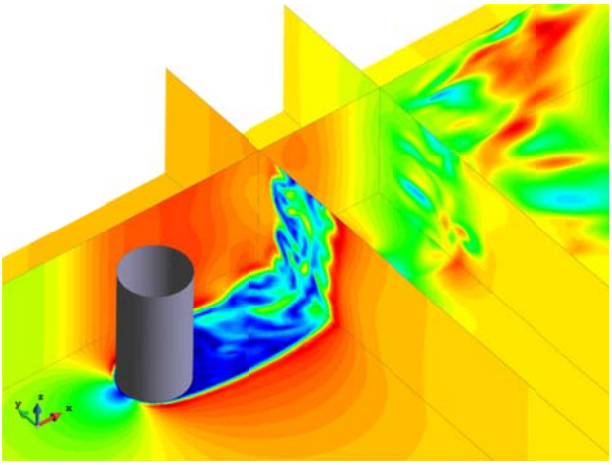
Applications of Turbulence Modeling in Civil Engineering

J. Cotela
E. Oñate
R. Rossi



Monograph CIMNE Nº-159, January 2016

Applications of Turbulence Modeling in Civil Engineering

J. Cotela
E. Oñate
R. Rossi

Monograph CIMNE Nº-159, January 2016

International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, Spain

INTERNATIONAL CENTER FOR NUMERICAL METHODS IN ENGINEERING
Edificio C1, Campus Norte UPC
Gran Capitán s/n
08034 Barcelona, Spain
www.cimne.com

First edition: January 2016

APPLICATIONS OF TURBULENCE MODELING IN CIVIL ENGINEERING
Monograph CIMNE M159

ISBN: 978-84-945077-2-4
Depósito legal: B-3248-2016

Abstract

This work explores the use of stabilized finite element formulations for the incompressible Navier-Stokes equations to simulate turbulent flow problems. Turbulence is a challenging problem due to its complex and dynamic nature and its simulation if further complicated by the fact that it involves fluid motions at vastly different length and time scales, requiring fine meshes and long simulation times. A solution to this issue is turbulence modeling, in which only the large scale part of the solution is retained and the effect of smaller turbulent motions is represented by a model, which is generally dissipative in nature.

In the context of finite element simulations for fluids, a second problem is the apparition of numerical instabilities. These can be avoided by the use of stabilized formulations, in which the problem is modified to ensure that it has a stable solution. Since stabilization methods typically introduce numerical dissipation, the relation between *numerical* and *physical* dissipation plays a crucial role in the accuracy of turbulent flow simulations. We investigate this issue by studying the behavior of stabilized finite element formulations based on the Variational Multiscale framework and on Finite Calculus, analyzing the results they provide for well-known turbulent problems, with the final goal of obtaining a method that both ensures numerical stability and introduces physically correct turbulent dissipation.

Given that, even with the use of turbulence models, turbulent flow problems require significant computational resources, we also focused on programming and parallel implementation aspects of finite element codes, and in particular in ensuring that our solver can perform efficiently on distributed memory architectures and high-performance computing clusters.

Finally, we have developed an adaptive mesh refinement technique to improve the quality of unstructured tetrahedral meshes, again with the goal of enabling the simulation of large turbulent flow problems. This technique combines an error estimator based on Variational Multiscale principles with a simple refinement procedure designed to work in a distributed memory context and we have applied it to the simulation of both turbulent and non-Newtonian flow problems.

Resum

Aquest treball estudia la possibilitat d'utilitzar formulacions estabilitzades d'elements finits de les equacions de Navier-Stokes incompressibles per a la simulació de problemes de flux turbulent. La descripció de la turbulència és un repte, ja que es tracta d'un problema altament dinàmic i complex i la seva simulació numèrica es veu complicada pel fet que hi intervenen moviments de masses fluides amb dimensions i temps característics molt diferents i per tant requereix malles de càlcul molt fines i temps de simulació llargs. Això s'ha provat de resoldre mitjançant l'ús de models de turbulència, mantenint únicament la part de la solució de més gran escala i introduint un model de l'efecte dels moviments de petita escala, que acostuma a tenir un efecte dissipatiu.

En el context de la simulació de fluids amb elements finits es planteja un segon problema amb l'aparició d'inestabilitats numèriques. Aquestes es poden evitar amb l'ús de formulacions estabilitzades, en les quals el problema es modifica per assegurar que tingui una solució estable. Ja que els mètodes d'estabilització típicament introdueixen dissipació addicional, la relació entre la dissipació numèrica i la dissipació física té un paper fonamental en la qualitat de la solució. Per investigar aquest fenomen hem estudiat el comportament de diferents formulacions d'elements finits basades en mètodes variacionals de subescala (VMS) i en el càlcul finit (FIC) en termes del seu comportament en la simulació de problemes turbulents de referència, amb l'objectiu final de trobar un mètode que a la vegada garanteixi l'estabilitat de la solució i introdueixi la dissipació turbulenta físicament necessària.

Tenint en compte que, fins i tot quan s'utilitzen models de turbulència, la simulació de problemes de flux turbulent requereix molts recursos de càlcul, també hem estudiat aspectes de la implementació paral·lela de programes d'elements finits per tal de garantir que el nostre codi pot treure partit d'arquitectures de memòria distribuïda i servidors de càlcul d'alt rendiment.

Finalment, hem desenvolupat una tècnica de refinament adaptatiu de malla que permeti millorar la qualitat de malles de càlcul tetraèdriques, novament amb la intenció de facilitar la simulació de grans problemes de flux turbulent. Aquesta tècnica combina un estimador d'error basat en els principis de la formulació variacional de subescala

amb un procediment de refinament dissenyat per funcionar fàcilment en un context de memòria distribuïda i s'ha utilitzat per simular problemes de flux turbulent i no-Newtonià.

A la meva família

Acknowledgments

In writing this document, which represents the culmination of the work that has kept me busy during the last years, it becomes clear to me how much of it has been shaped by the influence of many people who have helped me reach this point. To all of them, I am in debt and, while I am sure that making a complete list would be impossible, I will try to at least mention as many as I can.

First of all, I want to thank Eugenio Oñate for giving me the opportunity to start the present work and having the patience to let me finish it. This would never have been possible without this support.

My most sincere thanks to Riccardo Rossi. Most of what is written here I have learned from him. Again, without his encouragement and his support this work might have never reached its conclusion.

I am very grateful to Pooyan Dadvand, who has taught me a lot, and not only about programming. I also extend this gratitude to the Kratos team that he and Riccardo have built and of which I am proud to be a part of.

In addition, I want to express my gratitude for the opportunity of working at CIMNE, a stimulating research environment full of interesting people. I want to thank my office mates, past and present, for the many interesting conversations. Thanks to Jordi Rubio, Guillermo, Ricardo and specially to Pavel, Kazem, Julio, Adelina and Roberto, who helped me adapt when I landed there. Thanks to the GiD team, who have shared many coffees and beers with me, to Miguel Ángel and Salva, to Antonia and all the rest.

I want to thank Prof. Marc Parlange for hosting me at EPFL and the members of his team for making me feel at home in Lausanne. This is specially true for Marco Giometto who, in addition to being a good friend, has always encouraged me to collaborate with him in interesting problems.

This is also my opportunity to thank Roland Wüchner and Michael Andre from TUM, Enrico Stecca from Padova and many others who I have had the pleasure of collaborating with during these years.

Given the subject matter of the present work, I would be ungrateful if I did not thank Ramon Codina, Javier Príncipe, Matías Ávila and Joan Baiges for answering my

repeated questions about subscale models over the years.

I want to thank the friends that I have made over the years in and around CIMNE for their support, specially Pablo and Alessandro who helped me improve this document, but also Miquel, Lorenzo, *il Capitano* Enrico Fusto, Ilaria, Massimo, Fran, Stefano and Savvas.

I am also grateful for the opportunities I have had of working with master students. I have the feeling that I have learned much more from Daniel, Gerhard, Elisa, Matthias and Sonja than I was ever able to teach them.

I want to acknowledge the support of the *Ministerio de Educación* and the *Col·legi d'Enginyers de Camins, Canals i Ports*, Mónica de Mier and Manuel Doblaré at Abengoa for allowing me to use their data as a reference in our tests and the uLites project team providing me their test model.

Thanks to my family for their constant support and encouragement, and very specially to my parents and grandparents for making me the person I am today. This work is dedicated to them.

And finally, a heartfelt thank you to Rossella, for far too many reasons to list, but specially for her support during the final rush to finish this text. You have made all bad days seem brighter.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

1.1.1 Turbulence modeling

Turbulent flows are a subject of paramount interest in engineering, physics and earth sciences, since they govern many phenomena involving moving fluids like volumes of air or water. Turbulent phenomena appear in flows of all sizes, from large scale processes such as the atmospheric flow of air or oceanic currents to local problems such as wind loads over lightweight structures or the efficiency of a wind turbine. However, the complex nature of turbulent flows means that they can rarely be solved analytically and, as a result, we have to rely on experimental studies or numerical simulations for their analysis.

Experimental studies, which typically involve either campaigns of field measurements or wind tunnel tests, tend to be expensive and complex endeavors (see for example [113]) and, although they are an essential tool in providing a deeper understanding of the problem, tend to provide localized measurements such as point recordings or line averages, which are limited to the regions where sensors are placed, or indirect measurements, such as trajectories of tracers. From this point of view, numerical modeling represents a less costly alternative to experiments with the added advantage of providing a global image of the velocity and pressure fields for the entire volume of interest. Unfortunately, numerical study of turbulent problems is not without difficulty. The fundamental problem in turbulent flow simulations is the wide range of time and spatial scales involved [101], which have to be taken into account in the analysis.

Taking as an example the flow around an obstacle, the largest fluid motions, or eddies, in the flow can be assumed to have a characteristic size L comparable to the size of the obstacle itself. Under a turbulent flow regime, coherent flow structures tend to break and the energy associated to these motions is transferred to successively

smaller eddies until, for very localized fluctuations, viscous effects dominate and the energy is dissipated. This process can be formalized in terms of the energy associated to motions of a characteristic wavelength in what is known as the Kolmogorov energy cascade [64, 65, 101] and the size at which viscous dissipation dominates is defined as the Kolmogorov length η.

Turbulent flows can be characterized by the Reynolds number Re, a dimensionless parameter indicative of the balance between inertial and viscous forces in the problem,

$$\mathrm{Re} = \frac{\rho U L}{\mu} \tag{1.1}$$

where ρ is the fluid's density and μ its dynamic viscosity and U is a characteristic velocity of the flow. The Kolmogorov length scale can be related to the largest motions of the flow by means of the Reynolds number [101] as

$$\frac{L}{\eta} = \mathrm{Re}^{\frac{3}{4}} \tag{1.2}$$

To properly represent the full range of turbulent motions in a numerical simulation, the size of the problem domain would be a multiple of the size of the obstacle on each dimension, while the grid size would have to be small enough to capture motions on the Kolmogorov scale. For a $3D$ problem, this implies that the total number of elements would be on the order of $\mathrm{Re}^{9/4}$. Keeping in mind that the Reynolds number in engineering applications is typically on the order of $10^4 \sim 10^6$, the number of elements involved in a typical simulation would put it beyond the reach of most computers and scientific computation clusters. As a result, this approach, which is known as Direct Numerical Simulation (DNS) is only applicable to problems with a moderate Reynolds number and is not commonly used in engineering.

The only viable option in most cases is to introduce a turbulence model to reduce the computational requirements of the simulation. This typically involves neglecting the smallest motions and modifying the problem to include terms that model the effect of the neglected motions on the problem. The addition of such model increases the minimum element size required to simulate the problem, extending the range of problems that can be simulated with given computational resources. Turbulence models can be grouped into two broad families, Reynolds Averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES).

RANS models are based on rewriting the problem in terms of time-averaged variables (or time- and space-averaged, if there are spatial directions of homogeneity) and introducing a model for the effect of turbulent motions on the averaged solution. This model has a dissipative effect, removing energy from the average motions, and most commonly takes the form of an added viscosity, called turbulent or eddy viscosity. The temporal and spatial distribution of the eddy viscosity is given by the specific model and is usually determined by the solution of one or more additional equations that describe the production and transport of turbulent quantities. Typical examples of RANS

models are the Spalart-Allmaras model [116], which involves one additional equation, or the κ–ϵ [21, 68] and κ–ω models [132], which introduce two additional variables.

LES models are based on introducing a spatial filter to smooth out high wave number fluctuations in the solution and writing the problem equations in terms of filtered quantities. This allows the separation of larger flow features, influenced by the geometry of the fluid domain and the boundary conditions, from the smaller motions, which are assumed to have an universal behavior and thus can be replaced by local dissipation model, which is problem-independent. The most well known member of this family of models is the Smagorinsky method [114], but many variants and alternatives exist [101].

Note that, while the use of turbulence models makes the simulation of turbulent flows possible in terms of mesh resolution and computational resources, many flows of practical interest still require fine meshes and long simulation times in order to obtain statistically steady solutions and reliable statistical measurements. As a result, the simulation of complex flows at high Reynolds number still requires significant computational resources, frequently beyond what a single computer can provide, and has to be solved using High Performance Computing (HPC) clusters.

1.1.2 Stabilized finite element formulations for turbulent flows

A second problem frequently encountered in the finite element simulation of incompressible flows is the apparition of numerical instabilities, which are a consequence of the incompressibility constraint and the effect of the convective term in the equations for convection-dominated flows (see for example [39] for an introduction to this topic). In the context of finite elements, one possible solution for this issue is the use of stabilization techniques, where the original Galerkin weak form of the problem is modified to obtain a stable formulation. The modified equations are characterized by the addition of new terms that are typically dissipative in nature, in a way that ensures consistency, that is, that the modified problem tends to the original equations as the simulation mesh is refined.

For turbulent flows in particular, the interaction between turbulence models and stabilization terms is an active area of research. While they have clearly different origins and motivations (turbulence models are typically motivated using physical arguments, while stabilization methods are purely numerical in nature), both types of methods have a dissipative effect on the solution, which has raised some questions regarding their interaction or the possibility of using a unified formulation to provide both stability and turbulence modeling. This subject has been studied for example for Streamline-Upwind Petrov-Galerkin (SUPG) stabilization [123, 127], Finite Calculus (FIC) based formulations [92, 93] and stabilization techniques within the Variational Multiscale (VMS) framework.

In the case of VMS formulations, the stabilized problem is motivated by a separation of scales, differentiating a large scale part of the solution, that can be represented by

the finite element mesh, from a small scale part, which remains unresolved, by means of a projection of the solution onto the finite element mesh [53, 55]. This concept has intriguing parallels with the spatial filtering introduced in LES methods and can be understood as a mesh-induced filtering. This has motivated multiple investigations studying the use VMS formulations in turbulence modeling [3, 9–11, 30, 32, 44, 47, 58, 59, 102], analyzing their relationship both from the theoretical point of view and through its application to the simulation of turbulent flows of interest.

1.2 Objectives and methodology

The main topic of this monograph is the study of the applicability of stabilized finite element techniques to the solution of turbulent flow problems of practical interest in engineering, with a special focus in analyzing the behavior of different formulations as a LES-like turbulence model. The research is organized around two main lines: The first of them is centered on stabilized finite element formulations and their relationship with turbulence modeling, while the second focuses on computational techniques to facilitate the calculation of complex flows in large simulation domains.

Regarding the relationship between stabilized formulations and turbulence modeling, we choose to focus on two types of methods. The first of them is the VMS framework, where we intend to analyze the behavior of two well-known formulations, algebraic subgrid scales and orthogonal subgrid scales, as well as dynamic subscales, while the second is a new formulation derived from the application of the FIC balance to the incompressible Navier-Stokes equations. We have implemented a fluid solver based on the different techniques and used it to analyze their performance.

The first topic that will be discussed in terms of the computational aspects of the present work is the use of parallel programming techniques for the simulation of finite element problems in a distributed memory context. In this sense, all developments have been made with a parallel implementation in mind, choosing algorithms and implementations that are suited to a parallel implementation in preference to alternatives that are not, and the parallel capabilities of the implemented code have been evaluated.

A related topic in relation to the calculation of large problems is the use of adaptive mesh refinement techniques to simplify the mesh generation procedure and reduce the overall number of elements required to perform the simulation. We have developed a technique based on a simple refinement technique that can work in a distributed memory environment, which has been implemented to work in combination with the fluid solver. During the course of the present work we also found the opportunity to apply these refinement techniques to non-Newtonian flow problems. The results of our investigation in this area will also be presented.

The methods presented in this document have been implemented within the open source Kratos Multiphysics finite element framework [34, 35], which is based on C++

and Python and has parallel calculation capabilities, some of which have been developed or expanded as part of the present work. This implementation has been used to simulate all the examples considered and generate the presented results.

1.3 Outline of this document

The present document is organized along the main goals outlined above. In Chapter 2 we introduce VMS methods, with a special emphasis on dynamic subscale approximations, and review the arguments that have been used in the literature to relate them to LES formulations. Our tests have driven us to propose a variant of the method characterized by a new model for the pressure subscale. We use both the standard approach and our modified formulation to simulate a turbulent channel flow benchmark problem, which allows us to compare our results to DNS data.

In Chapter 3 we present a new stabilized formulation for incompressible flows based on the FIC balance principles. This formulation is also used to simulate the channel flow problem, as well as other benchmark examples.

Chapter 4 is concerned with different aspects of the parallel implementation of the finite element flow solver that has been used to perform the simulations presented in the previous chapters, including tests of the parallel performance of the solver. In the same chapter we present the approach we used to record spatial averages and variances in a distributed memory context.

Chapter 5 presents an adaptive mesh refinement technique that combines a parallel refinement algorithm based on edge subdivision with an error indicator motivated by the VMS formulation. This technique is then used to solve both turbulent and non-Newtonian flow cases.

Finally, in Chapter 6 we summarize the work and present its main conclusions, as well as proposing future lines of research.

Chapter 2

Variational multiscale stabilization for turbulent flow problems

2.1 Introduction

Variational multiscale (VMS) methods [53, 55] provide a theoretical framework for the design of stabilized finite element formulations based on the separation of the solution into resolved and unresolved parts, which is achieved through the definition of large scale and small scale solution spaces. The projection of the original equations onto the large scale space gives an equivalent problem that depends on the small scale variables, while the projection of the original equations onto the small scale space is used to motivate a model for the effect of the small scale variables, which are not solved, to the large scale solution.

This methodology has interesting parallels with LES turbulence models, which use a spatial filter to introduce a separation between the resolved and unresolved parts of the solution. This fact has motivated research into the relationship between VMS stabilized formulations and LES methods, and in particular on the possible use of VMS methods as turbulence models. In the present chapter we review some current trends in VMS formulations, including the derivation of algebraic and orthogonal models for the small scales, and the arguments that have been used to relate them to turbulence modeling. We also direct our attention to dynamic subscale models [28, 30], which have been shown to provide good results in turbulent flow simulations without requiring the use of an explicit turbulence model [7, 32, 102].

In spite of the success of the method in the mentioned tests, some aspects of the behavior of VMS methods as turbulence models are not completely understood. In particular, the solution of the problem has a degree of dependency on the precise definition of the stabilization parameters (see for example [7]) and on using or neglecting the pressure small scale [32]. We investigate this issue through the analysis of the results

of turbulent channel flow simulations. Additionally, we propose a new model for the pressure small scale, which we found to provide more accurate results in our simulations.

The chapter is organized as follows: in the next pages we review the state of the art for the formulations of interest, introducing the Galerkin weak form of the incompressible Navier-Stokes equations Section 2.2 and using it Section 2.3 to obtain the complete stabilized equations for the different variants that will be considered in this work. We continue by presenting the arguments that have been used to justify their use as a turbulence model in Section 2.4. In Section 2.5 we present the finite element solver that we have implemented based on the described VMS formulations. In Section 2.6 we present our proposal for a new model of pressure small scale and in Section 2.7 we use both this and standard VMS methods to simulate a turbulent channel flow test case, analyzing the solution in terms of different measured statistics of the flow and their comparison with reference DNS data. Finally, we present our conclusions in Section 2.8.

2.2 Variational form of the Navier-Stokes equations

2.2.1 Problem statement

The incompressible Navier-Stokes equations state the balance of linear momentum and mass in a fluid domain Ω, given by

$$\rho\,\partial_t \boldsymbol{u} + \rho\,\boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nabla\cdot\boldsymbol{\sigma} = \boldsymbol{f} \qquad \text{in } \Omega\times[0,T) \tag{2.1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad \text{in } \Omega\times[0,T) \tag{2.2}$$

where $\boldsymbol{u}$ is the fluid velocity, ρ its density, $\boldsymbol{\sigma}$ represents the stress tensor and $\boldsymbol{f}$ the external forces acting on the domain.

The problem described by Eqs. (2.1) and (2.2) must be completed with suitable initial and boundary conditions. We denote the domain boundary as $\partial\Omega$ and introduce its partition into Dirichlet (Γ_D) and Neumann (Γ_N) parts, verifying $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \emptyset$. The initial and boundary conditions for the problem can be expressed as:

$$\boldsymbol{u} = \boldsymbol{u}_0 \qquad \text{in } \Omega,\ t=0 \tag{2.3}$$

$$\boldsymbol{u} = \boldsymbol{u}_D \qquad \text{in } \Gamma_D\times[0,T) \tag{2.4}$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{t} \qquad \text{in } \Gamma_N\times[0,T) \tag{2.5}$$

where $\boldsymbol{u}_0$ is the initial velocity field, $\boldsymbol{u}_D$ represents the imposed velocity on the Dirichlet boundary, $\boldsymbol{n}$ the outer normal vector and $\boldsymbol{t}$ the imposed traction acting along the Neumann boundary. Note that the initial velocity $\boldsymbol{u}_0$ must be chosen to be divergence free to ensure that Eq. (2.2) is verified at all times.

For Newtonian fluids, the stress tensor $\boldsymbol{\sigma}$ can be related to the fluid velocity $\boldsymbol{u}$ and

pressure p using the constitutive relation

$$\boldsymbol{\sigma} = -p\,\boldsymbol{I} + 2\mu\left(\nabla^s\boldsymbol{u} - \frac{1}{3}\,(\nabla\cdot\boldsymbol{u})\,\boldsymbol{I}\right) \tag{2.6}$$

where $\boldsymbol{I}$ is the second order identity tensor, μ the fluid's viscosity and $\nabla^s\boldsymbol{u}$ the symmetric gradient of velocity, defined as

$$\nabla^s\boldsymbol{u} = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right) \tag{2.7}$$

To obtain a finite element formulation for the Navier-Stokes equations we need to rewrite them in weak form. We multiply Eq. (2.1) by a velocity test function $\boldsymbol{w}$, defined to be zero on the Dirichlet boundary Γ_D, and Eq. (2.2) by a pressure test function q. Integrating the resulting expressions over the fluid domain we obtain

$$\int_\Omega \boldsymbol{w}\cdot(\rho\,\partial_t\boldsymbol{u} + \rho\,\boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nabla\cdot\boldsymbol{\sigma})\;\mathrm{d}\Omega = \int_\Omega \boldsymbol{w}\cdot\boldsymbol{f}\,\mathrm{d}\Omega \tag{2.8}$$

$$\int_\Omega q\,\nabla\cdot\boldsymbol{u}\,\mathrm{d}\Omega = 0 \tag{2.9}$$

The Neumann boundary condition can be introduced in the formulation by using the product rule on the stress term in Eq. (2.8) and expressing the resulting divergence as a boundary integral, which allows us to write

$$\begin{aligned} -\int_\Omega \boldsymbol{w}\,\nabla\cdot\boldsymbol{\sigma}\,\mathrm{d}\Omega &= \int_\Omega \nabla\boldsymbol{w} : \boldsymbol{\sigma}\,\mathrm{d}\Omega - \int_\Omega \nabla\cdot(\boldsymbol{w}\cdot\boldsymbol{\sigma})\;\mathrm{d}\Omega \\ = \int_\Omega \nabla\boldsymbol{w} : \boldsymbol{\sigma}\,\mathrm{d}\Omega + \int_{\partial\Omega} \boldsymbol{w}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma &= \int_\Omega \nabla\boldsymbol{w} : \boldsymbol{\sigma}\,\mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}\cdot\boldsymbol{t}\,\mathrm{d}\Gamma \end{aligned} \tag{2.10}$$

Substituting Eq. (2.10) into Eq. (2.8) and introducing the definition of the stress tensor in Eq. (2.6) we obtain

$$\begin{aligned} \int_\Omega \boldsymbol{w}\cdot(\rho\,\partial_t\boldsymbol{u} + \rho\,\boldsymbol{u}\cdot\nabla\boldsymbol{u})\,\mathrm{d}\Omega + \int_\Omega \nabla^s\boldsymbol{w} :\; 2\mu\left(\nabla^s\boldsymbol{u} - \frac{1}{3}\,(\nabla\cdot\boldsymbol{u})\right)\;\mathrm{d}\Omega \\ - \int_\Omega \nabla\cdot\boldsymbol{w}\,p\,\mathrm{d}\Omega = \int_\Omega \boldsymbol{w}\cdot\boldsymbol{f}\,\mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}\cdot\boldsymbol{t}\,\mathrm{d}\Gamma \end{aligned} \tag{2.11}$$

$$\int_\Omega q\,\nabla\cdot\boldsymbol{u}\,\mathrm{d}\Omega = 0 \tag{2.12}$$

Eqs. (2.11) and (2.12), together with the initial and Dirichlet boundary conditions, allow us to state the weak form of the problem. In addition, they also impose regularity requirements on the solution, test functions and problem data, as it must be ensured that all the integrals that appear in the equations remain bounded. This is only briefly

discussed here, directing the reader to more specific literature on this topic for a rigorous formulation (see for example [41, 124]). In general, for any two given functions f, g we want to ensure that

$$\int_\Omega f g \, \mathrm{d}\Omega < \infty$$

We define the L^2 norm of a function as

$$\|f\|_{L^2(\Omega)} = \left(\int_\Omega f^2 \, \mathrm{d}\Omega \right)^{\frac{1}{2}}$$

and functions with bounded L^2 norm are said to be square-integrable. The space of functions that are square-integrable in Ω is denoted as $L^2(\Omega)$.

