput. Methods. Appl. Mech. & Eng.*, vol. 146, no. 3-4, pp. 231–252, 1997.

[154] G. Hauke, M. H. Doweidar, and M. Miana, "The multiscale approach to error estimation and adaptivity," *Comput. Methods. Appl. Mech. & Eng.*, vol. 195, no. 13, pp. 1573–1593, 2006.

[155] J. Baiges and R. Codina, "Variational Multiscale error estimators for solid mechanics adaptive simulations: An Orthogonal Subgrid Scale approach," *Comput. Methods. Appl. Mech. & Eng.*, vol. 325, pp. 37–55, 2017.

[156] L. El Alaoui and A. Ern, "Residual and hierarchical a posteriori error estimates for nonconforming mixed finite element methods," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 38, no. 6, 903–929, 2004.

[157] M. Arnela, O. Guasch, R. Codina, and H. Espinoza, "Finite element computation of diphthong sounds using tuned two-dimensional vocal tracts," in *Proc. of 7th Forum Acousticum, Kraków, Poland*, 2014.

[158] M. Arnela and O. Guasch, "Two-dimensional vocal tracts with three-dimensional behavior in the numerical generation of vowels," *The Journal of the Acoustical Society of America*, vol. 135, no. 1, pp. 369–379, 2014.

[159] R. Blandin, M. Arnela, R. Laboissière, X. Pelorson, O. Guasch, A. V. Hirtum, and X. Laval, "Effects of higher order propagation modes in vocal tract like geometries," *The Journal of the Acoustical Society of America*, vol. 137, no. 2, pp. 832–843, 2015.

[160] M. H. Krane, "Aeroacoustic production of low-frequency unvoiced speech sounds," *The Journal of the Acoustical Society of America*, vol. 118, no. 1, pp. 410–427, 2005.

[161] J. Lighthill, *Waves in fluids. 1978*, 1978.

[162] C. H. Shadle, "The effect of geometry on source mechanisms of fricative consonants," *J. Phonetics*, vol. 19, pp. 409–424, 1991.

[163] M. Howe and R. McGowan, "Aeroacoustics of [s]," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, vol. 461, no. 2056, pp. 1005–1028, 2005.

[164] C. Zhang, W. Zhao, S. Frankel, and L. Mongeau, "Computational aeroacoustics of phonation, Part II: Effects of flow parameters and ventricular folds," *The Journal of the Acoustical Society of America*, vol. 112, no. 5, pp. 2147–2154, 2002.

[165] G. Ramsay and C. Shadle, "The influence of geometry on the initiation of turbulence in the vocal tract during the production of fricatives," in *7th International Seminar on Speech Production (ISSP)*, 2006, pp. 581–588.

[166] A. Van Hirtum, X. Grandchamp, X. Pelorson, K. Nozaki, and S. Shimojo, "Large eddy simulation and *'in vitro'* experimental validation of flow around a teeth-shaped obstacle," *International Journal of Applied Mechanics*, vol. 2, pp. 265–279, 2010.

[167] J. Cisonni, K Nozaki, A. Van Hirtum, X. Grandchamp, and S Wada, "Numerical simulation of the influence of the orifice aperture on the flow around a teeth-shaped obstacle," *Fluid Dynamics Research*, vol. 45, no. 2, p. 025 505, 2013.

[168] K. Nozaki, "Numerical simulation of sibilant [s] using the real geometry of a human vocal tract," *High Performance Computing on Vector Systems 2010*, pp. 137–148, 2010.

[169] A. Pont, O. Guasch, J. Baiges, R. Codina, and A. van Hirtum, "Large eddy simulation of sibilant [s] aeroacoustics," Submitted.

[170] O. Guasch, A. Pont, J. Baiges, and R. Codina, "Concurrent finite element simulation of quadrupolar and dipolar flow noise in low Mach number aeroacoustics," *Computers & Fluids*, vol. 133, pp. 129–139, 2016.

[171] A. Van Hirtum, Y. Fujiso, and K. Nozaki, "The role of initial flow conditions for sibilant fricative production," *The Journal of the Acoustical Society of America*, vol. 136, no. 6, pp. 2922–2925, 2014.

[172] C. W. Rowley and D. R. Williams, "Dynamics and control of high-Reynolds-number flow over open cavities," *Annu. Rev. Fluid Mech.*, vol. 38, pp. 251–276, 2006.