Although no proof is given here, it can be shown that, for any given time instant t, it is sufficient to require that both the momentum test function $\boldsymbol{w}$, the velocity solution $\boldsymbol{u}$ and their first order derivatives belong to $L^2(\Omega)$. The space of functions verifying this property is a Hilbert space commonly denoted as $H^1(\Omega)$ in functional analysis. For the mass conservation test function q and the pressure solution p, it is enough to require them to be square-integrable, as their spatial derivatives do not appear in Eqs. (2.11) and (2.12).

Considering that $\boldsymbol{u}$ must verify the Dirichlet boundary condition when evaluated in the Dirichlet boundary Γ_D and $\boldsymbol{w}$ is zero in Γ_D by definition, the solutions (for any fixed instant in time) and test functions must be contained in the spaces of functions given by

$$\begin{aligned} \boldsymbol{u} &\in H^1_D = \left\{ \boldsymbol{u} \in H^1(\Omega) \;\middle|\; \boldsymbol{u} = \boldsymbol{u}_D \text{ in } \Gamma_D \right\} \\ \boldsymbol{w} &\in H^1_0 = \left\{ \boldsymbol{w} \in H^1(\Omega) \;\middle|\; \boldsymbol{w} = \boldsymbol{0} \text{ in } \Gamma_D \right\} \\ p, q &\in L^2(\Omega) \end{aligned}$$

Likewise, the external forces $\boldsymbol{f}$ must be such that the domain integral in the right hand side of Eq. (2.11) is well defined. Given that $\boldsymbol{w} \in H^1(\Omega)$, this is equivalent to requiring the forces to belong to the dual of that space, denoted as $H^{-1}(\Omega)$. There is a similar requirement on the traction $\boldsymbol{t}$, as it has to be integrable when multiplied by the test function $\boldsymbol{w}$ restricted to the Neumann boundary, but it will not be formally stated here.

Finally, to make sure that the dynamic problem is well-posed it is sufficient to require that $\|\boldsymbol{u}\|_{L^2(\Omega)}$ and $\|\partial u_i / \partial x_j\|_{L^2(\Omega)}$ square-integrable along the time interval of the problem. This is denoted as $\boldsymbol{u} \in L^2(0, T, H^1_D(\Omega))$. In the case of the pressure it is enough to enforce that the L^2 norm is square-integrable in time, that is, $p \in L^2(0, T, L^2(\Omega))$.

2.2.2 Conservation properties

Before we introduce the variational form of the problem, there is an important remark to be made about the convective term in the momentum equation Eq. (2.1). As long as

the velocity field is strongly (point-wise) divergence-free, the following three expressions are identical:

$$\boldsymbol{u}\cdot\nabla\boldsymbol{u} = \nabla\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) = \frac{1}{2}\,\boldsymbol{u}\cdot\nabla\boldsymbol{u} + \frac{1}{2}\,\nabla\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) \tag{2.13}$$

or, in variational form,

$$\begin{aligned} \int_\Omega \boldsymbol{w}\cdot(\boldsymbol{u}\cdot\nabla\boldsymbol{u})\;\mathrm{d}\Omega = \\ -\int_\Omega \nabla\boldsymbol{w} : (\boldsymbol{u}\otimes\boldsymbol{u})\;\mathrm{d}\Omega + \int_{\Gamma_N} (\boldsymbol{w}\cdot\boldsymbol{u})\,(\boldsymbol{u}\cdot\boldsymbol{n})\;\mathrm{d}\Gamma = \\ \frac{1}{2}\int_\Omega \boldsymbol{w}\cdot(\boldsymbol{u}\cdot\nabla\boldsymbol{u})\;\mathrm{d}\Omega - \frac{1}{2}\int_\Omega \nabla\boldsymbol{w} : (\boldsymbol{u}\otimes\boldsymbol{u})\;\mathrm{d}\Omega + \frac{1}{2}\int_{\Gamma_N} (\boldsymbol{w}\cdot\boldsymbol{u})\,(\boldsymbol{u}\cdot\boldsymbol{n})\;\mathrm{d}\Gamma \end{aligned} \tag{2.14}$$

The three expressions in Eq. (2.13) are respectively known as *non-conservative*, *conservative* and *skew-symmetric* forms of the convective term. However, in practice, the expressions in Eq. (2.14) are not equivalent for the discrete problem, as the velocity solution is only divergence-free in a weak (integral) sense. In fact, as shown in [29], each of them gives rise to a variational problem with different conservation properties. While the continuous Navier-Stokes equations enforce the balance of linear and angular momentum, as well as kinetic energy, none of the discrete variants ensures conservation of the three quantities at the same time. According to the analysis in the same reference, the variational form resulting from each expression conserves the quantities listed in Table 2.1.

Convective term	Linear momentum	Angular mom.	Kinetic energy
Non-conservative	For equal $\boldsymbol{u}$-p interpolations	No	No
Conservative	Yes	Yes	No
Skew-symmetric	For equal $\boldsymbol{u}$-p interpolations	No	Yes

Table 2.1: Conserved quantities in the discrete Navier-Stokes equations depending on the expression of the convective term, according to [29].

In the present work we have used the skew-symmetric form, as the kinetic energy balance is an important concern in the present study and something that can be measured to to gain insight on turbulent flow simulations. In fact, we can report that using a formulation based on the skew-symmetric form resulted in a better fit to DNS data in our preliminary tests for the channel flow simulations presented in Section 2.7.

2.2.3 Galerkin weak form

Using the skew-symmetric form for the convective term, the Galerkin weak form of the Navier-Stokes problem can be stated as

Find $\boldsymbol{u} \in L^2\left(0,T,H^1_D(\Omega)\right)$, $p \in L^2\left(0,T,L^2(\Omega)\right)$ *such that,* $\forall \boldsymbol{w} \in H^1_0(\Omega), \forall q \in L^2(\Omega)$,

$$\begin{aligned}\int_\Omega \boldsymbol{w}\cdot\left(\rho\,\partial_t\boldsymbol{u}+\rho\frac{1}{2}\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)\,\mathrm{d}\Omega-\int_\Omega\nabla\boldsymbol{w}:\rho\frac{1}{2}\left(\boldsymbol{u}\otimes\boldsymbol{u}\right)\,\mathrm{d}\Omega\\ +\int_\Omega\nabla^s\boldsymbol{w}:\,2\mu\left(\nabla^s\boldsymbol{u}-\frac{1}{3}\left(\nabla\cdot\boldsymbol{u}\right)\right)\,\mathrm{d}\Omega-\int_\Omega\nabla\cdot\boldsymbol{w}\,p\,\mathrm{d}\Omega=\\ \int_\Omega\boldsymbol{w}\cdot\boldsymbol{f}\,\mathrm{d}\Omega+\int_{\Gamma_N}\boldsymbol{w}\cdot\left(\boldsymbol{t}+\rho\frac{1}{2}\left(\boldsymbol{u}\cdot\boldsymbol{n}\right)\boldsymbol{u}\right)\,\mathrm{d}\Gamma\\ \int_\Omega q\,\nabla\cdot\boldsymbol{u}\,\mathrm{d}\Omega=0\end{aligned}$$

Unfortunately, this problem is not straightforward to solve using finite elements, as its discrete version is not numerically stable. In fact, to ensure that that problem of finding a pair $\boldsymbol{u}, p$ that satisfies the above weak form for all suitable choices of $\boldsymbol{w}, q$ has a stable solution, the discrete spaces in which the solution is sought must verify the *inf-sup* or Ladyzhenskaya-Babuška-Brezzi (LBB) condition, given by

$$\inf_{q\neq 0\ q\in Q}\sup_{\boldsymbol{w}\neq\boldsymbol{0}\ \boldsymbol{w}\in V}\frac{\int_\Omega q\nabla\cdot\boldsymbol{w}\,\mathrm{d}\Omega}{\|q\|_Q\,\|\boldsymbol{w}\|_V}\geq C \tag{2.15}$$

where V and Q are spaces containing the velocity and pressure solutions, respectively, and C is a positive constant. In practice, satisfaction of the LBB condition implies the use of a higher order interpolation for velocity than for pressure, as is done for example in Taylor-Hood elements [122].

Numerical instabilities in the discrete solution of the Navier-Stokes may also appear in convection-dominated flows, that is, when the convective term is large in relation to the viscous term. In turbulent flows, this issue can be understood from a physical point of view by noting that viscous dissipation occurs predominantly due to high velocity gradients at small scales. If the finite element mesh is coarse, these gradients cannot be reproduced and dissipation is underestimated, which leads to energy accumulation on the larger scales and the eventual loss of convergence of the solution.

Both instabilities can be *cured* by the use of a stabilized formulation, which involves obtaining a modified weak form not restricted by the inf-sup condition of Eq. (2.15). This has the advantage of allowing the use of equal order interpolations for velocity and pressure. Two classical stabilized formulations for the Navier-Stokes equations are known as Streamline-Upwind Petrov-Galerkin (SUPG) [57] and Galerkin-Least Squares (GLS) [56]. Other alternatives are methods within the VMS framework, which are the main subject of the present chapter, and the Finite Calculus (FIC) approach, presented in Chapter 3.

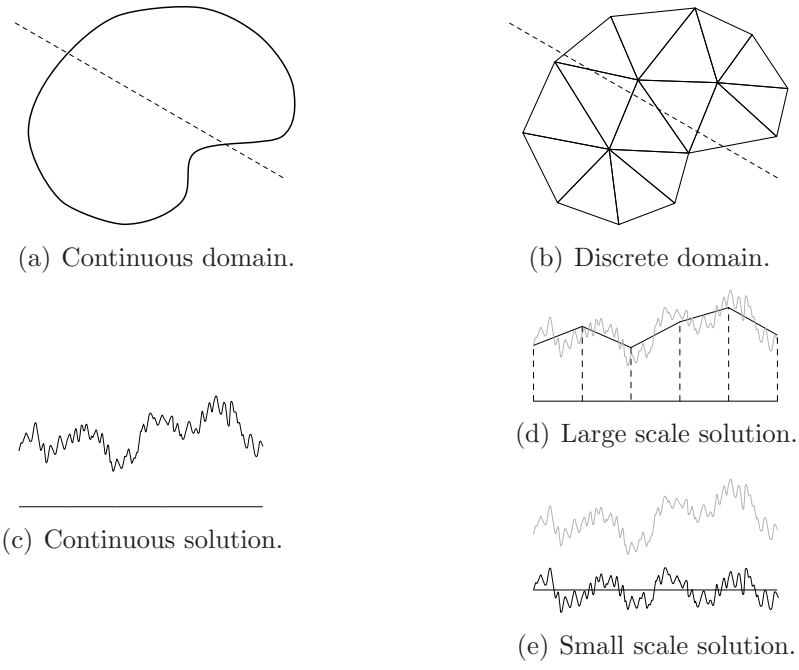
2.3 Variational multiscale stabilization

2.3.1 Scale separation

The Variational Multiscale (VMS) method, introduced in [53, 55, 60] is a theoretical framework for the development of stabilized finite element methods that has been used extensively during the last two decades in finite element formulations for fluid problems. The basic premise of this method is the separation of the problem variables into large scale $(\cdot)_h$ and small scale, or subscale, values $(\cdot)_s$, which in our case corresponds to

$$\boldsymbol{u} = \boldsymbol{u}_h + \boldsymbol{u}_s \quad p = p_h + p_s$$
$$\boldsymbol{w} = \boldsymbol{w}_h + \boldsymbol{w}_s \quad q = q_h + q_s$$

The $(\cdot)_h$ notation chosen to represent the large scales should be understood as a reference to the finite element mesh size h. In practice, scale separation is closely related to the spatial discretization used to solve the problem. This is shown graphically in Figure 2.1, where the solution along a line is represented. The large scales correspond to the part of the solution that can be described using the finite element interpolation, while the small scales represent fine features of the solution that cannot be reproduced due to the limitations of the discrete interpolation.



(a) Continuous domain.

(b) Discrete domain.

(c) Continuous solution.

(d) Large scale solution.

(e) Small scale solution.

Figure 2.1: Scale separation.

Once a finite element discretization is defined, the space containing the large scale part of the solution can be identified with that of the admissible finite element functions

on the discrete domain. This implies that, instead of working with test functions and solutions defined on infinite-dimensional spaces of functions, we now seek a solution on a restricted, finite-dimensional space of admissible solutions, expressed as

$$\boldsymbol{u}_h \in W_h \subset W \equiv L^2\left(0,T,H_D^1(\Omega)\right) \qquad p_h \in Q_h \subset Q \equiv L^2\left(0,T,L^2(\Omega)\right)$$

This in turn allows us to define spaces containing the small scale part of the solution $\boldsymbol{w}_s \in W_s$, $q_s \in Q_s$. The subscale spaces complete the corresponding large scale space, that is, $W = W_h \oplus W_s$, $Q = Q_h \oplus Q_s$. As a result, it is clear that small scale spaces are infinite-dimensional, unlike the large scale ones, and have to be approximated in order to obtain a solution. There is no single way to approximate them and, in fact, the choice of approximate space for the small scales is one of the defining features of the solution method.

We introduce the following compact notation

$$\begin{aligned}
\mathcal{B}\left(\boldsymbol{w},\boldsymbol{a},\boldsymbol{u}\right) &= \int_\Omega \boldsymbol{w}\cdot\left(\rho\,\partial_t\boldsymbol{u} + \rho\frac{1}{2}\boldsymbol{a}\cdot\nabla\boldsymbol{u}\right)\,\mathrm{d}\Omega - \int_\Omega \nabla\boldsymbol{w} : \rho\frac{1}{2}\left(\boldsymbol{a}\otimes\boldsymbol{u}\right)\,\mathrm{d}\Omega \\
&\quad + \int_\Omega \nabla^s\boldsymbol{w} :\, 2\mu\left(\nabla^s\boldsymbol{u} - \frac{1}{3}\left(\nabla\cdot\boldsymbol{u}\right)\right)\,\mathrm{d}\Omega + \int_{\Gamma_N}\boldsymbol{w}\cdot\rho\frac{1}{2}\left(\boldsymbol{a}\cdot\boldsymbol{n}\right)\boldsymbol{u}\,\mathrm{d}\Gamma \\
\mathcal{D}\left(\boldsymbol{w},p\right) &= \int_\Omega \nabla\cdot\boldsymbol{w}\,p\,\mathrm{d}\Omega \\
\mathcal{L}\left(\boldsymbol{w}\right) &= \int_\Omega \boldsymbol{w}\cdot\boldsymbol{f}\,\mathrm{d}\Omega + \int_{\Gamma_N}\boldsymbol{w}\cdot\boldsymbol{t}\,\mathrm{d}\Gamma
\end{aligned}$$

Note that we have modified the convective terms in $\mathcal{B}\left(\boldsymbol{w},\boldsymbol{a},\boldsymbol{u}\right)$ to introduce a convection velocity $\boldsymbol{a}$. This notation is introduced for convenience, as it will allow us to introduce a linearized version of the operator later. Using the compact notation, the weak form of the problem can be expressed as

$$\mathcal{B}\left(\boldsymbol{w},\boldsymbol{u},\boldsymbol{u}\right) - \mathcal{D}\left(\boldsymbol{w},p\right) = \mathcal{L}\left(\boldsymbol{w}\right) \tag{2.16}$$

$$\mathcal{D}\left(\boldsymbol{u},q\right) = 0 \tag{2.17}$$

We introduce the scale separation of the problem variables in Eqs. (2.16) and (2.17). Testing against large scale functions, we can write

$$\mathcal{B}\left(\boldsymbol{w}_h,\boldsymbol{a},\boldsymbol{u}_h+\boldsymbol{u}_s\right) - \mathcal{D}\left(\boldsymbol{w}_h,p_h+p_s\right) = \mathcal{L}\left(\boldsymbol{w}_h\right) \tag{2.18}$$

$$\mathcal{D}\left(\boldsymbol{u}_h+\boldsymbol{u}_s,q_h\right) = 0 \tag{2.19}$$

If the small scale test functions are used instead, the following expression is obtained

$$\mathcal{B}\left(\boldsymbol{w}_s,\boldsymbol{a},\boldsymbol{u}_h+\boldsymbol{u}_s\right) - \mathcal{D}\left(\boldsymbol{w}_s,p_h+p_s\right) = \mathcal{L}\left(\boldsymbol{w}_s\right) \tag{2.20}$$

$$\mathcal{D}\left(\boldsymbol{u}_h+\boldsymbol{u}_s,q_s\right) = 0 \tag{2.21}$$

The large scale problem given by Eqs. (2.18) and (2.19) represents the finite element approximation of the original problem, now containing terms that describe the effect of the unresolved scales on the large scale solution. These additional terms cannot be evaluated in practice, as the small scale variables are not known. VMS methods are based on the definition of a model for the small scale variables, which is motivated by the small scale problem of Eqs. (2.20) and (2.21). This model can then be introduced in the large scale equations, closing the problem.

Before we apply this procedure to our problem, it is convenient to operate on Eqs. (2.18) and (2.19) to eliminate all spatial derivatives of small scale functions, obtaining an expression that depends on $\boldsymbol{u}_s$ and p_s, which will be modeled, but not on their gradients, which will remain unknown. We start by separating the terms involving the large and small scale parts of the solution

$$\begin{aligned}\mathcal{B}\left(\boldsymbol{w}_h, \boldsymbol{a}, \boldsymbol{u}_h\right) - \mathcal{D}\left(\boldsymbol{w}_h, p_h\right) + \mathcal{B}\left(\boldsymbol{w}_h, \boldsymbol{a}, \boldsymbol{u}_s\right) - \mathcal{D}\left(\boldsymbol{w}_h, p_s\right) &= \mathcal{L}\left(\boldsymbol{w}_h\right) \\ \mathcal{D}\left(\boldsymbol{u}_h, q_h\right) + \mathcal{D}\left(\boldsymbol{u}_s, q_h\right) &= 0\end{aligned}$$

Integrating by parts within each element in the mesh, differential operators acting on $\boldsymbol{u}_s$ and p_s are moved to the test functions. To do so, we introduce the notation of Ω^e to refer to the part of the problem domain corresponding to element e and Γ^e to denote its boundary.

$$\begin{aligned}&\mathcal{B}\left(\boldsymbol{w}_h, \boldsymbol{a}, \boldsymbol{u}_h\right) - \mathcal{D}\left(\boldsymbol{w}_h, p_h\right) + \int_\Omega \boldsymbol{w}_h \, \rho \partial_t \boldsymbol{u}_s \, \mathrm{d}\Omega \\ &- \sum_e \int_{\Omega^e} \rho \left(\boldsymbol{a} \cdot \nabla \boldsymbol{w}_h\right) \cdot \boldsymbol{u}_s \, \mathrm{d}\Omega + \sum_e \int_{\Gamma^e} \boldsymbol{w} \cdot \rho \frac{1}{2} \left(\boldsymbol{a} \cdot \boldsymbol{n}\right) \boldsymbol{u}_s \, \mathrm{d}\Gamma \\ &- \sum_e \int_{\Omega^e} 2\mu \nabla \cdot \left(\nabla^s \boldsymbol{w}_h - \frac{1}{3}\left(\nabla \cdot \boldsymbol{w}_h\right)\right) \boldsymbol{u}_s \, \mathrm{d}\Omega - \int_\Omega \nabla \cdot \boldsymbol{w}_h \, p_s \, \mathrm{d}\Omega \\ &+ \sum_e \int_{\Gamma^e} 2\mu \boldsymbol{w}_h \cdot \left(\nabla^s \boldsymbol{u}_s \cdot \boldsymbol{n} - \frac{1}{3}\left(\nabla \cdot \boldsymbol{u}_s\right) \boldsymbol{n}\right) \mathrm{d}\Gamma = \mathcal{L}\left(\boldsymbol{w}_h\right)\end{aligned} \tag{2.22}$$

$$\mathcal{D}\left(\boldsymbol{u}_h, q_h\right) = \sum_e \int_{\Omega^e} \nabla q_h \, \boldsymbol{u}_s \, \mathrm{d}\Omega - \sum_e \int_{\Gamma^e} q_h \boldsymbol{u}_s \cdot \boldsymbol{n} \, \mathrm{d}\Gamma \tag{2.23}$$

In the present work, the boundary integrals appearing in Eqs. (2.22) and (2.23) will be neglected. This is common in VMS formulations, and can be understood as assuming that the small scale unknowns vanish on element boundaries.

In the following we express this element-by-element integration using the notation

$$\sum_e \int_{\Omega^e} \mathrm{d}\Omega = \int_{\Sigma \, \Omega^e} \mathrm{d}\Omega$$

2.3.2 Small scale equation

Now the question is to define a model for $\boldsymbol{u}_s$ and p_s that can be used to evaluate the domain integrals in Eqs. (2.22) and (2.23). To do this, we start from the small scale problem, given by Eqs. (2.20) and (2.21), which can be developed to read

$$\begin{aligned}\int_\Omega \boldsymbol{w}_s \cdot \rho \left(\partial_t \boldsymbol{u}_h + \partial_t \boldsymbol{u}_s + \frac{1}{2}\, \boldsymbol{a} \cdot \nabla \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \right) \mathrm{d}\Omega - \int_\Omega \nabla \boldsymbol{w}_s : \rho \frac{1}{2} \left(\boldsymbol{a} \otimes \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \right) \mathrm{d}\Omega \\ + \int_\Omega \nabla^s \boldsymbol{w}_s : 2\mu \left(\nabla^s \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) - \frac{1}{3} \nabla \cdot \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \boldsymbol{I} \right) \mathrm{d}\Omega - \int_\Omega \nabla \cdot \boldsymbol{w}_s \, \left(p_h + p_s \right) \mathrm{d}\Omega = \\ \int_\Omega \boldsymbol{w}_s \cdot \boldsymbol{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}_s \cdot \left(\boldsymbol{t} + \rho \frac{1}{2} \left(\boldsymbol{a} \cdot \boldsymbol{n} \right) \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \right) \mathrm{d}\Gamma \end{aligned} \tag{2.24}$$

$$\int_\Omega q_s \, \nabla \cdot \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \mathrm{d}\Omega = 0 \tag{2.25}$$

Again, it is convenient to use integration by parts element-by-element on some of the terms in the momentum equation, obtaining the expression

$$\begin{aligned}\int_{\Sigma\, \Omega^e} \boldsymbol{w}_s \cdot \rho \left(\partial_t \boldsymbol{u}_h + \partial_t \boldsymbol{u}_s + \frac{1}{2}\, \boldsymbol{a} \cdot \nabla \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) + \frac{1}{2} \nabla \cdot \left(\boldsymbol{a} \otimes \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \right) \right) \mathrm{d}\Omega \\ + \int_{\Sigma\, \Omega^e} \boldsymbol{w}_s \nabla \cdot \left(\left(p_h + p_s \right) \boldsymbol{I} - 2\mu \left(\nabla^s \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) - \frac{1}{3} \nabla \cdot \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \boldsymbol{I} \right) \right) \mathrm{d}\Omega \\ + \sum_e \int_{\Gamma^e} \boldsymbol{w}_s \left(\left(p_h + p_s \right) \cdot \boldsymbol{n} + 2\mu \left(\nabla^s \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) - \frac{1}{3} \nabla \cdot \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \right) \cdot \boldsymbol{n} \right) \mathrm{d}\Gamma_e \\ + \int_{\Gamma_N} \boldsymbol{w}_s \cdot \rho \frac{1}{2} \left(\boldsymbol{a} \cdot \boldsymbol{n} \right) \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \mathrm{d}\Gamma - \sum_e \int_{\Gamma^e} \boldsymbol{w}_s \cdot \rho \frac{1}{2} \left(\boldsymbol{a} \cdot \boldsymbol{n} \right) \left(\boldsymbol{u}_h + \boldsymbol{u}_s \right) \mathrm{d}\Gamma = \\ \int_\Omega \boldsymbol{w}_s \cdot \boldsymbol{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}_s \cdot \boldsymbol{t} \, \mathrm{d}\Gamma \end{aligned} \tag{2.26}$$

Note that the elemental boundary integrals in Eq. (2.26) vanish over internal boundaries because they involve either the exact traction or the exact velocity over the boundary, which are both continuous. Similarly, they cancel out with the corresponding traction in the Neumann boundary. As a result, all boundary terms in Eq. (2.26) can be eliminated in the following.

Rearranging terms in Eqs. (2.26) and (2.25) to separate large and small scale un-

knowns we can write

$$\begin{aligned}\int_{\Sigma\,\Omega^e} \boldsymbol{w}_s \cdot \rho \left(\partial_t \boldsymbol{u}_s + \frac{1}{2}\,\boldsymbol{a}\cdot\nabla\boldsymbol{u}_s + \frac{1}{2}\,\nabla\cdot(\boldsymbol{a}\otimes\boldsymbol{u}_s)\right)\mathrm{d}\Omega \\ \int_{\Sigma\,\Omega^e} \boldsymbol{w}_s \cdot \left(\nabla p_s - 2\mu\nabla\cdot\left(\nabla^s\boldsymbol{u}_s - \frac{1}{3}\,(\nabla\cdot\boldsymbol{u}_s)\,\boldsymbol{I}\right)\right)\mathrm{d}\Omega = \\ \int_\Omega \boldsymbol{w}_s\cdot\boldsymbol{f}\,\mathrm{d}\Omega - \int_{\Sigma\,\Omega^e} \boldsymbol{w}_s \cdot \rho \left(\partial_t \boldsymbol{u}_h + \frac{1}{2}\,\boldsymbol{a}\cdot\nabla\boldsymbol{u}_h + \frac{1}{2}\,\nabla\cdot(\boldsymbol{a}\otimes\boldsymbol{u}_h)\right)\mathrm{d}\Omega \\ - \int_{\Sigma\,\Omega^e} \boldsymbol{w}_s \cdot \left(\nabla p_h - 2\mu\nabla\cdot\left(\nabla^s\boldsymbol{u}_h - \frac{1}{3}\,(\nabla\cdot\boldsymbol{u}_h)\,\boldsymbol{I}\right)\right)\mathrm{d}\Omega\end{aligned} \tag{2.27}$$

$$\int_\Omega q_s\,\nabla\cdot\boldsymbol{u}_s\,\mathrm{d}\Omega = -\int_\Omega q_s\,\nabla\cdot\boldsymbol{u}_h\,\mathrm{d}\Omega \tag{2.28}$$

Eqs. (2.27) and (2.28) can be understood as the L^2 projection onto the space of small scales $W_s \times Q_s$ of a differential equation, in the same sense as Eqs. (2.8) and (2.9) represent the L^2 projection onto $W \times Q$ of the Navier-Stokes equations. Moreover, observing the right hand side terms of Eq. (2.27), we can see that it represents the projection of the strong-form linear momentum equation applied to the large scale part of the solution $\boldsymbol{u}_h$,p_h. The same can be said about the right hand side of Eq. (2.27), which corresponds to the mass conservation equation applied to the large scale velocity.

Denoting the projection onto the small scale spaces W_s, Q_s with $\Pi_{V_s}(\cdot)$ and $\Pi_{Q_s}(\cdot)$, respectively, Eqs. (2.27) and (2.28) equation can be stated as

$$\begin{aligned}\Pi_{V_s}\left(\rho\partial_t\boldsymbol{u}_s + \rho\frac{1}{2}\,\boldsymbol{a}\cdot\nabla\boldsymbol{u}_s + \rho\frac{1}{2}\,\nabla\cdot(\boldsymbol{a}\otimes\boldsymbol{u}_s)\right. \\ \left. +\nabla p_s - 2\mu\nabla\cdot\left(\nabla^s\boldsymbol{u}_s - \frac{1}{3}\,(\nabla\cdot\boldsymbol{u}_s)\,\boldsymbol{I}\right)\right) = \\ \Pi_{V_s}\left(\boldsymbol{f} - \rho\partial_t\boldsymbol{u}_h - \rho\frac{1}{2}\,\boldsymbol{a}\cdot\nabla\boldsymbol{u}_h - \rho\frac{1}{2}\,\nabla\cdot(\boldsymbol{a}\otimes\boldsymbol{u}_h)\right. \\ \left. -\nabla p_s + 2\mu\nabla\cdot\left(\nabla^s\boldsymbol{u}_h - \frac{1}{3}\,(\nabla\cdot\boldsymbol{u}_h)\,\boldsymbol{I}\right)\right)\end{aligned} \tag{2.29}$$

$$\Pi_{Q_s}(\nabla\cdot\boldsymbol{u}_s) = \Pi_{Q_s}(-\nabla\cdot\boldsymbol{u}_h) \tag{2.30}$$

Since Eqs. (2.29) and (2.30) must hold for all admissible small-scale test functions $\boldsymbol{w}_s$ and q_s, they are equivalent to imposing that the small scale variables $\boldsymbol{u}_s$, p_s verify the following problem in each element Ω^e:

$$\begin{aligned}\rho\partial_t\boldsymbol{u}_s + \rho\frac{1}{2}\,\boldsymbol{a}\cdot\nabla\boldsymbol{u}_s + \rho\frac{1}{2}\,\nabla\cdot(\boldsymbol{a}\otimes\boldsymbol{u}_s) + \nabla p_s \\ -2\mu\nabla\cdot\left(\nabla^s\boldsymbol{u}_s - \frac{1}{3}\,(\nabla\cdot\boldsymbol{u}_s)\,\boldsymbol{I}\right) = \boldsymbol{R}^m(\boldsymbol{u}_h, p_h) - \boldsymbol{\xi}_h \qquad \text{in } \Omega^e\times[0,T)\end{aligned} \tag{2.31}$$

$$\nabla\cdot\boldsymbol{u}_s = R^c(\boldsymbol{u}_h) - \delta_h \qquad \text{in } \Omega^e\times[0,T) \tag{2.32}$$

where $\boldsymbol{R}^m(\boldsymbol{u}_h, p_h)$ and $R^c(\boldsymbol{u}_h)$ represent the residual form of the Navier-Stokes equations applied to the large scale variables

$$\boldsymbol{R}^m(\boldsymbol{u}_h, p_h) = \boldsymbol{f} - \rho\partial_t \boldsymbol{u}_h - \rho\frac{1}{2}\boldsymbol{a}\cdot\nabla\boldsymbol{u}_h - \rho\frac{1}{2}\nabla\cdot(\boldsymbol{a}\otimes\boldsymbol{u}_h) - \nabla p_h + 2\mu\nabla\cdot\left(\nabla^s\boldsymbol{u}_h - \frac{1}{3}(\nabla\cdot\boldsymbol{u}_h)\boldsymbol{I}\right) \tag{2.33}$$

$$R^c(\boldsymbol{u}_h) = -\nabla\cdot\boldsymbol{u}_h \tag{2.34}$$

and $\boldsymbol{\xi}_h$ and δ_h are chosen to enforce that right hand sides of Eqs. (2.31) and (2.32) belong to the corresponding small scale space.

As mentioned in the previous pages, the small scale spaces are infinite-dimensional and have to be approximated before the problem given by Eqs. (2.31) and (2.32) can be solved. In practice, the definition of an approximate small scale space corresponds to a choice of projectors $\Pi_{V_s}(\cdot)$ and $\Pi_{Q_s}(\cdot)$. The most straightforward possibility is to use the entire residual (without projecting to a particular space), which corresponds to considering operators $\Pi_{V_s}(\cdot)$, $\Pi_{Q_s}(\cdot)$ equal to the identity function or, equivalently, $\boldsymbol{\xi}_h = \boldsymbol{0}$ and $\delta_h = 0$. This formulation gives rise to the algebraic sub-grid scale (ASGS) method [27].

Another well-known choice consists in taking a small scale space that is orthogonal to the large scale space. If $\Pi_{V_h}(\cdot)$ and $\Pi_{Q_h}(\cdot)$ are the L^2 projection onto the large scale spaces W_h and Q_h respectively, then the projections in Eqs. (2.29) and (2.30) are defined as $\Pi_{V_s}(\cdot) \approx \Pi^{\perp}_{V_h}(\cdot)$, $\Pi_{Q_s}(\cdot) \approx \Pi^{\perp}_{Q_h}(\cdot)$. In this case, $\boldsymbol{\xi}_h$ and δ_h are chosen to subtract from the equation the part of the residuals that belongs to the finite element space, that is,

$$\boldsymbol{\xi}_h = \Pi_{V_h}(\boldsymbol{R}^m(\boldsymbol{u}_h, p_h)) \tag{2.35}$$

$$\delta_h = \Pi_{Q_h}(R^c(\boldsymbol{u}_h)) \tag{2.36}$$

This choice leads to the orthogonal sub-scale (OSS) method, presented in [25, 28].

Note that, due to their definition, $\boldsymbol{\xi}_h$ and δ_h belong to the space of finite element functions and can be constructed from their values on mesh nodes using standard finite element interpolation functions.

A second important remark is that the calculation of $\boldsymbol{\xi}_h$ and δ_h requires knowledge of the finite element solutions $\boldsymbol{u}_h$, p_h and, as a result, it is coupled to the solution of the stabilized Navier-Stokes equations. In principle, this would double the number of nodal degrees of freedom of the problem. However, in practice, given that the Navier-Stokes problem is non-linear and has to be solved iteratively anyway, the projection problem can be implemented in an staggered way, updating the projections after each non-linear Navier-Stokes iteration.

The problem for the small scales, given by Eqs. (2.31) and (2.32), is not usually

solved. Instead, it is approximated by an expression of the form

$$\rho\partial\boldsymbol{u}_s + \frac{1}{\boldsymbol{\tau}_u}\boldsymbol{u}_s \approx \boldsymbol{R}^m\left(\boldsymbol{u}_h, p_h\right) - \boldsymbol{\xi}_h \qquad \frac{1}{\tau_p}p_s \approx R^c\left(\boldsymbol{u}_h\right) - \delta_h \tag{2.37}$$

where second order tensor $\boldsymbol{\tau}_u$ and the scalar τ_p, known as stabilization parameters, are algorithmic quantities that have to be defined to complete the method.

A motivation for this expression can be found in [28], where the parameters in Eq. (2.37) are designed to ensure that the L^2 norm of the modeled subscale variables is the same as that of the exact small scale values. A different justification, based on the approximation of the Green's function of the small scale problem, is provided in [9, 55].

2.3.3 Quasi-static small scale models

While Eq. (2.37) represents the complete small scale model derived from the dynamic Navier-Stokes equations, many VMS formulations that can be found in the literature (see for example [9, 27, 46]) neglect the time variation of the velocity small scale model and define the small scales as

$$\boldsymbol{u}_s \approx \boldsymbol{\tau}_u\left(\boldsymbol{R}^m\left(\boldsymbol{u}_h, p_h\right) - \boldsymbol{\xi_h}\right) \qquad p_s \approx \tau_p\left(R^c\left(\boldsymbol{u}_h\right) - \delta_h\right) \tag{2.38}$$

This choice corresponds to the assumption that the velocity small scales adapt automatically to the large scale residual. Following the nomenclature of [27], we refer to models based on Eq. (2.38) as *quasi-static subscales*, as opposed to *dynamic subscale* models, based on Eq. (2.37).

To complete the formulation, a definition for the stabilization parameters is needed. We follow the approach of [27], where the velocity stabilization parameter is taken to be a diagonal matrix $\boldsymbol{\tau}_u = \tau_u\boldsymbol{I}$ and

$$\tau_u = \left(\frac{c_1\mu}{h^2} + \frac{c_2\rho\left\|\boldsymbol{a}\right\|}{h}\right)^{-1} \tag{2.39}$$

$$\tau_p = \frac{h^2}{c_1\tau_u} = \mu + \frac{c_2\left\|\boldsymbol{a}\right\|h}{c_1} \tag{2.40}$$

where h is a characteristic length of the element and c_1, c_2 are constants, which, for linear finite elements, are usually defined as $c_1 = 4$, $c_2 = 2$ (this is the case for example in [26] or [29]). However, the studies presented in [7] for a turbulent channel flow in the low Mach number regime suggest that the choice of values for these parameters can have an impact on the solution. Based on the results presented in that reference and in [32], we have adopted $c_1 = 8$, $c_2 = 2$ for our tests.

As pointed out in [30], the use of quasi-static subscales leaves open the possibility of instabilities appearing for small time steps once the problem is discretized in time. The

same instability is studied in [15] for the Stokes problem, where it is shown that it can be neutralized if the stabilization parameter satisfies the condition

$$\delta t \geq C\tau_u \tag{2.41}$$

where δt is the time step and C is a constant. To avoid this instability, the stabilization parameter τ_u can be replaced by the modified expression

$$\tau_t = \left(\frac{\rho}{\delta t} + \frac{c_1 \mu}{h^2} + \frac{c_2 \rho \left\| \boldsymbol{a} \right\|}{h} \right)^{-1} \tag{2.42}$$

although, as remarked in [30], this introduces a dependency of the solution on the time step, even for problems that result in a stationary solution.

2.3.4 Dynamic small-scale models

If, instead of the quasi-static model of Eq. (2.38), we use the dynamic model of (2.37), the time evolution of the velocity small scale $\boldsymbol{u}_s$ has to be taken into account. This was achieved in [28, 30] by introducing a time discretization for the small scale acceleration, resulting in the time-discrete small scale model

$$\rho \frac{\boldsymbol{u}_s^{n+\theta} - \boldsymbol{u}_s^n}{\delta t} + \frac{1}{\tau_u} \boldsymbol{u}_s^{n+\theta} = \left(\boldsymbol{R}^m \left(\boldsymbol{u}_h, p_h \right) - \boldsymbol{\xi}_h \right) \big|_{n+\theta} \tag{2.43}$$

Taking $\theta = 1$, which corresponds to a backward Euler time scheme, we can write a closed expression for the small scale velocity, given by

$$\left(\frac{\rho}{\delta t} + \frac{1}{\tau_u} \right) \boldsymbol{u}_s^{n+1} = \left(\boldsymbol{R}^m \left(\boldsymbol{u}_h, p_h \right) - \boldsymbol{\xi}_h \right) \big|_{n+1} + \frac{\rho}{\delta t} \boldsymbol{u}_s^n \tag{2.44}$$

As remarked in [30], the effective stabilization parameter in Eq. (2.44) is

$$\left(\frac{\rho}{\delta t} + \frac{1}{\tau_u} \right)^{-1} = \left(\frac{\rho}{\delta t} + \frac{c_1 \mu}{h^2} + \frac{c_2 \rho \left\| \boldsymbol{a} \right\|}{h} \right)^{-1} \tag{2.45}$$

which is precisely the expression introduced as τ_t in Eq. (2.42) and prevents the apparition of instabilities due to small time steps. However, unlike in quasi-static approximations, when the problem has a stationary solution, the dependency on the time step is eliminated as, in that case, $\boldsymbol{u}_s^{n+1} = \boldsymbol{u}_s^n$ and the quasi-static model of Eq. (2.38) is recovered.

From the point of view of its implementation, the main difference between the quasi-static model of Eq. (2.38) and Eq. (2.44) is that the latter introduces the old value of the velocity small scale $\boldsymbol{u}_s^n$ in the model. This means that $\boldsymbol{u}_s$ has to be tracked in time. In practice, this implies evaluating and storing historical values for $\boldsymbol{u}_s$ on the integration points of the finite element mesh.

2.3.5 Complete equations

Once the small scale model is defined, it can be introduced in the large scale equations, given by Eqs. (2.22) and (2.23). The most general formulation that can be obtained from the material presented in the previous section is given by

Momentum equation

$$\begin{aligned}
&\int_\Omega \boldsymbol{w}_h \cdot \rho \left(\partial_t \boldsymbol{u}_h + \frac{1}{2}\, \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h \right) \mathrm{d}\Omega - \int_\Omega \nabla \boldsymbol{w}_h : \rho \frac{1}{2} \left(\boldsymbol{a} \otimes \boldsymbol{u}_h \right) \mathrm{d}\Omega \\
&- \int_\Omega \nabla \cdot \boldsymbol{w}_h \, p_h \, \mathrm{d}\Omega + \int_\Omega \nabla^s \boldsymbol{w}_h : 2\mu \left(\nabla^s \boldsymbol{u}_h - \frac{1}{3} \left(\nabla \cdot \boldsymbol{u}_h \right) \boldsymbol{I} \right) \mathrm{d}\Omega \\
&+ \int_\Omega \boldsymbol{w}_h \cdot \rho \partial_t \boldsymbol{u}_s \, \mathrm{d}\Omega - \int_{\Sigma \, \Omega^e} \rho \left(\boldsymbol{a} \cdot \nabla \boldsymbol{w}_h \right) \tau_t \left(\boldsymbol{R}^m \left(\boldsymbol{u}_h, p_h \right) - \boldsymbol{\xi}_h + \frac{\rho}{\delta t} \boldsymbol{u}_s^n \right) \mathrm{d}\Omega \\
&- \int_\Omega \nabla \cdot \boldsymbol{w}_h \tau_p \left(R^c \left(\boldsymbol{u}_h \right) - \delta_h \right) \mathrm{d}\Omega = \int_\Omega \boldsymbol{w}_h \boldsymbol{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}_h \left(\boldsymbol{t} - \rho \frac{1}{2} \left(\boldsymbol{a} \cdot \boldsymbol{n} \right) \boldsymbol{u}_h \right) \mathrm{d}\Gamma
\end{aligned} \tag{2.46}$$

Mass conservation

$$\int_\Omega q_h \nabla \cdot \boldsymbol{u}_h \, \mathrm{d}\Omega = \int_{\Sigma \, \Omega^e} \nabla q_h \, \tau_t \left(\boldsymbol{R}^m \left(\boldsymbol{u}_h, p_h \right) - \boldsymbol{\xi}_h + \frac{\rho}{\delta t} \boldsymbol{u}_s^n \right) \mathrm{d}\Omega \tag{2.47}$$

The first two rows of Eq. (2.46), in combination with its right hand side, constitute the standard Galerkin discretization of the momentum equation. The terms in the third row model the effect of the velocity small scale fluctuations and the velocity small scale itself, respectively, on the large scale equations, while the first term in the last row of Eq. (2.46) represents the effect of pressure small scales.

Analogously, the left hand side of Eq. (2.47) represents the Galerkin weak form of the incompressibility equation, while its right hand side models the effect of the small scales in mass conservation.

While Eqs. (2.46) and (2.47) represent a general VMS formulation, they can be particularized to recover several well-known stabilized methods:

D-ASGS Dynamic algebraic sub-grid scales [30]. The algebraic approximation to the small scales is characterized by using an identity projector to define the small scale space, which implies that the projection terms $\boldsymbol{\xi}_h$ and δ_h are zero.

Q-ASGS Quasi-static algebraic sub-grid scales [27]. A quasi-static approximation to the small scales can be recovered by neglecting all terms involving either the small scale acceleration $\partial_t \boldsymbol{u}_s$ or the old small-scale velocities $\boldsymbol{u}_s^n$ and replacing τ_t by the static stabilization parameter τ_u. Additionally, as the small scales are algebraic, projection terms $\boldsymbol{\xi}_h$ and δ_h are also zero.

D-OSS Dynamic orthogonal subgrid-scales [28, 30]. If the small scale space is assumed to be orthogonal to the large scale space, the integral involving $\boldsymbol{w}_h$ and $\partial_t \boldsymbol{u}_s$ is zero, as it corresponds to the L^2 product of two terms belonging to orthogonal spaces.

Q-OSS Quasi-static orthogonal subgrid-scales [28] can be recovered from the original expression by neglecting all terms involving $\partial_t \boldsymbol{u}_s$ or $\boldsymbol{u}_s^n$ and replacing τ_t by τ_u.

For OSS formulations, the projections $\boldsymbol{\xi}_h$ and δ_h are defined as the L^2 projections of the momentum and mass residuals, respectively, onto the finite element mesh. Applying this definition, they can be obtained as the solution of the projection problem

$$\begin{aligned}\int_\Omega \boldsymbol{w}_h \cdot \boldsymbol{\xi}_h \,\mathrm{d}\Omega &= \int_\Omega \boldsymbol{w}_h \cdot \boldsymbol{R}^m\left(\boldsymbol{u}_h, p_h\right) \mathrm{d}\Omega \\ &= \int_\Omega \boldsymbol{w}_h \left(\boldsymbol{f} - \rho \partial_t \boldsymbol{u}_h - \rho \frac{1}{2}\, \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h - \rho \frac{1}{2} \nabla \cdot \left(\boldsymbol{a} \otimes \boldsymbol{u}_h\right) \right) \mathrm{d}\Omega \qquad (2.48) \\ &\quad + \int_\Omega \boldsymbol{w}_h \left(2\mu \nabla \cdot \left(\nabla^s \boldsymbol{u}_h - \frac{1}{3} \left(\nabla \cdot \boldsymbol{u}_h\right) \boldsymbol{I} \right) - \nabla p_h \right) \mathrm{d}\Omega \\ \int_\Omega q_h \delta_h \,\mathrm{d}\Omega &= \int_\Omega q_h R^c\left(\boldsymbol{u}_h\right) \mathrm{d}\Omega = -\int_\Omega q_h \nabla \cdot \boldsymbol{u}_h \,\mathrm{d}\Omega \qquad (2.49)\end{aligned}$$

The only question that remains to close the formulation is to provide a formal definition for the auxiliary convection velocity $\boldsymbol{a}$, something that we have been deliberately avoiding up to this point. This variable was introduced to linearize the convective term, and in practice we can identify it with the last known value of velocity, which corresponds to a Picard linearization of the momentum equation. However, in a context of scale separation, should we use the large scale velocity $\boldsymbol{u}_h$ or the full velocity $\boldsymbol{u} = \boldsymbol{u}_h + \boldsymbol{u}_s$?

Both choices result in viable stabilized formulations. The classical approach is to use only the large scale velocity $\boldsymbol{u}_h$, as it corresponds to the finite element solution. However, from a theoretical point of view, using the full velocity $\boldsymbol{u}_h + \boldsymbol{u}_s$ has interesting implications for turbulence modeling, which will be the main focus of Section 2.4.

2.4 VMS methods and Large Eddy Simulation

The concept of scale separation introduced by VMS formulations has some parallels with Large Eddy Simulation (LES) methods for the simulation of turbulent flows. LES turbulence models are also based on separating large and small motions in the flow, but in the LES approach this is is typically achieved through the introduction a filtering operation [101], defined as

$$\overline{\boldsymbol{u}}\left(\boldsymbol{x}, t\right) = \int_{-\Delta/2}^{\Delta/2} \int_{-\Delta/2}^{\Delta/2} \int_{-\Delta/2}^{\Delta/2} G\left(\boldsymbol{x} - \boldsymbol{\chi}\right) \boldsymbol{u}\left(\boldsymbol{\chi}, t\right) \,\mathrm{d}\boldsymbol{\chi} \qquad (2.50)$$

where $G(\boldsymbol{x}-\boldsymbol{\chi})$ is a filter function defined on the interval $[-\Delta/2, \Delta/2]$ and Δ is known as the filter width.

Applying a filter function to Eqs. (2.1) and (2.2) allows us to write the filtered Navier-Stokes equations, given by

$$\rho\,\partial_t\overline{\boldsymbol{u}} + \rho\nabla\cdot(\overline{\boldsymbol{u}}\otimes\overline{\boldsymbol{u}}) - \rho\nabla\cdot\boldsymbol{\tau}^R - \nabla\cdot(2\mu\nabla^s\overline{\boldsymbol{u}}) + \nabla\overline{p} = \overline{\boldsymbol{f}} \qquad \text{in } \Omega\times[0,T) \qquad (2.51)$$

$$\nabla\cdot\overline{\boldsymbol{u}} = 0 \qquad \text{in } \Omega\times[0,T) \qquad (2.52)$$

where we have used the conservative form of the convective term and $\boldsymbol{\tau}^R$ is the subgrid stress tensor, defined as

$$\boldsymbol{\tau}^R = \overline{\boldsymbol{u}\otimes\boldsymbol{u}} - \overline{\boldsymbol{u}}\otimes\overline{\boldsymbol{u}} \qquad (2.53)$$

The subgrid stress tensor represents the effect of the small scale motions on the large scale part of the solution and and is an unknown, since the quantity $\overline{\boldsymbol{u}\otimes\boldsymbol{u}}$ cannot be obtained from the filtered velocity $\overline{\boldsymbol{u}}$. This means that Eqs. (2.51) and (2.52) do not represent a closed expression. However, if the filter width is chosen small enough that the filtered-out small scale motions can be assumed to lie in the inertial subrange, Kolmogorov's hypotheses [64, 65] tell us that they have an isotropic, universal (problem-independent) behavior. LES methods use this approach to motivate a model for $\boldsymbol{\tau}^R$ and justify its introduction in the filtered equations, closing the formulation.

Introducing the subgrid velocity $\tilde{\boldsymbol{u}} = \boldsymbol{u} - \overline{\boldsymbol{u}}$, the subgrid stress tensor $\boldsymbol{\tau}^R$ can be rewritten using the Leonard decomposition as

$$\boldsymbol{\tau}^R = \boldsymbol{L} + \boldsymbol{C} + \boldsymbol{R} \qquad (2.54)$$

where each of the individual terms is defined as:

$$\boldsymbol{L} = \overline{\overline{\boldsymbol{u}}\otimes\overline{\boldsymbol{u}}} - \overline{\boldsymbol{u}}\otimes\overline{\boldsymbol{u}} \qquad \text{Leonard stress} \qquad (2.55)$$

$$\boldsymbol{C} = \overline{\tilde{\boldsymbol{u}}\otimes\overline{\boldsymbol{u}}} + \overline{\overline{\boldsymbol{u}}\otimes\tilde{\boldsymbol{u}}} \qquad \text{Cross stress} \qquad (2.56)$$

$$\boldsymbol{R} = \overline{\tilde{\boldsymbol{u}}\otimes\tilde{\boldsymbol{u}}} \qquad \text{Reynolds stress} \qquad (2.57)$$

The different terms in Eqs. (2.55)–(2.57) represent subgrid stresses due to the interaction between resolved motions (Leonard stresses), to the interaction between large and unresolved motions (cross stresses) and to the effect of completely unresolved motions (Reynolds stresses).

Like filtering in LES methods, scale separation in VMS formulations introduces a clear division between the resolved and unresolved parts of the solution, which in this case is achieved through the L^2 projection to the finite element mesh. This projection to the mesh was introduced in writing the large scale equation, given by Eqs. (2.18) and (2.19), which is rewritten using the conservative form of the convective term to be

consistent with the LES expression above, obtaining

$$\begin{aligned}\int_\Omega \boldsymbol{w}_h \cdot \rho\left(\partial_t \boldsymbol{u}_h + \partial_t \boldsymbol{u}_s\right)\,\mathrm{d}\Omega - \int_\Omega \rho\nabla\boldsymbol{w}_h : (\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes(\boldsymbol{u}_h+\boldsymbol{u}_s)\,\mathrm{d}\Omega \\ + \int_\Omega \nabla^s \boldsymbol{w}_h : 2\mu\nabla^s\left(\boldsymbol{u}_h+\boldsymbol{u}_s\right)\,\mathrm{d}\Omega - \int_\Omega \nabla\cdot\boldsymbol{w}_h\left(p_h+p_s\right)\,\mathrm{d}\Omega = \int_\Omega \boldsymbol{w}_h\cdot\boldsymbol{f}\,\mathrm{d}\Omega\end{aligned} \tag{2.58}$$

$$\int_\Omega q_h \nabla\cdot\left(\boldsymbol{u}_h+\boldsymbol{u}_s\right)\,\mathrm{d}\Omega = 0 \tag{2.59}$$

where boundary terms and terms related to the trace of the viscous stresses have been omitted for clarity and the full velocity $\boldsymbol{u}_h + \boldsymbol{u}_s$ has been used for both arguments of the convective term.

An analogy can be established between Eqs. (2.51) and (2.52), which represent the filtered Navier-Stokes equations and Eqs. (2.58) and (2.59), which expresses the projection of the Navier-Stokes equations to the finite element mesh. This was analyzed in [30, 58], where it is remarked that the convective term in Eq. (2.58) can be expanded as

$$\begin{aligned}\int_\Omega \rho\nabla\boldsymbol{w}_h : (\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes(\boldsymbol{u}_h+\boldsymbol{u}_s)\,\mathrm{d}\Omega = \int_\Omega \rho\nabla\boldsymbol{w}_h : \boldsymbol{u}_h\otimes\boldsymbol{u}_h\,\mathrm{d}\Omega \\ + \int_\Omega \rho\nabla\boldsymbol{w}_h : \boldsymbol{u}_s\otimes\boldsymbol{u}_h\,\mathrm{d}\Omega + \int_\Omega \rho\nabla\boldsymbol{w}_h : \boldsymbol{u}_h\otimes\boldsymbol{u}_s\,\mathrm{d}\Omega + \int_\Omega \rho\nabla\boldsymbol{w}_h : \boldsymbol{u}_s\otimes\boldsymbol{u}_s\,\mathrm{d}\Omega\end{aligned} \tag{2.60}$$

The last three terms in Eq. (2.60) can be understood as a variational version of the LES subgrid stress tensor $\boldsymbol{\tau}^R$. Ignoring the density, they can be rearranged as

$$\begin{aligned}&\int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{u}_s\otimes\boldsymbol{u}_h\,\mathrm{d}\Omega + \int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{u}_h\otimes\boldsymbol{u}_s\,\mathrm{d}\Omega + \int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{u}_s\otimes\boldsymbol{u}_s\,\mathrm{d}\Omega \\ &= \int_\Omega \nabla\boldsymbol{w}_h : (\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes(\boldsymbol{u}_h+\boldsymbol{u}_s)\,\mathrm{d}\Omega - \int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{u}_h\otimes\boldsymbol{u}_h\,\mathrm{d}\Omega \\ &= \int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{\tau}^R_{VMS}\,\mathrm{d}\Omega\end{aligned} \tag{2.61}$$

where we introduced $\boldsymbol{\tau}^R_{VMS}$ as a variational analogue of the LES subgrid stress tensor.

Furthermore, $\boldsymbol{\tau}^R_{VMS}$ can be decomposed into Cross and Reynolds terms as:

$$\int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{u}_s\otimes\boldsymbol{u}_h\,\mathrm{d}\Omega + \int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{u}_h\otimes\boldsymbol{u}_s\,\mathrm{d}\Omega \qquad \text{Cross stress} \tag{2.62}$$

$$\int_\Omega \nabla\boldsymbol{w}_h : \boldsymbol{u}_s\otimes\boldsymbol{u}_s\,\mathrm{d}\Omega \qquad \text{Reynolds stress} \tag{2.63}$$

while an analogue of the Leonard stress, representing the contribution of the resolved

velocities $\boldsymbol{u}_h$ to the unresolved stresses, appears in the corresponding small scale equation

$$\begin{aligned}\int_\Omega \nabla \boldsymbol{w}_h : \boldsymbol{u}_h \otimes \boldsymbol{u}_h \,\mathrm{d}\Omega - \int_\Omega \nabla \boldsymbol{w} : \boldsymbol{u}_h \otimes \boldsymbol{u}_h \,\mathrm{d}\Omega = \\ - \int_\Omega \nabla \boldsymbol{w}_s : \boldsymbol{u}_h \otimes \boldsymbol{u}_h \,\mathrm{d}\Omega \qquad\qquad \text{Leonard stress}\end{aligned} \tag{2.64}$$

Assuming that the grid size is small enough for the unresolved scales to be in the inertial subrange, we can observe that scale separation and projection to the finite element mesh play a similar role to that of filtering in classical LES methods. It is worth mentioning that the concept of mesh-induced filtering has also been explored by the LES community, where it is known as *implicit filtering*. This is the basis of the Monotone Integrated Large-Eddy Simulation (MILES) approach of Boris *et al.* [16], where the authors propose not using an explicit model for the subgrid stresses and relying instead on a specially designed numerical method to introduce the correct dissipation for a given mesh resolution.

A different approach to VMS-based LES, which will only be mentioned here, is based on a three–level scale separation (see for example [45, 46] or the review of [44]). In such approaches, the small scales are in turn divided into *resolved* and *unresolved* small scales. The presence of two levels of small scales can be used to either introduce explicit LES model terms to represent the effect of the unresolved small scales on the problem or to calibrate the amount of dissipation that is introduced using the variational equivalent of the Germano identity [83, 84].

2.4.1 The VMS kinetic energy balance

Given that the derivation of VMS formulations is based exclusively in numerical and mathematical arguments, the physical behavior of VMS methods, in terms of reproducing the expected dissipation rates when the small scales are on the inertial subrange, has to be verified. This topic has been studied in [58, 59] for the formulation that we are calling Q-ASGS and in [47, 102] for OSS-based methods.

An energy balance for the original Navier-Stokes problem can be obtained by taking $\boldsymbol{w} = \boldsymbol{u}$ in the Galerkin weak form of the momentum equation, given by Eq. (2.11), which produces

$$\begin{aligned}\int_\Omega \boldsymbol{u} \cdot \rho \partial_t \boldsymbol{u} \,\mathrm{d}\Omega + \int_\Omega \boldsymbol{u} \cdot \left(\rho\, \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) \mathrm{d}\Omega + \int_\Omega 2\mu \nabla^s \boldsymbol{u} : \nabla^s \boldsymbol{u} \,\mathrm{d}\Omega \\ - \int_\Omega \nabla \cdot \boldsymbol{u} p \,\mathrm{d}\Omega = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{f} \,\mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w} \cdot \boldsymbol{t} \,\mathrm{d}\Gamma\end{aligned} \tag{2.65}$$

We are interested in obtaining a balance for the kinetic energy, $E = \rho \boldsymbol{u} \cdot \boldsymbol{u}/2$. Using the fact that the full velocity is incompressible and the equality $u\, \partial u/\partial x = 1/2\, \partial u^2/\partial x$,

Eq. (2.65) can be expressed as

$$\underbrace{\int_\Omega \partial_t E \,\mathrm{d}\Omega}_{I} + \underbrace{\int_\Omega \boldsymbol{u}\cdot\nabla E \,\mathrm{d}\Omega}_{II} = \underbrace{\int_\Omega \boldsymbol{u}\cdot\boldsymbol{f} \,\mathrm{d}\Omega}_{III} + \underbrace{\int_{\Gamma_N} \boldsymbol{u}\cdot\boldsymbol{t} \,\mathrm{d}\Gamma}_{IV} - \underbrace{\int_\Omega 2\mu\nabla^s\boldsymbol{u} : \nabla^s\boldsymbol{u} \,\mathrm{d}\Omega}_{V} \quad (2.66)$$

Eq. (2.66) expresses the balance of total kinetic energy E in the domain, and the individual terms represent energy storage (I) and convection (II), the power exerted by external forces (III) and boundary tensions (IV) and finally viscous dissipation (V).

We can obtain an equivalent expression for the kinetic energy contained in the large scale motions, $E_h = \rho\boldsymbol{u}_h\cdot\boldsymbol{u}_h/2$, by taking $\boldsymbol{w}_h = \boldsymbol{u}_h$ in Eq. (2.58), $q_h = p_h$ in Eq. (2.59) and adding the two equations, obtaining:

$$\begin{aligned}\int_\Omega \boldsymbol{u}_h\cdot\rho\left(\partial_t\boldsymbol{u}_h + \partial_t\boldsymbol{u}_s\right)\,\mathrm{d}\Omega + \int_\Omega \boldsymbol{u}_h\cdot\rho\nabla\cdot\left((\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes(\boldsymbol{u}_h+\boldsymbol{u}_s)\right)\,\mathrm{d}\Omega \\ + \int_\Omega \nabla^s\boldsymbol{u}_h : 2\mu\nabla^s\left(\boldsymbol{u}_h+\boldsymbol{u}_s\right)\,\mathrm{d}\Omega - \int_\Omega \nabla\cdot\boldsymbol{u}_h\left(p_h+p_s\right)\,\mathrm{d}\Omega \\ + \int_\Omega p_h\nabla\cdot\left(\boldsymbol{u}_h+\boldsymbol{u}_s\right)\,\mathrm{d}\Omega = \int_\Omega \boldsymbol{u}_h\cdot\boldsymbol{f}\,\mathrm{d}\Omega\end{aligned} \quad (2.67)$$

Note that in Eq. (2.67), and for the remainder of this section, we neglected all boundary integrals to simplify the discussion. This is equivalent to considering a problem with homogeneous Dirichlet boundary conditions.

It is convenient to integrate some of the terms in Eq. (2.67) by parts and rearrange the convective term as follows:

$$\begin{aligned}&\int_\Omega \boldsymbol{u}_h\cdot\rho\nabla\cdot\left((\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes(\boldsymbol{u}_h+\boldsymbol{u}_s)\right)\,\mathrm{d}\Omega = \\ &\quad= \int_\Omega \boldsymbol{u}_h\cdot\rho\nabla\cdot\left((\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes\boldsymbol{u}_h\right)\,\mathrm{d}\Omega + \int_\Omega \boldsymbol{u}_h\cdot\rho\nabla\cdot\left((\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes\boldsymbol{u}_s\right)\,\mathrm{d}\Omega \\ &\quad\quad= \int_\Omega \boldsymbol{u}_h\cdot\rho\left(\boldsymbol{u}_h+\boldsymbol{u}_s\right)\cdot\nabla\boldsymbol{u}_h\,\mathrm{d}\Omega - \int_\Omega \rho\nabla\boldsymbol{u}_h : (\boldsymbol{u}_h+\boldsymbol{u}_s)\otimes\boldsymbol{u}_s\,\mathrm{d}\Omega \\ &\quad\quad\quad= \int_\Omega (\boldsymbol{u}_h+\boldsymbol{u}_s)\cdot\nabla E_h\,\mathrm{d}\Omega - \int_\Omega \boldsymbol{u}_s\rho\left(\boldsymbol{u}_h+\boldsymbol{u}_s\right)\cdot\nabla\boldsymbol{u}_h\,\mathrm{d}\Omega\end{aligned}$$

where we have used the fact that the exact velocity $\boldsymbol{u}_h + \boldsymbol{u}_s$ is divergence free.

With this, Eq. (2.67) can be restated as

$$\begin{aligned}&\underbrace{\int_\Omega \partial_t E_h\,\mathrm{d}\Omega}_{I} + \underbrace{\int_\Omega \boldsymbol{u}\cdot\nabla E_h\,\mathrm{d}\Omega}_{II} = \underbrace{\int_\Omega \boldsymbol{u}_h\cdot\boldsymbol{f}\,\mathrm{d}\Omega}_{III} - \underbrace{\int_\Omega 2\mu\nabla^s\boldsymbol{u}_h : \nabla^s\boldsymbol{u}_h\,\mathrm{d}\Omega}_{IV} + \underbrace{\int_\Omega p_s\nabla\cdot\boldsymbol{u}_h\,\mathrm{d}\Omega}_{V} \\ &\qquad\underbrace{- \int_\Omega \boldsymbol{u}_h\cdot\rho\partial_t\boldsymbol{u}_s\,\mathrm{d}\Omega + \int_\Omega \boldsymbol{u}_s\cdot\left(\rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}_h + \nabla p_h + \nabla\cdot\left(2\mu\nabla^s\boldsymbol{u}_h\right)\right)\,\mathrm{d}\Omega}_{VI}\end{aligned} \quad (2.68)$$

Analogously to the complete energy balance, Eq. (2.68) can be understood as a balance for the kinetic energy associated to large scale motions. Terms I and II represent the storage and convection of large scale kinetic energy, while term III represents the power exerted by the external forces on large scale motions. Term IV is the viscous dissipation associated to the large scale motions, which can be assumed to be negligible for high Reynolds numbers. The remaining terms represent the transfer of energy between large scale and residual motions, playing an analogous role to production terms in the filtered Navier-Stokes equations.

Finally, we define the residual kinetic energy as $k_r = E - E_h$. A balance statement for k_r can be obtained by subtracting Eq. (2.68) from Eq. (2.66), which results in

$$\begin{aligned} \underbrace{\int_\Omega \partial_t k_r \,\mathrm{d}\Omega}_{I} + \underbrace{\int_\Omega \boldsymbol{u}\cdot\nabla k_r \,\mathrm{d}\Omega}_{II} &= \underbrace{\int_\Omega \boldsymbol{u}_s\cdot\boldsymbol{f}\,\mathrm{d}\Omega}_{III} \\ \underbrace{-\int_\Omega 2\mu\nabla^s\boldsymbol{u}:\nabla^s\boldsymbol{u}\,\mathrm{d}\Omega}_{IV} + \underbrace{\int_\Omega 2\mu\nabla^s\boldsymbol{u}_h:\nabla^s\boldsymbol{u}_h\,\mathrm{d}\Omega}_{V} - \underbrace{\int_\Omega p_s\nabla\cdot\boldsymbol{u}_h\,\mathrm{d}\Omega}_{VI} & \\ \underbrace{+\int_\Omega \boldsymbol{u}_h\cdot\rho\partial_t\boldsymbol{u}_s\,\mathrm{d}\Omega - \int_\Omega \boldsymbol{u}_s\cdot\left(\rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}_h+\nabla p_h+\nabla\cdot(2\mu\nabla^s\boldsymbol{u}_h)\right)\mathrm{d}\Omega}_{VII} & \end{aligned} \tag{2.69}$$

In Eq. (2.69), terms I and II represent the storage and convection of residual kinetic energy, while term III represents the power exerted by the external forces on small scale (high wavenumber) motions. The next two terms represent the difference between the total viscous dissipation (IV) and the large scale viscous dissipation (V), which was already accounted for in the large scale energy balance of Eq. (2.68). Again, we remark that, in practice, term V is expected to be negligible in comparison to term IV, since viscous dissipation occurs predominantly for motions in the range of the Kolmogorov length scale, while $\boldsymbol{u}_h$ will contain only motions on a much larger scale h, lying on the inertial subrange. Finally, terms VI and VII represent the production of residual energy due to the pressure and velocity small scales, respectively, and are exactly the same (but now with opposite sign) as the production terms in Eq. (2.68).

Terms VI and VII can be modified by noting that, since $p_s \in Q_s$ and $\boldsymbol{u}_s \in V_s$,

$$\int_\Omega p_s\nabla\cdot\boldsymbol{u}_h\,\mathrm{d}\Omega = \int_\Omega p_s\Pi_{Q_s}\left(\nabla\cdot\boldsymbol{u}_h\right)\,\mathrm{d}\Omega \tag{2.70}$$

$$\begin{aligned} \int_\Omega \boldsymbol{u}_s\cdot\left(\rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}_h+\nabla p_h+\nabla\cdot(2\mu\nabla^s\boldsymbol{u}_h)\right)\,\mathrm{d}\Omega = & \\ \int_\Omega \boldsymbol{u}_s\cdot\Pi_{V_s}\big(\rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}_h+\nabla p_h+\nabla\cdot(2\mu\nabla^s\boldsymbol{u}_h)\big)\,\mathrm{d}\Omega & \end{aligned} \tag{2.71}$$

The last step to obtain the residual kinetic energy balance is to introduce the small scale models for velocity and pressure in Eqs. (2.70) and (2.71). Noting that $(2\mu\nabla^s\boldsymbol{u}_h)$

represents the large scale viscous dissipation and will be negligible for high Reynolds numbers, we can write

$$\begin{aligned}
&\int_\Omega \boldsymbol{u}_s \cdot \Pi_{V_s}\left(\rho \boldsymbol{u}\cdot\nabla \boldsymbol{u}_h + \nabla p_h\right)\,\mathrm{d}\Omega + \int_\Omega p_s \Pi_{Q_s}\left(\nabla\cdot\boldsymbol{u}_h\right)\,\mathrm{d}\Omega = \\
&\underbrace{\int_\Omega \tau_u \Pi_{V_s}\left(\boldsymbol{f}\right)\cdot\Pi_{V_s}\left(\rho \boldsymbol{u}\cdot\nabla \boldsymbol{u}_h + \nabla p_h\right)\,\mathrm{d}\Omega}_{I} - \underbrace{\int_\Omega \tau_u \left|\Pi_{V_s}\left(\rho \boldsymbol{u}\cdot\nabla \boldsymbol{u}_h + \nabla p_h\right)\right|^2\,\mathrm{d}\Omega}_{II} \\
&+ \underbrace{\int_\Omega \tau_u \rho\partial_t \boldsymbol{u}_s\cdot\Pi_{V_s}\left(\rho \boldsymbol{u}\cdot\nabla \boldsymbol{u}_h + \nabla p_h\right)\,\mathrm{d}\Omega}_{III} - \underbrace{\int_\Omega \tau_p \left|\Pi_{Q_s}\left(\nabla\cdot\boldsymbol{u}_h\right)\right|^2\,\mathrm{d}\Omega}_{IV}
\end{aligned} \tag{2.72}$$

It is clear that terms II and IV, since the stabilization parameters τ_u and τ_p were defined as strictly positive. They play the role of energy sinks in the large scale energy balance of Eq. (2.68), while acting as sources on the residual energy equation Eq. (2.69). Term I is problem dependent but, if an OSS small scale model is used, it will vanish unless the external forces have a high-frequency (small scale) component. Finally, term III only exists for dynamic small scale models. This fact was used in [102] to justify that OSS formulations can account for backscatter (energy transfer from the small scales to the large ones) if a dynamic small scale model is used.

Starting from a similar reasoning, Guasch and Codina [47] use statistical and scaling arguments to show that OSS formulations extract energy from the large scale equations at the correct rate, provided that some constraints on the behavior of stabilization parameters are respected.

As a final remark on this topic, we compare the large scale energy balance to the filtered energy balance used in filter-based LES formulations. If the kinetic energy associated to the filtered velocity field is defined as $\overline{E} = \rho\overline{\boldsymbol{u}}\cdot\overline{\boldsymbol{u}}/2$, the balance for $\overline{E}$ can be obtained by multiplying the filtered linear momentum equation Eq. (2.51) by $\overline{\boldsymbol{u}}$ and integrating over the fluid domain (see [101]), resulting in

$$\int_\Omega \partial_t \overline{E}\,\mathrm{d}\Omega + \int_\Omega \overline{\boldsymbol{u}}\cdot\nabla\overline{E}\,\mathrm{d}\Omega = \int_\Omega \overline{\boldsymbol{u}}\cdot\overline{\boldsymbol{f}}\,\mathrm{d}\Omega - \int_\Omega 2\mu\nabla^s\overline{\boldsymbol{u}} : \nabla^s\overline{\boldsymbol{u}}\,\mathrm{d}\Omega + \int_\Omega \nabla^s\overline{\boldsymbol{u}} : \boldsymbol{\tau}^R\,\mathrm{d}\Omega \tag{2.73}$$

where boundary fluxes have been omitted.

We can introduce the definition of the VMS subgrid stress tensor $\boldsymbol{\tau}^R_{VMS}$ in Eq. (2.67), obtaining

$$\begin{aligned}
&\int_\Omega \boldsymbol{u}_h\cdot\rho\left(\partial_t\boldsymbol{u}_h + \partial_t\boldsymbol{u}_s\right)\,\mathrm{d}\Omega + \int_\Omega \rho\boldsymbol{u}_h\nabla\cdot\left(\boldsymbol{u}_h\otimes\boldsymbol{u}_h\right)\,\mathrm{d}\Omega \\
&\quad - \int_\Omega \rho\nabla\boldsymbol{u}_h : \boldsymbol{\tau}^R_{VMS}\,\mathrm{d}\Omega + \int_\Omega \nabla^s\boldsymbol{u}_h : 2\mu\nabla^s\left(\boldsymbol{u}_h + \boldsymbol{u}_s\right)\,\mathrm{d}\Omega \\
&\qquad - \int_\Omega \nabla\cdot\boldsymbol{u}_h\left(p_h + p_s\right) + \int_\Omega p_h\nabla\cdot\left(\boldsymbol{u}_h + \boldsymbol{u}_s\right) = \int_\Omega \boldsymbol{u}_h\cdot\boldsymbol{f}\,\mathrm{d}\Omega
\end{aligned} \tag{2.74}$$

where the conservative form of the convective term has been used. Rearranging some terms, Eq. (2.74) can be rewritten as

$$\underbrace{\int_\Omega \partial_t E_h \,\mathrm{d}\Omega}_{I} + \underbrace{\int_\Omega \boldsymbol{u}_h \cdot \nabla E_h \,\mathrm{d}\Omega}_{II} = \underbrace{\int_\Omega \boldsymbol{u}_h \cdot \boldsymbol{f} \,\mathrm{d}\Omega}_{III} - \underbrace{\int_\Omega 2\mu \nabla^s \boldsymbol{u}_h : \nabla^s \boldsymbol{u}_h \,\mathrm{d}\Omega}_{IV} + \underbrace{\int_\Omega p_s \nabla \cdot \boldsymbol{u}_h \,\mathrm{d}\Omega}_{V}$$
$$\underbrace{- \int_\Omega \boldsymbol{u}_h \cdot \rho \partial_t \boldsymbol{u}_s \,\mathrm{d}\Omega - \int_\Omega \boldsymbol{u}_s \cdot \left(\nabla p_h - \nabla \cdot (2\mu \nabla^s \boldsymbol{u}_h)\right) \,\mathrm{d}\Omega}_{VI} + \underbrace{\int_\Omega \rho \nabla^s \boldsymbol{u}_h : \boldsymbol{\tau}^R_{VMS} \,\mathrm{d}\Omega}_{VII} \tag{2.75}$$

where we have used the fact that $\boldsymbol{\tau}^R_{VMS}$ is a symmetric tensor to write $\nabla \boldsymbol{u}_h : \boldsymbol{\tau}^R_{VMS} = \nabla^s \boldsymbol{u}_h : \boldsymbol{\tau}^R_{VMS}$. Terms I to VI have an analogous interpretation to their counterparts in Eq. (2.68), but here we have obtained an additional term, VII, which explicitly represents the contribution of the residual subgrid stresses in the energy transfer. Comparing Eq. (2.75) to Eq. (2.73), we see that both LES and VMS approaches extract energy from the large scale problem through subgrid stresses, but that the variational approach gives rise to two additional energy transfer mechanisms, represented by terms V and VI, compared to filter-based LES.

2.5 Discrete problem

To obtain a finite element solver based on the VMS formulation introduced in Section 2.3 we need to discretize the simulation domain, both in space and in time, and linearize the problem to obtain a system of equations that can be inverted using a linear solver. We start by introducing a finite element partition Ω_h for the problem domain Ω. Given the discrete domain Ω_h, the large scale interpolation spaces V_h and Q_h can be identified with the standard finite element interpolation functions and the large scale part of the solution, $\boldsymbol{u}_h$ and p_h, can be represented using a finite element interpolation as

$$\boldsymbol{u}_h = \sum_a^{n_n} \boldsymbol{N}_a(\boldsymbol{x})\, \boldsymbol{u}_a \qquad p_h = \sum_a^{n_n} N_a(\boldsymbol{x})\, p_a \tag{2.76}$$

where n_n represents the number of nodes in the finite element mesh, $\boldsymbol{u}_a$ and p_a are the nodal values of the large scale variables $\boldsymbol{u}_h$ and p_h respectively, N_a represents the standard finite element basis function associated to node a and $\boldsymbol{N}_a$ its counterpart for vectorial variables, given by

$$\boldsymbol{N}_a = \begin{bmatrix} N_a & 0 & 0 \\ 0 & N_a & 0 \\ 0 & 0 & N_a \end{bmatrix}$$

Additionally, we introduce the following notation for the gradient and divergence of the finite element shape functions, which will be used to write the discrete form of the

differential operators involved in the problem

$$(\nabla N_a)^T = \begin{bmatrix} \frac{\partial N_a}{\partial x} & \frac{\partial N_a}{\partial y} & \frac{\partial N_a}{\partial z} \end{bmatrix} \qquad \nabla \cdot \boldsymbol{N}_a = \begin{bmatrix} \frac{\partial N_a}{\partial x} & \frac{\partial N_a}{\partial y} & \frac{\partial N_a}{\partial z} \end{bmatrix}$$

$$(\nabla \boldsymbol{N}_a)^T = \begin{bmatrix} \frac{\partial N_a}{\partial x} & 0 & 0 & \frac{\partial N_a}{\partial x} & 0 & 0 & \frac{\partial N_a}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_a}{\partial y} & 0 & 0 & \frac{\partial N_a}{\partial y} & 0 & 0 & \frac{\partial N_a}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_a}{\partial z} & 0 & 0 & \frac{\partial N_a}{\partial z} & 0 & 0 & \frac{\partial N_a}{\partial z} \end{bmatrix} \tag{2.77}$$

We also introduce the following operator to describe convection

$$\boldsymbol{a} \cdot \nabla \boldsymbol{N}_a = \begin{bmatrix} \boldsymbol{a} \cdot \nabla N_a & 0 & 0 \\ 0 & \boldsymbol{a} \cdot \nabla N_a & 0 \\ 0 & 0 & \boldsymbol{a} \cdot \nabla N_a \end{bmatrix} \tag{2.78}$$

and the strain rate matrix, which will be used to write the discrete version of the viscous term

$$\boldsymbol{B}_a^T = \begin{bmatrix} \frac{\partial N_a}{\partial x} & 0 & 0 & \frac{\partial N_a}{\partial y} & 0 & \frac{\partial N_a}{\partial z} \\ 0 & \frac{\partial N_a}{\partial y} & 0 & \frac{\partial N_a}{\partial x} & \frac{\partial N_a}{\partial z} & 0 \\ 0 & 0 & \frac{\partial N_a}{\partial z} & 0 & \frac{\partial N_a}{\partial y} & \frac{\partial N_a}{\partial x} \end{bmatrix} \tag{2.79}$$

Additionally, we define $\boldsymbol{U}$, $\dot{\boldsymbol{U}}$ and $\boldsymbol{P}$ as the vectors of nodal values of large scale velocity $\boldsymbol{u}_h$, acceleration $\partial_t \boldsymbol{u}_h$ and pressure p_h, respectively.

We introduce the finite element discretization of Eq. (2.76) in the variational formulation given by Eqs. (2.46) and (2.47) to obtain the matrix form of the problem. We will analyze the resulting expression for each of the variants we are considering in turn.

Note that in the present work we use linear finite elements, which can not be used to write second derivatives of the variables or test functions. As a result, terms involving $\nabla \cdot \nabla^s \boldsymbol{w}_h$ in Eq. (2.46) or the strong-form viscous term that appears in the residual $\boldsymbol{R}^m(\boldsymbol{u}_h, p_h)$ introduced in Eq. (2.33) will be neglected in the discrete form. It must be remarked that all terms lost in this way are related to viscous stresses, which should be small in turbulent flow problems.

2.5.1 Quasi-static ASGS formulation

The quasi-static ASGS formulation is in some sense the *classical* VMS formulation for the Navier-Stokes equations. It is also relatively simple, as it does not involve dynamic terms or projections, so we will present it first.

Starting from Eqs. (2.46) and (2.47), we can neglect all terms involving the projections $\boldsymbol{\xi}_h$ or δ_h. As the small scales are described using Eq. (2.38), terms involving $\boldsymbol{u}_s^n$ or $\partial_t \boldsymbol{u}_s$ in these equations can be ignored and the momentum stabilization parameter is τ_u, given by Eq. (2.39). After these simplifications, we introduce the finite element interpolation of Eq. (2.76) to describe the problem variables $\boldsymbol{u}_h$, p_h. Testing against each nodal basis function in turn we obtain a system of equations that can be expressed in matrix form as

$$\begin{aligned}\big[\boldsymbol{M} + \boldsymbol{S}_m(\tau_u, \boldsymbol{a})\big]\dot{\boldsymbol{U}} + \big[\boldsymbol{C}(\boldsymbol{a}) + \boldsymbol{K} + \boldsymbol{S}_u(\tau_u, \boldsymbol{a}) + \boldsymbol{H}_u(\tau_p)\big]\boldsymbol{U} \\ + \big[\boldsymbol{G} + \boldsymbol{S}_p(\tau_u, \boldsymbol{a})\big]\boldsymbol{P} = \boldsymbol{F} + \boldsymbol{T} + \boldsymbol{S}_f(\tau_u, \boldsymbol{a})\end{aligned} \tag{2.80}$$

$$\boldsymbol{Q}_m(\tau_u)\dot{\boldsymbol{U}} + \big[\boldsymbol{D} + \boldsymbol{Q}_u(\tau_u, \boldsymbol{a})\big]\boldsymbol{U} + \boldsymbol{Q}_p(\tau_u)\boldsymbol{P} = \boldsymbol{Q}_f(\tau_u) \tag{2.81}$$

Again, we have used a generic convection velocity $\boldsymbol{a}$ in all terms that have a nonlinear dependence of velocity. Doing so, we leave open the possibility of using either the full velocity $\boldsymbol{a} = \boldsymbol{u}_h + \boldsymbol{u}_s$ or the only large scale part $\boldsymbol{a} = \boldsymbol{u}_h$. Note that, for linear finite elements, the latter choice is equivalent to the Galerkin-Least Squares (GLS) method [56].

The different matrices in Eqs. (2.80) and (2.81) represent the discrete version of the operators in Eqs. (2.46) and (2.47) and can be built from the assembly of elemental contributions. In general, matrix $\boldsymbol{A}$ is constructed by the finite element assembly of elemental matrices of the form $\boldsymbol{A}^e$. For a finite element with N nodes, $\boldsymbol{A}^e$ can be defined using $N \times N$ blocks $\boldsymbol{A}^e_{ab}$, where a and b are local node indices. Using this notation, the standard Galerkin terms in the variational form of the problem give rise to the following elemental matrices

$$\boldsymbol{M}^e_{ab} = \int_{\Omega_e} \rho \boldsymbol{N}_a^T \boldsymbol{N}_b \,\mathrm{d}\Omega \tag{2.82}$$

$$\boldsymbol{C}(\boldsymbol{a})^e_{ab} = \int_{\Omega_e} \rho \frac{1}{2}\left(\boldsymbol{N}_a^T \boldsymbol{a}\cdot\nabla\boldsymbol{N}_b - (\boldsymbol{a}\cdot\nabla\boldsymbol{N}_a)^T \boldsymbol{N}_b\right)\mathrm{d}\Omega \tag{2.83}$$

$$+ \int_{\Gamma_N} \boldsymbol{N}_a^T \rho \frac{1}{2}(\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{N}_b \,\mathrm{d}\Gamma \tag{2.84}$$

$$\boldsymbol{K}^e_{ab} = \int_{\Omega_e} \boldsymbol{B}_a^T \boldsymbol{C}_\mu \boldsymbol{B}_b \,\mathrm{d}\Omega \tag{2.85}$$

$$\boldsymbol{G}^e_{ab} = -\int_{\Omega_e} (\nabla\cdot\boldsymbol{N}_a)^T N_b \,\mathrm{d}\Omega \tag{2.86}$$

$$\boldsymbol{D}^e_{ab} = \int_{\Omega_e} N_a \nabla\cdot\boldsymbol{N}_b \,\mathrm{d}\Omega = -\big(\boldsymbol{G}^e_{ba}\big)^T \tag{2.87}$$

$$\boldsymbol{F}^e_a = \int_{\Omega_e} \boldsymbol{N}_a^T \boldsymbol{f} \,\mathrm{d}\Omega \tag{2.88}$$

$$\boldsymbol{T}^e_a = \int_{\Gamma_N} \boldsymbol{N}_a^T \boldsymbol{t} \,\mathrm{d}\Gamma \tag{2.89}$$

When defining the viscous matrix $\boldsymbol{K}_{ab}^{e}$ in Eq. (2.85) we introduced the constitutive matrix $\boldsymbol{C}_{\mu}$, which is given by

$$\boldsymbol{C}_{\mu} = \begin{bmatrix} 4\mu/3 & -2\mu/3 & -2\mu/3 & 0 & 0 & 0 \\ -2\mu/3 & 4\mu/3 & -2\mu/3 & 0 & 0 & 0 \\ -2\mu/3 & -2\mu/3 & 4\mu/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{2.90}$$

In the same way, the stabilization terms in the Q-ASGS give rise to additional elemental matrices, expressed here as

$$\boldsymbol{S}_{m}\left(\tau_{u},\boldsymbol{a}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\boldsymbol{N}_{b}\,\mathrm{d}\Omega \tag{2.91}$$

$$\boldsymbol{S}_{u}\left(\tau_{u},\boldsymbol{a}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_{b}\,\mathrm{d}\Omega \tag{2.92}$$

$$\boldsymbol{S}_{p}\left(\tau_{u},\boldsymbol{a}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\nabla N_{b}\,\mathrm{d}\Omega \tag{2.93}$$

$$\boldsymbol{S}_{f}\left(\tau_{u},\boldsymbol{a}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\boldsymbol{f}\,\mathrm{d}\Omega \tag{2.94}$$

$$\boldsymbol{Q}_{m}\left(\tau_{u}\right)_{ab}^{e} = \int_{\Omega} \left(\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\boldsymbol{N}_{b}\,\mathrm{d}\Omega \tag{2.95}$$

$$\boldsymbol{Q}_{u}\left(\tau_{u},\boldsymbol{a}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_{b}\,\mathrm{d}\Omega = \left(\boldsymbol{S}_{p}\left(\tau_{u},\boldsymbol{a}\right)_{ba}^{e}\right)^{T} \tag{2.96}$$

$$\boldsymbol{Q}_{p}\left(\tau_{u}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\nabla\boldsymbol{N}_{b}\,\mathrm{d}\Omega \tag{2.97}$$

$$\boldsymbol{Q}_{f}\left(\tau_{u}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\nabla\boldsymbol{N}_{a}\right)^{T}\tau_{u}\,\boldsymbol{f}\,\mathrm{d}\Omega \tag{2.98}$$

$$\boldsymbol{H}_{u}\left(\tau_{p}\right)_{ab}^{e} = \int_{\Omega_{e}} \left(\nabla\cdot\boldsymbol{N}_{a}\right)^{T}\tau_{p}\,\nabla\cdot\boldsymbol{N}_{b}\,\mathrm{d}\Omega \tag{2.99}$$

2.5.2 Quasi-static OSS formulation

The next variant to be presented is the quasi-static OSS formulation. Compared to the Q-ASGS formulation, OSS is characterized by the inclusion of the projections $\boldsymbol{\xi}_{h}$ and δ_{h}, which make the small scale variables orthogonal to the large scale unknowns and should reduce the overall amount of numerical diffusion introduced in the problem. As in the previous case, we leave open the possibility of using either $\boldsymbol{a} = \boldsymbol{u}_{h} + \boldsymbol{u}_{s}$ or $\boldsymbol{a} = \boldsymbol{u}$ for convection and the corresponding stabilization terms. The matrix form of the Q-OSS

formulation can be expressed as

$$\begin{aligned}&\big[\boldsymbol{M}+\boldsymbol{S}_m\left(\tau_u,\boldsymbol{a}\right)\big]\dot{\boldsymbol{U}}+\big[\boldsymbol{C}\left(\boldsymbol{a}\right)+\boldsymbol{K}+\boldsymbol{S}_u\left(\tau_u,\boldsymbol{a}\right)+\boldsymbol{H}_u\left(\tau_p\right)\big]\boldsymbol{U}\\&+\big[\boldsymbol{G}+\boldsymbol{S}_p\left(\tau_u,\boldsymbol{a}\right)\big]\boldsymbol{P}=\boldsymbol{F}+\boldsymbol{T}+\boldsymbol{S}_f\left(\tau_u,\boldsymbol{a}\right)-\boldsymbol{S}_\Pi\left(\tau_u,\boldsymbol{a}\right)-\boldsymbol{H}_\Pi\left(\tau_u\right)\end{aligned}\tag{2.100}$$

$$\boldsymbol{Q}_m\left(\tau_u\right)\dot{\boldsymbol{U}}+\big[\boldsymbol{D}+\boldsymbol{Q}_u\left(\tau_u,\boldsymbol{a}\right)\big]\boldsymbol{U}+\boldsymbol{Q}_p\left(\tau_u\right)\boldsymbol{P}=\boldsymbol{Q}_f\left(\tau_u\right)-\boldsymbol{Q}_\Pi\left(\tau_u\right)\tag{2.101}$$

where the following three new terms terms, involving the projections, have been introduced:

$$\boldsymbol{S}_\Pi\left(\tau_u,\boldsymbol{a}\right)_a^e=\int_{\Omega_e}\left(\rho\boldsymbol{a}\cdot\nabla\boldsymbol{N}_a\right)^T\cdot\tau_u\,\boldsymbol{\xi}_h\,\mathrm{d}\Omega\tag{2.102}$$

$$\boldsymbol{Q}_\Pi\left(\tau_u\right)_a^e=\int_{\Omega_e}\left(\nabla N_a\right)^T\cdot\tau_u\,\boldsymbol{\xi}_h\,\mathrm{d}\Omega\tag{2.103}$$

$$\boldsymbol{H}_\Pi\left(\tau_p\right)_a^e=\int_{\Omega_e}\left(\nabla\cdot\boldsymbol{N}_a\right)^T\tau_p\,\boldsymbol{\delta}_h\,\mathrm{d}\Omega\tag{2.104}$$

The calculation of the projections involves the solution of an additional problem, given by Eqs. (2.48) and (2.49), which can be expressed in discrete form as

$$\boldsymbol{M}_\xi\boldsymbol{\Xi}=\boldsymbol{R}_\xi\tag{2.105}$$

$$\boldsymbol{M}_\delta\boldsymbol{\Delta}=\boldsymbol{R}_\delta\tag{2.106}$$

where $\boldsymbol{\Xi}$ and $\boldsymbol{\Delta}$ represent the vectors of nodal values of $\boldsymbol{\xi}_h$ and δ_h respectively and the remaining matrices and vectors are given by

$$\boldsymbol{M}_{\xi\,ab}^e=\int_{\Omega_e}\boldsymbol{N}_a^T\,\boldsymbol{N}_b\,\mathrm{d}\Omega\tag{2.107}$$

$$\boldsymbol{M}_{\delta\,ab}^e=\int_{\Omega_e}N_a\,N_b\,\mathrm{d}\Omega\tag{2.108}$$

$$\boldsymbol{R}_{\xi\,a}^e=\int_{\Omega_e}\boldsymbol{N}_a^T\cdot\boldsymbol{R}^m\left(\boldsymbol{u}_h,p_h\right)\mathrm{d}\Omega\tag{2.109}$$

$$\boldsymbol{R}_{\delta\,ab}^e=\int_{\Omega_e}N_a\,R^c\left(\boldsymbol{u}_h\right)\mathrm{d}\Omega\tag{2.110}$$

Eqs. (2.105) and (2.106) represent an additional problem, coupled to Eqs (2.100) and (2.101), which effectively doubles the number of nodal unknowns in the problem. However, since the system matrices for the projection problem, defined by Eqs. (2.107) and (2.108), are effectively mass matrices, they can be replaced by the corresponding diagonal mass matrix, which allows us to obtain an approximate projection while avoiding the solution of an additional system.

Note that the matrices $\boldsymbol{S}_m\left(\tau_u,\boldsymbol{a}\right)$ and $\boldsymbol{Q}_m\left(\tau_u\right)$ that appear in Eqs. (2.100) and (2.101) are not strictly necessary in OSS based formulations, since they involve the acceleration

term that appears in the residual $\boldsymbol{R}^m(\boldsymbol{u}_h, p_h)$. As noted in [27], the acceleration of the large scale lies in the large scale space V_h and therefore its projection should be itself. As a result, we could neglect both these terms if we also not take into account $\partial_t \boldsymbol{u}_h$ when evaluating the residual that appears in the projection right hand side vector $\boldsymbol{R}_\xi$. However, as the projection is only calculated approximately using a diagonal mass matrix, we have found it convenient to take these terms into account to improve stability.

2.5.3 Dynamic ASGS formulation

When we use a dynamic approximation for the subscales we have keep track and update the small scale values at each integration point of the mesh. The dynamic small scale velocities are defined by the local problem given by Eq. (2.37), which was discretized in time using a Backward Euler time scheme to produce Eq. (2.43). Eq. (2.43) provides an expression for $\boldsymbol{u}_s^{n+1}$ in terms of the residual and the old subscale value $\boldsymbol{u}_s^n$. The same Backward Euler scheme can also be used to obtain the following time-discrete expression for the small scale acceleration $\partial_t \boldsymbol{u}_s$ that appears in the variational form of the problem:

$$\partial_t \boldsymbol{u}_s \approx \frac{\boldsymbol{u}_s^{n+1} - \boldsymbol{u}_s^n}{\delta t} = \frac{\tau_t}{\delta t}\left(\boldsymbol{R}^m(\boldsymbol{u}_h, p_h)\big|_{n+1} - \boldsymbol{\xi}_h\right) + \rho\frac{\tau_t}{\delta t^2}\boldsymbol{u}_s^n - \frac{1}{\delta t}\boldsymbol{u}_s^n \tag{2.111}$$

Introducing Eq. (2.111) in Eqs. (2.46), we obtain a modified momentum equation with time-discrete small scales, given by

$$\begin{aligned}
&\int_\Omega \boldsymbol{w}_h \cdot \rho\left(\partial_t \boldsymbol{u}_h + \frac{1}{2}\boldsymbol{a}\cdot\nabla\boldsymbol{u}_h\right)\mathrm{d}\Omega - \int_\Omega \nabla\boldsymbol{w}_h : \rho\frac{1}{2}(\boldsymbol{a}\otimes\boldsymbol{u}_h)\,\mathrm{d}\Omega \\
&- \int_\Omega \nabla\cdot\boldsymbol{w}_h\, p_h\,\mathrm{d}\Omega + \int_\Omega \nabla^s\boldsymbol{w}_h :\, 2\mu\left(\nabla^s\boldsymbol{u}_h - \frac{1}{3}(\nabla\cdot\boldsymbol{u}_h)\boldsymbol{I}\right)\mathrm{d}\Omega \\
&+ \int_{\Sigma\,\Omega^e} \rho\boldsymbol{w}_h\cdot\left(\frac{\tau_t}{\delta t}(\boldsymbol{R}^m(\boldsymbol{u}_h,p_h) - \boldsymbol{\xi}_h) + \rho\frac{\tau_t}{\delta t^2}\boldsymbol{u}_s^n\right)\mathrm{d}\Omega - \int_\Omega \nabla\cdot\boldsymbol{w}_h\tau_p\left(R^c(\boldsymbol{u}_h) - \delta_h\right)\mathrm{d}\Omega \\
&- \int_{\Sigma\,\Omega^e} \rho(\boldsymbol{a}\cdot\nabla\boldsymbol{w}_h)\,\tau_t\left(\boldsymbol{R}^m(\boldsymbol{u}_h,p_h) - \boldsymbol{\xi}_h + \frac{\rho}{\delta t}\boldsymbol{u}_s^n\right)\mathrm{d}\Omega = \\
&\int_\Omega \boldsymbol{w}_h\boldsymbol{f}\,\mathrm{d}\Omega + \int_{\Gamma_N}\boldsymbol{w}_h\left(\boldsymbol{t} - \rho\frac{1}{2}(\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{u}_h\right)\mathrm{d}\Gamma + \int_{\Sigma\,\Omega^e}\rho\boldsymbol{w}_h\frac{1}{\delta t}\boldsymbol{u}_s^n\,\mathrm{d}\Omega
\end{aligned} \tag{2.112}$$

where the first term in the third row and the last term on the right hand side appear from the time discretization of the small scale acceleration. Note that all terms without an n index are evaluated at the current time step. Although they play no role in ASGS stabilization, we have included the projections $\boldsymbol{\xi}_h$ and $\boldsymbol{\delta}_h$ both in Eq. (2.111) and in Eq. (2.112), since the same expressions will be used as a starting point for the dynamic OSS method.

The matrix form of the D-ASGS formulation can be written as

$$\begin{aligned}&\big[\boldsymbol{M}+\hat{\boldsymbol{S}}_m\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)\big]\dot{\boldsymbol{U}}+\big[\boldsymbol{C}\left(+\right)\boldsymbol{K}+\hat{\boldsymbol{S}}_u\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)+\boldsymbol{H}_u\left(\tau_p\right)\big]\boldsymbol{U}\\&+\big[\boldsymbol{G}+\hat{\boldsymbol{S}}_p\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)\big]\boldsymbol{P}=\boldsymbol{F}+\boldsymbol{T}+\hat{\boldsymbol{S}}_f\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)+\boldsymbol{S}_d\left(\tau_t,\boldsymbol{u}_s^n\right)-\hat{\boldsymbol{S}}_d\left(\tau_t\right)\boldsymbol{u}_s^n\end{aligned} \tag{2.113}$$

$$\boldsymbol{Q}_m\left(\tau_t\right)\dot{\boldsymbol{U}}+\big[\boldsymbol{D}+\boldsymbol{Q}_u\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)+\boldsymbol{Q}_d\left(\tau_t,\boldsymbol{u}_s^n\right)\big]\boldsymbol{U}+\boldsymbol{Q}_p\left(\tau_t\right)\boldsymbol{P}=\boldsymbol{Q}_f\left(\tau_t\right) \tag{2.114}$$

where we have introduced three new terms on the right hand side involving the old subscale velocity, defined as

$$\boldsymbol{S}_d\left(\tau_t,\boldsymbol{u}_s^n\right)_a^e=\int_{\Omega_e}\left(\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_a\right)^T\cdot\rho\frac{\tau_t}{\delta t}\,\boldsymbol{u}_s^n\,\mathrm{d}\Omega \tag{2.115}$$

$$\hat{\boldsymbol{S}}_d\left(\tau_t\right)\boldsymbol{u}_{s\,a}^{n\,e}=\int_{\Omega_e}\rho\,\left(\boldsymbol{N}_a\right)^T\cdot\rho\frac{\tau_t}{\delta t^2}\,\boldsymbol{u}_s^n\,\mathrm{d}\Omega \tag{2.116}$$

$$\boldsymbol{Q}_d\left(\tau_t,\boldsymbol{u}_s^n\right)_a^e=\int_{\Omega_e}\left(\nabla\boldsymbol{N}_a\right)^T\cdot\rho\frac{\tau_t}{\delta t}\,\boldsymbol{u}_s^n\,\mathrm{d}\Omega \tag{2.117}$$

The terms that arise from the residual in the expression for the small scale acceleration, Eq. (2.111), have been grouped with the corresponding terms in the small scale velocity model, resulting in the following modified stabilization matrices:

$$\hat{\boldsymbol{S}}_m\left(\tau_t,\boldsymbol{a}\right)_{ab}^e=\int_{\Omega_e}\rho\left(\boldsymbol{a}\cdot\nabla\boldsymbol{N}_a-\frac{1}{\delta t}\boldsymbol{N}_a\right)^T\tau_t\,\boldsymbol{N}_b\,\mathrm{d}\Omega \tag{2.118}$$

$$\hat{\boldsymbol{S}}_u\left(\tau_t,\boldsymbol{a}\right)_{ab}^e=\int_{\Omega_e}\rho\left(\boldsymbol{a}\cdot\nabla\boldsymbol{N}_a-\frac{1}{\delta t}\boldsymbol{N}_a\right)^T\tau_t\,\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{N}_b\,\mathrm{d}\Omega \tag{2.119}$$

$$\hat{\boldsymbol{S}}_p\left(\tau_t,\boldsymbol{a}\right)_{ab}^e=\int_{\Omega_e}\rho\left(\boldsymbol{a}\cdot\nabla\boldsymbol{N}_a-\frac{1}{\delta t}\boldsymbol{N}_a\right)^T\tau_t\,\nabla N_b\,\mathrm{d}\Omega \tag{2.120}$$

$$\hat{\boldsymbol{S}}_f\left(\tau_t,\boldsymbol{a}\right)_{ab}^e=\int_{\Omega_e}\rho\left(\boldsymbol{a}\cdot\nabla\boldsymbol{N}_a-\frac{1}{\delta t}\boldsymbol{N}_a\right)^T\tau_t\,\boldsymbol{f}\,\mathrm{d}\Omega \tag{2.121}$$

Note that, unlike in the quasi-static variants, we will only consider using the full velocity $\boldsymbol{u}_h+\boldsymbol{u}_s$ to calculate convection for dynamic subscale models.

2.5.4 Dynamic OSS formulation

Finally, we consider the discrete form of the dynamic OSS formulation. Here, contrary to what happened for the D-ASGS formulation, the integral involving $\boldsymbol{w}_h$ and $\partial_t\boldsymbol{u}_s$ can be eliminated *a priori*, since it corresponds to the L^2 inner product of two terms that belong to orthogonal subspaces. This results in the following matrix formulation:

$$\begin{aligned}&\big[\boldsymbol{M}+\boldsymbol{S}_m\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)\big]\dot{\boldsymbol{U}}+\big[\boldsymbol{C}\left(+\right)\boldsymbol{K}+\boldsymbol{S}_u\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)+\boldsymbol{H}_u\left(\tau_p\right)\big]\boldsymbol{U}\\&\qquad+\big[\boldsymbol{G}+\boldsymbol{S}_p\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)\big]\boldsymbol{P}=\\&\boldsymbol{F}+\boldsymbol{T}+\boldsymbol{S}_f\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)+\boldsymbol{S}_d\left(\tau_t,\boldsymbol{u}_s^n\right)-\boldsymbol{S}_\Pi\left(\tau_t,\boldsymbol{u}_h+\boldsymbol{u}_s\right)-\boldsymbol{H}_\Pi\left(\tau_p\right)\end{aligned} \tag{2.122}$$

$$\boldsymbol{Q}_m(\tau_t)\dot{\boldsymbol{U}} + \left[\boldsymbol{D} + \boldsymbol{Q}_u(\tau_t, \boldsymbol{u}_h + \boldsymbol{u}_s)\right]\boldsymbol{U} + \boldsymbol{Q}_p(\tau_t)\boldsymbol{P} = \boldsymbol{Q}_f(\tau_t) + \boldsymbol{Q}_d(\tau_t, \boldsymbol{u}_s^n) - \boldsymbol{Q}_\Pi(\tau_t) \tag{2.123}$$

where all involved elemental matrices have already been defined in the previous sections.

Note that, as presented for the Q-OSS method, the projections can be solved using a diagonal mass matrix to avoid the solution of an additional system. In this case, there is also the possibility of re-introducing in Eq. (2.122) and (2.123) all terms neglected using an orthogonality argument. To do so, we can replace $\boldsymbol{S}_m(\tau_t, \boldsymbol{u}_h + \boldsymbol{u}_s)$, $\boldsymbol{S}_u(\tau_t, \boldsymbol{u}_h + \boldsymbol{u}_s)$, $\boldsymbol{S}_p(\tau_t, \boldsymbol{u}_h + \boldsymbol{u}_s)$ and $\boldsymbol{S}_d(\tau_t, \boldsymbol{u}_s^n)$ by their D-ASGS variants, given by Eqs. (2.118)–(2.121), and subtracting $\hat{\boldsymbol{S}}_d(\tau_t)\,\boldsymbol{u}_s^n$, given by Eq. (2.116), from the right hand side of Eq. (2.122).

2.5.5 Time integration

Regardless of the VMS variant used, once the problem has been discretized in space we obtain an equivalent matrix problem written in terms of the vectors of nodal velocities $\boldsymbol{U}$, pressures $\boldsymbol{P}$ and accelerations $\dot{\boldsymbol{U}}$ that can be expressed in general form as

$$\tilde{\boldsymbol{M}}\begin{bmatrix}\dot{\boldsymbol{U}}\\ \boldsymbol{0}\end{bmatrix} + \tilde{\boldsymbol{C}}\begin{bmatrix}\boldsymbol{U}\\ \boldsymbol{P}\end{bmatrix} = \tilde{\boldsymbol{F}} \tag{2.124}$$

We need to introduce a time discretization to write the accelerations in terms the velocities. For this we use the Bossak time integration method (see [54]), which can be described as a member of the generalized-α Newmark family of methods with second order accuracy in time. The basic expression of the Newmark method, which is commonly used in solid mechanics problem and written in terms of displacements d, velocities u and accelerations $\dot{u}$, is

$$d^{n+1} = d^n + \Delta t\, u^n + \frac{\Delta t^2}{2}\left[(1-2\beta_N)\,\dot{u}^n + 2\beta_N\right] \tag{2.125}$$

$$u^{n+1} = u^n + \Delta t\left[(1-\gamma_N)\,\dot{u}^n + \gamma_N \dot{u}^{n+1}\right] \tag{2.126}$$

where β_N and γ_N are constant parameters. In fluid dynamics it is convenient to rewrite Eq. (2.125) and (2.126) in terms of velocities, as these are the main variables of the problem, resulting in

$$\dot{u}_{n+1} = \frac{1}{\gamma_N \Delta t}\left(u^{n+1} - u^n\right) - \left(\frac{1}{\gamma_N} - 1\right)\dot{u}^n \tag{2.127}$$

$$d^{n+1} = d^n + \Delta t\left(1 - \frac{\beta_N}{\gamma_N}\right)u^n + \Delta t^2 \frac{\gamma_N - 2\beta_N}{2\gamma_N}\dot{u}^n + \frac{\beta_N \Delta t}{\gamma_N}u^{n+1} \tag{2.128}$$

Note that for the velocity based formulation, the displacements only appear in the equation for the new displacements, Eq. (2.128). As a result, both the displacements and the equation to obtain them can be omitted from the problem if we are not interested in their values.

The Bossak method is a generalization of the Newmark method based on introducing a relaxation factor on the acceleration of the system

$$(1-\alpha_B)\,\tilde{\boldsymbol{M}}\begin{bmatrix}\dot{\boldsymbol{U}}^{n+1}\\ \boldsymbol{0}\end{bmatrix}+\alpha_B\tilde{\boldsymbol{M}}\begin{bmatrix}\dot{\boldsymbol{U}}^{n}\\ \boldsymbol{0}\end{bmatrix}+\tilde{\boldsymbol{C}}\begin{bmatrix}\boldsymbol{U}^{n+1}\\ \boldsymbol{P}^{n+1}\end{bmatrix}=\tilde{\boldsymbol{F}} \tag{2.129}$$

In the Bossak scheme, Eq. (2.127) is used to discretize in time Eq. (2.129), obtaining a system that depends exclusively on velocities, pressures and their spatial gradients. After rearranging some terms, this yields the following time-discrete problem

$$\begin{aligned}\left(\frac{1-\alpha_B}{\gamma_N\Delta t}\tilde{\boldsymbol{M}}+\tilde{\boldsymbol{C}}\right)\begin{bmatrix}\boldsymbol{U}^{n+1}\\ \boldsymbol{P}^{n+1}\end{bmatrix}=&\\ \tilde{\boldsymbol{F}}-\frac{1-\alpha_B}{\gamma_N\Delta t}\tilde{\boldsymbol{M}}\begin{bmatrix}\boldsymbol{U}^{n}\\ \boldsymbol{0}\end{bmatrix}+\left\{(1-\alpha_B)\left(\frac{1}{\gamma_N}-1\right)+\alpha_B\right\}\tilde{\boldsymbol{M}}\begin{bmatrix}\dot{\boldsymbol{U}}^{n}\\ \boldsymbol{0}\end{bmatrix}&\end{aligned} \tag{2.130}$$

A common choice for the Bossak parameter is $\alpha_B = -0.3$, which provides maximum damping of high-frequency oscillations. The Newmark parameters are then chosen to be

$$\gamma_N=\frac{1}{2}-\alpha_B \qquad \beta_N=\frac{(1-\alpha_B)^2}{4}$$

2.5.6 Linearization of the large scale problem

The system described in Eq. (2.130) is non-linear due to the fact that the convective operator, the stabilization parameters, multiple stabilization matrices and the projections all depend on the current value of velocity. We define the residual of the problem at time step $n+1$ after i non-linear iterations as

$$\begin{aligned}\boldsymbol{R}\left(\boldsymbol{U}^{n+1,\,i},\boldsymbol{P}^{n+1,\,i}\right)=\tilde{\boldsymbol{F}}-\frac{1-\alpha_B}{\gamma_N\Delta t}\tilde{\boldsymbol{M}}\begin{bmatrix}\boldsymbol{U}^{n}\\ \boldsymbol{0}\end{bmatrix}+&\\ \left\{(1-\alpha_B)\left(\frac{1}{\gamma_N}-1\right)+\alpha_B\right\}\tilde{\boldsymbol{M}}\begin{bmatrix}\dot{\boldsymbol{U}}^{n}\\ \boldsymbol{0}\end{bmatrix}-\left(\frac{1-\alpha_B}{\gamma_N\Delta t}\tilde{\boldsymbol{M}}+\tilde{\boldsymbol{C}}\right)\begin{bmatrix}\boldsymbol{U}^{n+1,\,i}\\ \boldsymbol{P}^{n+1,\,i}\end{bmatrix}&\end{aligned} \tag{2.131}$$

The problem now consists in finding $\boldsymbol{U}^{n+1,\,i+1}$, $\boldsymbol{P}^{n+1,\,i+1}$ such that $\boldsymbol{R}^{n+1,\,i+1}=\boldsymbol{0}$. Denoting the increment between two successive iterations with $\boldsymbol{\delta U}^i=\boldsymbol{U}^{n+1,\,i+1}-\boldsymbol{U}^{n+1,\,i}$, we use a first order Taylor decomposition to write the zero of Eq. (2.131) as

$$\boldsymbol{R}\left(\boldsymbol{U}^{n+1,\,i+1},\boldsymbol{P}^{n+1,\,i+1}\right)=\boldsymbol{R}\left(\boldsymbol{U}^{n+1,\,i},\boldsymbol{P}^{n+1,\,i}\right)+\frac{\partial\boldsymbol{R}\left(\boldsymbol{U}^{n+1,\,i+1},\boldsymbol{P}^{n+1,\,i+1}\right)}{\partial\left(\boldsymbol{\delta U}^i,\boldsymbol{\delta P}^i\right)}\begin{bmatrix}\boldsymbol{\delta U}^i\\ \boldsymbol{\delta P}^i\end{bmatrix}=0 \tag{2.132}$$

We use Picard iterations, evaluating all matrices and vectors using the last known values of the variables. With this approximation, the system matrix can be written as

$$\left.\frac{\partial\boldsymbol{R}\left(\boldsymbol{U}^{n+1,\,i+1},\boldsymbol{P}^{n+1,\,i+1}\right)}{\partial\left(\boldsymbol{\delta U}^i,\boldsymbol{\delta P}^i\right)}\right|_i\approx\frac{1-\alpha_B}{\gamma_N\Delta t}\tilde{\boldsymbol{M}}+\tilde{\boldsymbol{C}} \tag{2.133}$$

which means that the linear system of equations that is assembled and solved at each iteration is

$$-\left(\frac{1-\alpha_B}{\gamma_N \Delta t}\tilde{\boldsymbol{M}}+\tilde{\boldsymbol{C}}\right)\begin{bmatrix}\boldsymbol{\delta U}^i\\ \boldsymbol{\delta P}^i\end{bmatrix}=\boldsymbol{R}\left(\boldsymbol{U}^{n+1,\,i},\boldsymbol{P}^{n+1,\,i}\right) \tag{2.134}$$

The problem given by Eq. (2.134) is solved iteratively until the increments of the system variables $\boldsymbol{\delta U}^i$ and $\boldsymbol{\delta P}^i$ or the residual vector $\boldsymbol{R}\left(\boldsymbol{U}^{n+1,\,i},\boldsymbol{P}^{n+1,\,i}\right)$ are smaller than a predefined tolerance.

2.5.7 Tracking of dynamic subscales

The dynamic small scale problem was discretized in time as Eq. (2.44). Using the full velocity, which is divergence-free, as the convective velocity, and neglecting the viscous stress term, since we are restricting ourselves to linear finite elements, we expand the residual $\boldsymbol{R}^m\left(\boldsymbol{u}_h,p_h\right)$ that appears in Eq. (2.44) to obtain the following expression for the small scale velocity:

$$\frac{1}{\tau_t}\boldsymbol{u}_s^{n+1}=\boldsymbol{f}-\rho\left(\partial_t\boldsymbol{u}_h^{n+1}-\left(\boldsymbol{u}_h^{n+1}+\boldsymbol{u}_s^{n+1}\right)\cdot\nabla\boldsymbol{u}_h^{n+1}\right)-\nabla p_h^{n+1}-\boldsymbol{\xi}_h^{n+1}+\frac{\rho}{\partial t}\boldsymbol{u}_s^n \tag{2.135}$$

Eq. (2.135) is non-linear, since τ_t and $\boldsymbol{\xi}_h$ both depend on $\boldsymbol{u}_s^{n+1}$ when using $\boldsymbol{a}=\boldsymbol{u}_h+\boldsymbol{u}_s$. Moreover, it is coupled to the large scale problem through its dependence to $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1}. Similarly, under these assumptions, all terms that depend on τ_t or the convection velocity in the large scale problem require a value for $\boldsymbol{u}_s^{n+1}$ to be computed.

To update the value of the small scale velocity we follow the procedure presented in [7, 8]. Given known values of the large scale variables, $\boldsymbol{u}_h^{n+1}$ and p_h^{n+1}, we define a target function

$$\begin{aligned}\boldsymbol{g}\left(\boldsymbol{u}_s^{n+1,\,k}\right)=&\boldsymbol{f}-\rho\left(\partial_t\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}\right)-\nabla p_h^{n+1}\\&-\boldsymbol{\xi}_h^{n+1}+\frac{\rho}{\partial t}\boldsymbol{u}_s^n-\frac{1}{\tau_t^k}\boldsymbol{u}_s^{n+1,\,k}-\rho\,\boldsymbol{u}_s^{n+1,\,k}\cdot\nabla\boldsymbol{u}_h^{n+1}\end{aligned} \tag{2.136}$$

where we use τ_t^k to denote τ_t computed using $\boldsymbol{a}=\boldsymbol{u}_h^{n+1}+\boldsymbol{u}_s^{n+1,\,k}$ in Eq. (2.42). We use Newton-Raphson iterations to find a zero of $\boldsymbol{g}\left(\boldsymbol{u}_s^{n+1}\right)$, resulting in

$$-\frac{\partial\boldsymbol{g}\left(\boldsymbol{u}_s^{n+1,\,k}\right)}{\partial\boldsymbol{u}_s^{n+1,\,k}}\left(\boldsymbol{u}_s^{n+1,\,k+1}-\boldsymbol{u}_s^{n+1,\,k}\right)=\boldsymbol{g}\left(\boldsymbol{u}_s^{n+1,\,k}\right) \tag{2.137}$$

The tangent matrix in Eq. (2.137) is computed neglecting the dependence of τ_t on $\boldsymbol{u}_s^{n+1,\,k}$, resulting in:

$$-\frac{\partial\boldsymbol{g}\left(\boldsymbol{u}_s^{n+1,\,k}\right)}{\partial\boldsymbol{u}_s^{n+1,\,k}}\approx\frac{1}{\tau_t}\boldsymbol{I}+\left(\nabla\boldsymbol{u}_h^{n+1}\right)^T \tag{2.138}$$

With this we can obtain new values for $\boldsymbol{u}_s^{n+1}$ by iteratively solving Eq. (2.137) on each integration point. These can be used to evaluate the values of the convective velocity

and τ_t in the next iteration of the large scale problem. Similarly, once the large scale problem is converged and we advance to the next time step, Eq. (2.137) is solved once more to obtain the historical values for the small scale.

Note that, for OSS formulations, $\boldsymbol{\xi}_h^{n+1}$ also depends on the value of $\boldsymbol{u}_s^{n+1,k}$, as it represents the L^2 projection of $\boldsymbol{R}^m(\boldsymbol{u}_h, p_h)$. However, this dependence is ignored in the procedure outlined in this section. Taking it into account would imply solving the (global) projection problem, given by Eqs. (2.105) and (2.106), every time a new $\boldsymbol{u}_s^{n+1,k}$ is obtained.

2.5.8 Finite element solution algorithm

As a summary of the formulation described in this section, we present the full solution procedure for the method as Algorithm 2.1, including the calculation of nodal projections and tracking of dynamic small scales. For variants where they are not necessary, the corresponding step in the algorithm can be skipped.

Algorithm 2.1 VMS incompressible flow solver.

for n in $[0, N]$ **do**
  Set $n = n + 1$, $t = t + \Delta t$
  Set $\boldsymbol{u}_h^{n+1,0} = \boldsymbol{u}_s^n$, $p_h^{n+1,0} = p_s^n$
  while $\left\| \boldsymbol{R}\left(\boldsymbol{U}^{n+1,i}, \boldsymbol{P}^{n+1,i}\right) \right\| \leq$ tolerance **do**
    Set $i = i + 1$
    for each integration point **do**
      Given $\boldsymbol{u}_h^{n+1,i}$, $p_h^{n+1,i}$, obtain $\boldsymbol{u}_s^{n+1,i}$ by iteratively solving Eq. (2.137).
    end for
    Assemble and solve the global system of Eq. (2.134) for $\boldsymbol{\delta U^i}$, $\boldsymbol{\delta P^i}$.
    Update $\boldsymbol{u}_h^{n+1,i}$, $p_h^{n+1,i}$.
    Use Eqs. (2.105) and (2.106) to find new projections $\boldsymbol{\xi}_h^{n+1,i}$, $\delta_h^{n+1,i}$.
  end while
  Calculate the large scale acceleration $\partial_t \boldsymbol{u}_h^{n+1}$ according to Eq. (2.127).
  for each integration point **do**
    Solve Eq. (2.137) to obtain historical values for the small scale $\boldsymbol{u}_s^{n+1}$.
  end for
end for

This procedure was implemented within the Kratos Multiphysics finite element framework and used to compute all numerical test cases presented in this chapter.

2.6 A new model for the pressure subscale

As described in the previous pages, the VMS formulation provides a stabilized method for the simulation of the Navier-Stokes equations that can be understood as a turbulence model, as justified in Section 2.4. However, its application in numerical simulations shows that the results have a strong dependency on the choice of the stabilization parameters. This is observed both in our own results and in the literature (see for example [7, 32], which use basically the same formulation we have described), where it can be observed that the obtained mean velocity profile in the turbulent channel flow changes depending on the precise definition of τ_u and on whether the pressure small scale is considered or taken to be zero. In particular, the results in [32] suggest that the optimal choice is problem-dependent, since some test cases provide a better fit to the reference data when the pressure small scale is kept, while in other cases neglecting it yields better results.

All this suggests that the current design for the stabilization parameters does not capture the correct behavior for at least some turbulent flow problems. To further investigate this issue, we study an alternate design for the pressure subscale. We start by motivating the design for the stabilization parameters we have been using up to this point and follow by presenting an alternative.

2.6.1 On the design of the stabilization parameters

Recall that the small scale model is motivated by the small scale problem, given by Eqs. (2.31) and (2.32), repeated here in simplified form as

$$\rho\partial_t \boldsymbol{u}_s + \rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{u}_s - 2\mu\nabla\cdot\nabla^s\boldsymbol{u}_s + \nabla p_s = \boldsymbol{R}^m\left(\boldsymbol{u}_h, p_h\right) - \boldsymbol{\xi}_h \qquad \text{in } \Omega^e\times[0,T) \tag{2.139}$$

$$\nabla\cdot\boldsymbol{u}_s = R^c\left(\boldsymbol{u}_h\right) - \delta_h \qquad \text{in } \Omega^e\times[0,T) \tag{2.140}$$

where we have written the convective term in non-conservative form and neglected the trace of $\nabla^s\boldsymbol{u}_s$. The motivation for the stabilization parameters used up to now, τ_u (or τ_t) and τ_p, is given in [28], where the response of Eqs. (2.139) and (2.140) to high wave number excitations (which we are most interested in, since the small scales represent highly fluctuating motions) is analysed. In such circumstances, it is observed that

$$\rho\,\boldsymbol{a}\cdot\nabla\boldsymbol{u}_s - 2\mu\nabla\cdot\nabla^s\boldsymbol{u}_s \sim \boldsymbol{u}_s\left[\left(c_1\frac{\mu}{h^2}\right)^2 + \left(c_2\frac{\rho\left\|\boldsymbol{a}\right\|}{h}\right)^2\right]^{1/2} \tag{2.141}$$

$$\left\|\nabla p_s\right\| \sim p_s\frac{1}{h} \qquad \nabla\cdot\boldsymbol{u}_s \sim \frac{1}{h}\left\|\boldsymbol{u}_s\right\| \tag{2.142}$$

The design for τ_u given in Eq. (2.39) can be justified by noting that $1/\tau_u$ has the same limit behavior as the term in Eq. (2.141) when either the convective or the viscous terms are significantly larger than the other. The justification for τ_p is given by introducing

Eq. (2.141) in the static version of Eq. (2.139) and taking the divergence. Assuming that $\boldsymbol{R}^m(\boldsymbol{u}_h, p_h)$ and $\boldsymbol{a}$ are divergence-free, we obtain

$$\left[\left(c_1\frac{\mu}{h^2}\right)^2+\left(c_2\frac{\rho\,\|\boldsymbol{a}\|}{h}\right)^2\right]^{1/2}\nabla\cdot\boldsymbol{u}_s+\nabla\cdot\nabla p_s=0 \tag{2.143}$$

or, equivalently,

$$\frac{1}{\tau_u}\nabla\cdot\boldsymbol{u}_s+\nabla\cdot\nabla p_s=0 \tag{2.144}$$

The analysis in [28] also shows that the pressure Laplacian in Eq. (2.144) behaves as $-1/h^2$, resulting in

$$\frac{1}{\tau_u}\nabla\cdot\boldsymbol{u}_s-\frac{c_1}{h^2}p_s=0 \tag{2.145}$$

where the algorithmic constant c_1 is adopted by analogy with the viscous term in Eq. (2.141). From this expression, and using Eq. (2.140) the usual formulation for the pressure subscale is recovered

$$\frac{1}{\tau_p}p_s\approx\nabla\cdot\boldsymbol{u}_s=R^c(\boldsymbol{u}_h)-\delta_h\qquad\tau_p=\frac{h^2}{c_1\tau_u}=\mu+\frac{c_2\,\|\boldsymbol{a}\|\,h}{c_1} \tag{2.146}$$

Finally, we introduce the scaling argument of Eq. (2.141) back to Eq. (2.139) and obtain

$$\rho\partial_t\boldsymbol{u}_s+\frac{1}{\tau_u}\boldsymbol{u}_s+\nabla p_s=\boldsymbol{R}^m(\boldsymbol{u}_h,p_h)-\boldsymbol{\xi}_h \tag{2.147}$$

The small scale model we have been using up to this point, given by Eq. (2.37), is then recovered by neglecting ∇p_s, which effectively uncouples the small scale velocity and pressure models.

2.6.2 Alternative design for the pressure subscale

As an alternative, we propose a formulation which keeps the pressure gradient ∇p_s in Eq. (2.139) and uses it to introduce the pressure subscale. We start by analyzing how keeping the small scale pressure gradient modifies the large scale problem. Retaining ∇p_s means that Eq. (2.147) can be rewritten as

$$\rho\partial_t\boldsymbol{u}_s+\frac{1}{\tau_u}\boldsymbol{u}_s=\boldsymbol{R}^m(\boldsymbol{u}_h,p_h)-\boldsymbol{\xi}_h-\nabla p_s \tag{2.148}$$

hence the corresponding time-discrete small scale velocity model is

$$\frac{1}{\tau_t}\boldsymbol{u}_s^{n+1}=\boldsymbol{R}^m(\boldsymbol{u}_h,p_h)|_{n+1}-\boldsymbol{\xi}_h^{n+1}-\nabla p_s^{n+1}+\frac{1}{\delta t}\boldsymbol{u}_s^n \tag{2.149}$$

Introducing Eq. (2.149) as our small scale model in Eqs. (2.46) and (2.47) we can obtain the full problem corresponding to this formulation. The resulting formulation, particularized for the quasi-static subscale case to reduce the number of terms involved, can be expressed as

$$\begin{aligned}
&\int_\Omega \boldsymbol{w}_h \cdot \rho \left(\partial_t \boldsymbol{u}_h + \frac{1}{2}\, \boldsymbol{a} \cdot \nabla \boldsymbol{u}_h \right) \mathrm{d}\Omega - \int_\Omega \nabla \boldsymbol{w}_h : \rho \frac{1}{2} \left(\boldsymbol{a} \otimes \boldsymbol{u}_h \right) \mathrm{d}\Omega \\
&+ \int_\Omega \nabla^s \boldsymbol{w}_h :\, 2\mu \left(\nabla^s \boldsymbol{u}_h - \frac{1}{3} \left(\nabla \cdot \boldsymbol{u}_h \right) \boldsymbol{I} \right) \mathrm{d}\Omega - \int_\Omega \nabla \cdot \boldsymbol{w}_h p_s \, \mathrm{d}\Omega \\
&\underline{- \int_{\Sigma \Omega^e} \rho \left(\boldsymbol{a} \cdot \nabla \boldsymbol{w}_h \right) \tau_u \left(\boldsymbol{R}^m \left(\boldsymbol{u}_h, p_h \right) - \boldsymbol{\xi}_h - \nabla p_s \right) \mathrm{d}\Omega} - \int_\Omega \nabla \cdot \boldsymbol{w}_h \, p_h \, \mathrm{d}\Omega = \\
&\qquad \int_\Omega \boldsymbol{w}_h \boldsymbol{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}_h \left(\boldsymbol{t} - \rho \frac{1}{2} \left(\boldsymbol{a} \cdot \boldsymbol{n} \right) \boldsymbol{u}_h \right) \mathrm{d}\Gamma
\end{aligned} \tag{2.150}$$

$$\int_\Omega q_h \nabla \cdot \boldsymbol{u}_h \, \mathrm{d}\Omega = \underline{\int_{\Sigma \Omega^e} \nabla q_h \, \tau_u \left(\boldsymbol{R}^m \left(\boldsymbol{u}_h, p_h \right) - \boldsymbol{\xi}_h - \nabla p_s \right) \mathrm{d}\Omega} \tag{2.151}$$

where the terms including contributions from ∇p_s have been underlined. Note that the same reasoning could be applied to dynamic subscale formulations.

If we could define an approximate space for p_s and discretize Eqs. (2.150) and (2.151), we would be able to write a matrix problem of the type

$$\begin{bmatrix} \boldsymbol{A}_{uu} & \boldsymbol{A}_{up} \\ \boldsymbol{A}_{pu} & \boldsymbol{A}_{pp} \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} + \begin{bmatrix} \boldsymbol{E}_u \left(\tau_u, \boldsymbol{a}, \nabla p_s \right) \\ \boldsymbol{E}_p \left(\tau_u, \nabla p_s \right) \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_u \\ \boldsymbol{B}_p \end{bmatrix} \tag{2.152}$$

where matrix $\boldsymbol{A}$ and vector $\boldsymbol{B}$ represent the matrix problem resulting from either Eqs. (2.80) and (2.81) for the Q-ASGS formulation or Eqs. (2.100) and (2.101) for the Q-OSS formulation, $\boldsymbol{U}$ and $\boldsymbol{P}$ are the (large scale) vectors of nodal unknowns and $\boldsymbol{E}$ results from the contribution of ∇p_s to the underlined terms in Eqs. (2.150) and (2.151). Its precise definition, if ∇p_s was known at the integration points of each element, would be given by

$$\boldsymbol{E}_u \left(\tau_u, \boldsymbol{a}, \nabla p_s \right)^e_{ab} = \int_{\Omega_e} \rho \left(\boldsymbol{a} \cdot \nabla \boldsymbol{N}_a \right)^T \cdot \tau_u \, \nabla p_s \, \mathrm{d}\Omega \tag{2.153}$$

$$\boldsymbol{E}_p \left(\tau_u, \nabla p_s \right)^e_{ab} = \int_{\Omega_e} \nabla N_a \tau_u \, \nabla p_s \, \mathrm{d}\Omega \tag{2.154}$$

We need to provide a model for ∇p_s before we can complete the formulation. To do so, we take a little detour. Consider that the finite element velocity solution $\boldsymbol{u}_h$, which is not divergence-free, can be decomposed as

$$\boldsymbol{u}_h = \boldsymbol{\omega} + \nabla \phi$$

where $\boldsymbol{\omega}$ is a solenoidal field verifying $\nabla \cdot \boldsymbol{\omega} = 0$ and $\nabla\phi$ is a potential field. We identify the pressure small scale with the potential part of the solution ϕ, with the goal of obtaining a velocity field that is more divergence-free in some weak sense.

Taking the divergence of $\boldsymbol{u}_h$ and integrating over the element's domain, we write

$$\int_{\Omega_e} q_s \nabla \cdot \boldsymbol{u}_h \, \mathrm{d}\Omega = \int_{\Omega_e} q_s \nabla \cdot \nabla p_s \, \mathrm{d}\Omega \tag{2.155}$$

Integrating by parts both sides of Eq. (2.155) we obtain

$$-\int_{\Omega_e} \nabla q_s \cdot \boldsymbol{u}_h \, \mathrm{d}\Omega = -\int_{\Omega_e} \nabla q_s \nabla p_s \, \mathrm{d}\Omega \tag{2.156}$$

Eq. (2.156) can not be evaluated in practice, since the exact small scale space is infinite-dimensional, but can be estimated using an approximate small scale space. Our proposal is to assume that the the space for the small scale pressure can be approximated by the discontinous version of the large scale space, that is, functions that are linear within each element and discontinuous across element boundaries. The fact that our interpolation functions are discontinuous across element boundaries allows us to write a local problem on each element, given by the discrete form of Eq. (2.156). Denoting with $\hat{N}_a$ the small scale shape function for node a, we can write

$$\hat{\boldsymbol{D}}^e \boldsymbol{U} + \hat{\boldsymbol{L}}^e \boldsymbol{P}_s = \boldsymbol{0} \tag{2.157}$$

where $\boldsymbol{U}$ and $\boldsymbol{P}_s$ represent the nodal values of $\boldsymbol{u}_h$ and p_s on the element and the matrices are build from nodal contributions of the type

$$\hat{\boldsymbol{D}}^e_{ab} = \int_{\Omega_e} \hat{N}_a \, \nabla \cdot \boldsymbol{N}_b \, \mathrm{d}\Omega \tag{2.158}$$

$$\hat{\boldsymbol{L}}^e_{ab} = \int_{\Omega_e} \nabla \hat{N}_a \, \nabla \hat{N}_b \, \mathrm{d}\Omega \tag{2.159}$$

Note that, for the shape functions we are proposing, on a given element, $\hat{\boldsymbol{D}}^e$ is equivalent to the elemental discrete divergence matrix $\boldsymbol{D}^e$ of the large scale problem, given by Eq. (2.87).

Matrix $\hat{\boldsymbol{L}}^e$ represents the discrete form of a Laplacian problem and can be explicitly inverted on each element if additional restrictions are imposed on the variable p_s. In our tests, we have imposed that p_s is zero on average over the element, which allows us to write

$$\boldsymbol{P}_s = \left(\hat{\boldsymbol{L}}^e\right)^{-1} \hat{\boldsymbol{D}}^e \boldsymbol{U} \tag{2.160}$$

Going back to the enhanced matrix problem of Eq. (2.152), the small scale shape functions can be used to rewrite the additional terms $\boldsymbol{E}_u$ and $\boldsymbol{E}_p$ in terms of the nodal P_s vector

$$\begin{bmatrix} \boldsymbol{A}_{uu} & \boldsymbol{A}_{up} \\ \boldsymbol{A}_{pu} & \boldsymbol{A}_{pp} \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} + \begin{bmatrix} \hat{\boldsymbol{E}}_u(\tau_u, \boldsymbol{a}) \\ \hat{\boldsymbol{E}}_p(\tau_u) \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_u \\ \boldsymbol{B}_p \end{bmatrix} \tag{2.161}$$

where we introduced the new finite element matrices

$$\hat{\boldsymbol{E}}_u\left(\tau_u, \boldsymbol{a}, \nabla p_s\right)_{ab}^e = \int_{\Omega_e} \rho\left(\boldsymbol{a}\cdot\nabla \boldsymbol{N}_a\right)^T \cdot \tau_u\, \hat{N}_b \,\mathrm{d}\Omega \tag{2.162}$$

$$\hat{\boldsymbol{E}}_p\left(\tau_u, \nabla p_s\right)_{ab}^e = \int_{\Omega_e} \nabla N_a \tau_u\, \hat{N}_b \,\mathrm{d}\Omega \tag{2.163}$$

Finally, we substitute Eq. (2.160) in Eq. (2.161), eliminating the additional variables from the problem algebraically

$$\begin{bmatrix} \boldsymbol{A}_{uu} + \hat{\boldsymbol{E}}_u\left(\tau_u, \boldsymbol{a}\right)\left(\hat{\boldsymbol{L}}^e\right)^{-1}\hat{\boldsymbol{D}}^e & \boldsymbol{A}_{up} \\ \boldsymbol{A}_{pu} + \hat{\boldsymbol{E}}_p\left(\tau_u, \boldsymbol{a}\right)\left(\hat{\boldsymbol{L}}^e\right)^{-1}\hat{\boldsymbol{D}}^e & \boldsymbol{A}_{pp} \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_u \\ \boldsymbol{B}_p \end{bmatrix} \tag{2.164}$$

To complete the formulation we would need to define a model for p_s (as opposed to its gradient), which also appears in Eq. (2.150). However, since we imposed that p_s is zero on average in each element to ensure that $\hat{\boldsymbol{L}}^e$ can be inverted and we are using linear shape elements (which means that $\nabla\cdot\boldsymbol{N}_a$ is constant within each element), it can be verified that, for the proposed formulation,

$$\int_{\Omega_e} \nabla\cdot\boldsymbol{w}_h p_s \,\mathrm{d}\Omega = c_e \int_{\Omega_e} p_s \,\mathrm{d}\Omega = 0 \tag{2.165}$$

where c_e is a constant that depends on the shape of the element.

2.7 Application to the turbulent channel flow

The turbulent channel flow is a classical benchmark for LES formulations, in which a fluid circulates between two parallel walls. In the turbulent regime, the flow is characterized by a transfer of energy from the central regions to the zones close to the wall, achieved through turbulent motions, where it is dissipated through viscous friction. This problem represents a challenge for turbulence models in general and LES methods in particular and is well studied in the literature [101, 125]. For moderate Reynolds numbers, it is also within reach of direct numerical simulation (DNS), which means that simulations representing all scales of the flow are available in the literature. In particular, we will use data from the simulations of Moser *et al.* [81] to validate our results. Note that, while this particular problem could be simulated using DNS, we are not interested in fully resolving all scales of the flow, since we want to study the behavior of the formulation *as a LES method*.

There is abundant literature validating VMS (and other) formulations on this particular benchmark, which was simulated using dynamic subscales in [32], and in [7] for the low Mach number regime; with and without explicit LES (Smagorinsky) modelling

terms in [45], using VMS methods in combination with isogeometric finite element formulations [3, 9] and using SUPG stabilization by itself [127] or in combination with the Smagorinsky model [123]. As a result, this example allows us to validate our implementation and test our new approach for the pressure subscale. However, we also note that all previous studies, as far as we know, have used linear hexahedra or higher order interpolations. In contrast with previous studies, we want to use this example to compare the results obtained with tetrahedral and hexahedral elements since, while hexahedra provide a richer interpolation, tetrahedra are in many cases the only practical choice to discretize complex geometries, and we want to quantify the impact of using tetrahedral interpolation on the solution.

The simulation consists in modeling the flow between two parallel flat plates that are separated a distance 2δ. The flow is driven by a pressure gradient applied on the streamwise direction, $\mathrm{d}P/\mathrm{d}x$, which is balanced by the friction produced by the wall, τ_w. The wall friction is conventionally expressed as $\tau_w = \rho u_\tau^2$, where u_τ is defined as the friction velocity. Given that the forces acting on the problem must be in equilibrium, the pressure gradient and the wall friction are related by the following expressions (see [125] for example):

$$\tau_w = -\delta \frac{\mathrm{d}P}{\mathrm{d}x} = \rho u_\tau^2 \quad \text{or} \quad u_\tau = \left(-\frac{\delta}{\rho} \frac{\mathrm{d}P}{\mathrm{d}x} \right)^{\frac{1}{2}}$$

The Reynolds number can be given in terms of the friction velocity and the channel width as

$$\mathrm{Re}_\tau = \frac{u_\tau \delta}{\nu}$$

which is denoted by Re_τ to distinguish it from the bulk Reynolds number, computed using the average streamwise velocity of the flow (see [101]).

For our tests, the Reynolds number is set to $\mathrm{Re}_\tau = 395$, which can be obtained by setting the problem parameters to

$$\rho = 1\, Kg/m^3 \quad \nu = 1.472 \times 10^{-4}\, m^2/s \quad \delta = 1\, m \quad \frac{\mathrm{d}P}{\mathrm{d}x} = -3.372040 \times 10^{-3}\, N/m^2$$

The problem domain is restricted to $[0, 2\pi] \times [-\delta, \delta] \times [0, 2\pi/3]$ in the stream-wise, wall-normal and cross-stream directions respectively, which corresponds to the domain used in [9]. Zero Dirichlet conditions are applied on the solid walls and periodic boundary conditions are used for the remaining sides.

2.7.1 Effect of the simulation mesh

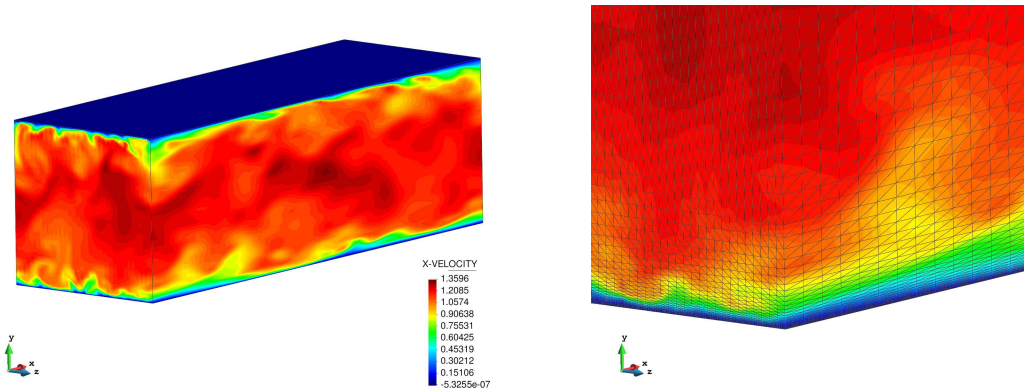
For a first set of tests, we simulate the problem using regular hexahedral and tetrahedral meshes. The meshes are defined by introducing 32 or 64 divisions on each coordinate direction, which immediately defines a mesh of 32^3 or 64^4 hexahedra. Tetrahedral meshes can then be obtained by splitting each hexahedra into six tetrahedra. The mesh nodes

are evenly distributed on the streamwise and cross-stream directions while, on the wall normal direction, they are distributed according to the law

$$y^i = \delta \frac{\tanh\left(w\left(\frac{2i}{n_{el}} - 1\right)\right)}{\tanh(w)} \qquad i \in [0, n_y]$$

where n_y is the number of divisions on the y direction and $y^i \in [-\delta, \delta]$. The weight w is chosen as $w = 2.432$ for $n_y = 64$ and $w = 2.927$ for $n_y = 32$ so that the first node has a dimensionless distance to the wall $y+ = yu_\tau/\nu = 1$.

The flow is simulated using a time step of $\Delta t = 0.02\,s$, starting from the average flow profile plus a random disturbance and let to evolve until a statistically steady regime is obtained. Discarding this initial transient phase, statistics are recorded on the integration points of the mesh using the method described in Chapter 4. Since the problem is statistically homogeneous, statistics can be obtained by ensemble-averaging data on planes corresponding to the same wall distance. This averaging of spatial and time data is known as Reynolds averaging in the context of turbulence modeling. We denote the (Reynolds) average value of a quantity u as $\langle u \rangle$ and the fluctuation as $u' = u - \langle u \rangle$. A snapshot of the obtained instantaneous streamwise velocity and the distribution of the elements close to the wall can be observed in Fig. 2.2 for a tetrahedral mesh and in Fig. 2.3 for a hexahedral mesh.

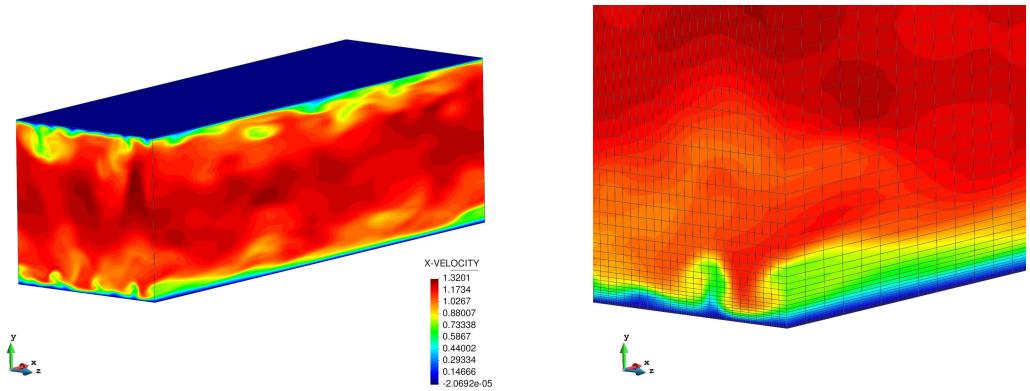


(a) Instantaneous streamwise velocity. (b) Detail of the mesh.

Figure 2.2: Channel flow – Solution and mesh for the 6×64^3 tetrahedra simulation.

We have measured both average velocities and velocity correlations, all of which can be compared to the DNS data of [81]. Note that the velocity fluctuation correlations can be identified with the turbulence kinetic energy, defined as

$$k = \frac{1}{2}\left(\langle u'u' \rangle + \langle v'v' \rangle + \langle w'w' \rangle\right)$$



(a) Instantaneous streamwise velocity. (b) Detail of the mesh.

Figure 2.3: Channel flow – Solution and mesh for the 64^3 hexahedra simulation.

The first set of results has been obtained using the Q-ASGS formulation and the different meshes. The obtained average is compared to the reference data in Fig. 2.4, while the velocity variances are presented in Fig. 2.5. Note that the results are expressed in terms of the dimensionless distance to the wall $y+ = (\delta - |y|)\, u_\tau/\nu$, which is the common practice for this problem. In this notation, $y+ = 0$ corresponds to the wall, while the channel center line is close to $y+ = 400$.

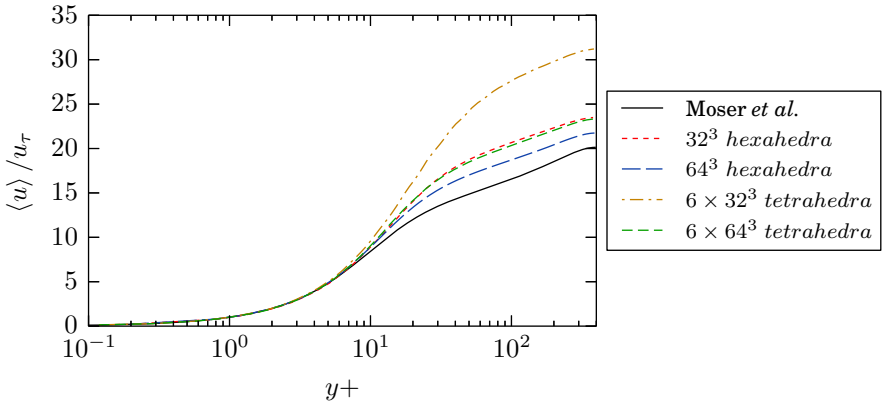
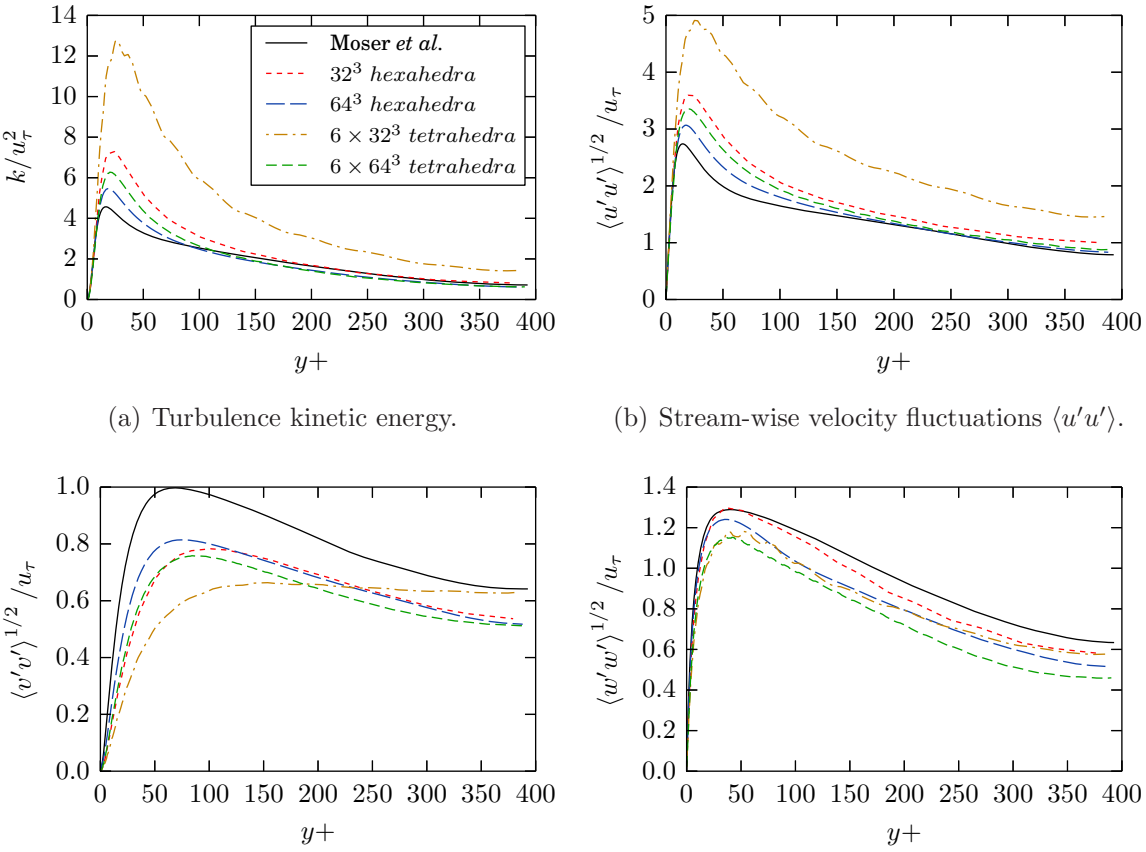


Figure 2.4: Channel flow – average stream-wise velocity profile obtained for the Q-ASGS formulation, using different meshes.

Not surprisingly, there is a noticeable change of behavior depending on the element type, with tetrahedra producing generally poorer results for a given mesh size than hexahedra. In particular, it can be observed that the results obtained using 6×64^3 tetrahedra are similar to those obtained 32^3 hexahedra, which suggests that an order of magnitude more tetrahedra are required in this case to obtain a solution comparable to



(a) Turbulence kinetic energy.

(b) Stream-wise velocity fluctuations $\langle u'u' \rangle$.

(c) Wall-normal velocity fluctuations $\langle v'v' \rangle$.

(d) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

Figure 2.5: Channel flow – velocity variances obtained for the Q-ASGS formulation, using different meshes.

hexahedra.

We can see that we generally overpredict the average streamwise velocity profile, which suggests that we are not dissipating enough linear momentum and stronger velocity gradients are required to achieve an equilibrium solution. Observing the measured velocity variances in Fig. 2.5, we can see that there are larger than expected velocity fluctuations on the streamwise direction. The dynamic evolution of the solution presents larger divergences from the average value than expected, which again suggests that we are slightly underestimating the dissipation. This situation is reversed on the wall-normal and cross-flow directions, where the obtained dissipation is generally under the DNS measurements.

2.7.2 Influence of the small scale model

The next set of tests is designed to evaluate the effect of the small scale model on the solution. We have chosen the 32^3 hexahedra mesh and used it to simulate the same case using the different models presented in Section 2.3. Based on the results presented in [32], we have chosen to neglect the pressure small scale for this set of tests, which corresponds to setting $\tau_p = 0$, as this was found to result in a better fit to DNS data in that reference. Note that the quasi-static results presented were calculated using only the large scale part of the solution in the convection term, while the full velocity $\boldsymbol{u}_h + \boldsymbol{u}_s$ was used for dynamic models.

The average stream-wise velocity profiles for this set of tests are presented Fig. 2.6, while the measured velocity variances are shown in Fig. 2.7. The results are in general comparable to those obtained for the same mesh in the previous set of tests and, while the qualitative behavior of the solution is properly captured, the average velocity is slightly overpredicted.

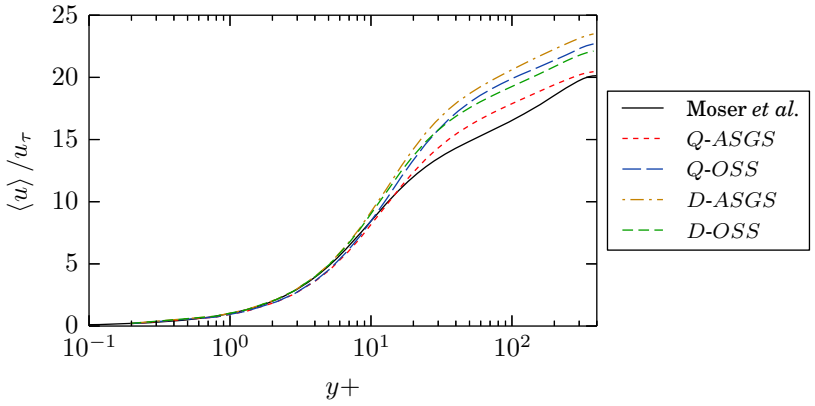
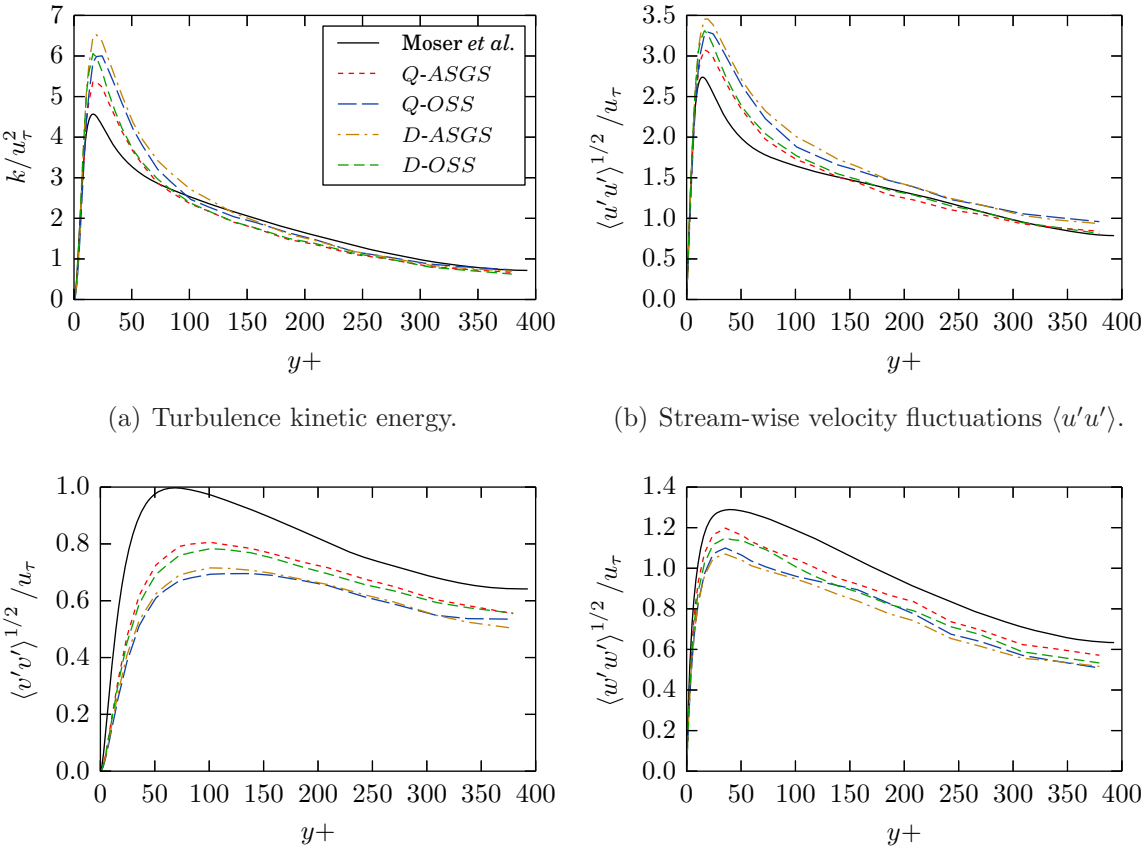


Figure 2.6: Channel flow – average stream-wise velocity profile obtained on the 32^3 hexahedra mesh using different small scale models.

From the obtained results, we can see that the choice of small scale model has an impact on the solution. However, we see that, in our case, the result that produces a closer approximation to the DNS stream-wise velocity profile is the Q-ASGS model, which is the most simplified one from the theoretical point of view.

In terms of the velocity variances, the points made for the previous set of tests still stand. The stream-wise velocity variances are generally overestimated compared to the expected results, while the fluctuations on the other directions are underestimated. Again, the different models introduce some differences in the final solution, with the Q-ASGS and D-OSS variants providing the closest match to DNS data.



(a) Turbulence kinetic energy.

(b) Stream-wise velocity fluctuations $\langle u'u' \rangle$.

(c) Wall-normal velocity fluctuations $\langle v'v' \rangle$.

(d) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

Figure 2.7: Channel flow – velocity variances obtained for the Q-ASGS formulation, using different meshes.

2.7.3 Turbulence kinetic energy balance

To obtain a deeper understanding of the results we measured the turbulence kinetic energy balance for the problem. A balance statement for the turbulence kinetic energy can be obtained using a procedure analogous to what presented in Section 2.4 for the residual energy k_r, using Reynolds averaging in place of filtering. Only the final expression is presented here, but the interested reader is directed to [101, 120] for detailed proof.

$$\underbrace{\partial_t k + \langle u_j \rangle \frac{\partial k}{\partial x_j}}_{I} = \underbrace{- \left\langle u_i' u_j' \right\rangle \frac{\partial \langle u_i \rangle}{\partial x_j}}_{II} \underbrace{- \frac{\left\langle u_j' k \right\rangle}{\partial x_j}}_{III} \underbrace{- \frac{1}{\rho} \frac{\partial \left\langle u_i' p' \right\rangle}{\partial x_i}}_{IV} \underbrace{+ \nu \frac{\partial^2 k}{\partial x_j^2}}_{V} \underbrace{- \nu \left\langle \frac{\partial u_i'}{\partial x_j} \frac{\partial u_i'}{\partial x_j} \right\rangle}_{VI} \quad (2.166)$$

where the terms represent, respectively:

I. Material variation (storage and advection) of turbulence kinetic energy;

II. production due to mean velocity shears;

III. turbulent diffusion: transport due to small eddies;

IV. pressure diffusion: redistribution due to local pressure gradients;

V. viscous diffusion and

VI. viscous dissipation.

For the turbulent channel flow, we have that the average velocity is exclusively in the streamwise (x) direction and that the flow is homogeneous in the streamwise and cross-stream directions. As a result, all terms involving either $\langle v \rangle$, $\langle w \rangle$ or spatial derivatives along the x or z directions can be can be neglected from Eq. (2.166), resulting in the simplified expression

$$\underbrace{\partial_t k}_{I} = \underbrace{-\langle u'v' \rangle \frac{\partial \langle u \rangle}{\partial y}}_{II} \underbrace{- \frac{\langle v'k \rangle}{\partial y}}_{III} \underbrace{- \frac{1}{\rho} \frac{\partial \langle v'p' \rangle}{\partial y}}_{IV} \underbrace{+ \nu \frac{\partial^2 k}{\partial y^2}}_{V} \underbrace{- \nu \left\langle \frac{\partial u_i'}{\partial y} \frac{\partial u_i'}{\partial y} \right\rangle}_{VI} \tag{2.167}$$

where the numbered terms have the same interpretation as the corresponding term in Eq. (2.166). Note that, once a statistically steady state has been reached, the storage term in Eq. (2.167) will also be zero, as in equilibrium the power introduced in the system by the external pressure gradient is exactly balanced by wall friction.

The different terms in Eq. (2.167) have been measured in the course of the simulations presented in the previous pages and are presented in graphical form in Figs. 2.8 to 2.17, again compared to DNS measurements from [81]. Note that in this case we are only presenting the part of the solution closer to the wall, $y+ \in [0, 200]$, as energy transfer phenomena are mostly localized close to the wall for this problem.

The values for term II in Eq. (2.167), turbulence kinetic energy production, are plotted in Fig. 2.8 for the different meshes and in Fig. 2.9 for the different small scale models. The production term measures the generation of small scale motions due to the shear of the average flow and, for this example, is expected to be positive throughout the domain, reaching a peak close to the wall, near $y+ = 10$. As an aside, the fact that this term is positive means that there is no backscatter in this problem.

We observe that our results are in qualitative agreement with DNS data, but tend to predict a peak in production at larger $y+$ (farther from the wall) than expected. This is most marked for the coarser tetrahedral mesh in Fig. 2.8(a), while hexahedral meshes predict the peak closer to the expected position in general. It is worth noting that the curves obtained from our simulation have a jagged appearance when compared to DNS data. This is due to the choice of linear interpolation functions, which means that the

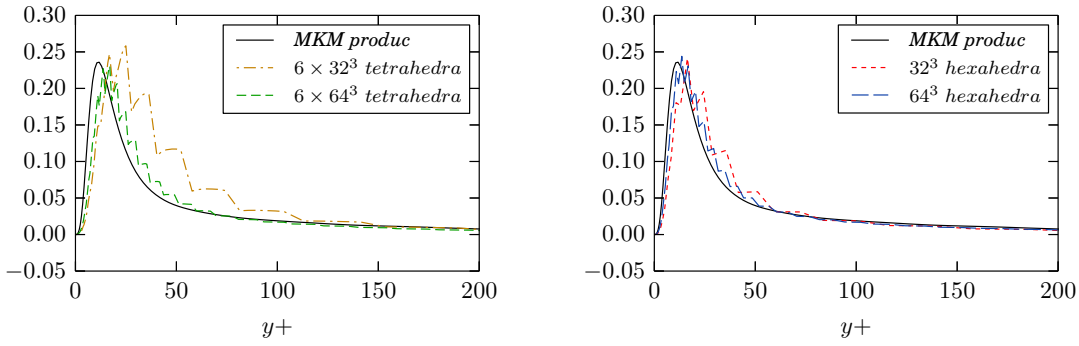


Figure 2.8: Channel flow – turbulence kinetic energy production for the Q-ASGS method.

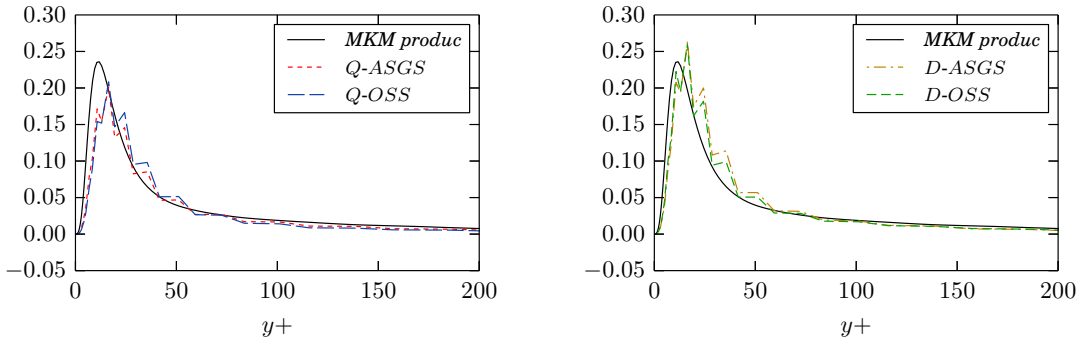


Figure 2.9: Channel flow – turbulence kinetic energy production in the 32^3 hexahedra case for the different small scale models.

simulated velocity gradient is constant within each finite element. This will also happen in any other results involving spatial gradients of finite element variables.

In terms of the different small scale models tested, we see that the dynamic models in Fig. 2.9(b) provide a better match to DNS data than the quasi-static models in Fig. 2.9(a), suggesting that dynamic models do in fact provide a better description of turbulent phenomena compared to the simpler quasi-static models.

As a final remark, note that the 32^3 hexahedra, Q-ASGS curve in Fig. 2.8(b) does not coincide with the Q-ASGS curve in Fig. 2.9(a). The difference between the two curves is due to the pressure small scale, which was considered in the first case and neglected in the latter. It is interesting to observe that retaining it results in a better prediction of the production term, despite the fact that we saw in the previous pages that neglecting the pressure small scale results in a closer approximation of the average stream-wise velocity.

Term III of Eq. (2.167), which corresponds to turbulent diffusion, is presented in

Fig. 2.10 for the different meshes. Turbulent diffusion is expected to transfer energy from the intermediate region of the channel towards the wall, which means that it will act as a source of turbulence kinetic energy (positive values) close to the wall and as a drain (negative values) at large $y+$. We see that our results follow the expected distribution of values, closer when using hexahedral meshes. However, it is interesting to note that the positive peak, near $y+ = 10$, is underestimated in the finer simulations.

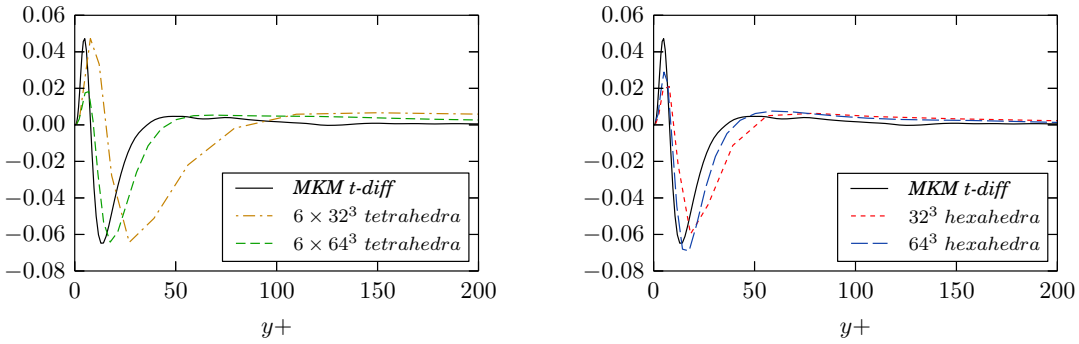


Figure 2.10: Channel flow – turbulent diffusion for the Q-ASGS method.

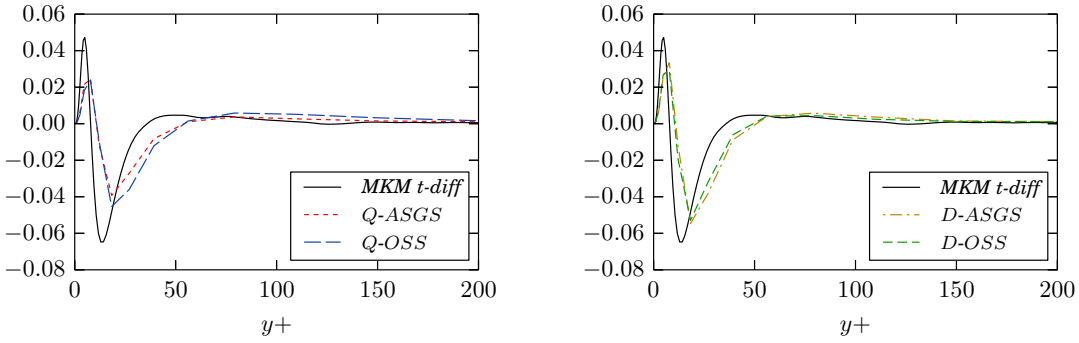


Figure 2.11: Channel flow – turbulent diffusion in the 32^3 hexahedra case for the different small scale models.

The measured turbulent diffusion for the different small scale models is presented in Fig. 2.11 and shows the same type of dependence on the model that we observed for the production term. Again, dynamic subscales provide a better approximation to DNS data than quasi-static ones and, comparing Fig. 2.10(b) to Fig. 2.11(a), using the pressure small scale in the Q-ASGS case provides a closer match than neglecting it.

Turning our attention to term IV, which represents pressure diffusion, we provide the results corresponding to different meshes in Fig. 2.12 and to the different small scale models in Fig. 2.13. This term corresponds to the relation between local pressure and

velocity fluctuations and it has a qualitative behavior that is similar to that of turbulent diffusion, transporting energy closer to the wall, but a smaller magnitude in general. As in the previous term, we can observe that the obtained results tend to underestimate sharp peaks. However, in this case, we detect an unexpected behavior close to the wall for the finest hexahedral mesh, as can be seen in Fig. 2.12(b), where the 64^3 hexahedra curve shows a negative peak close to the wall. As a possible interpretation of this result, we note that elements are highly stretched in that region and that we are using an average element size to define our stabilization parameter. This could have unexpected effects on the consistency of the stabilization terms and might be adding some error in our calculations.

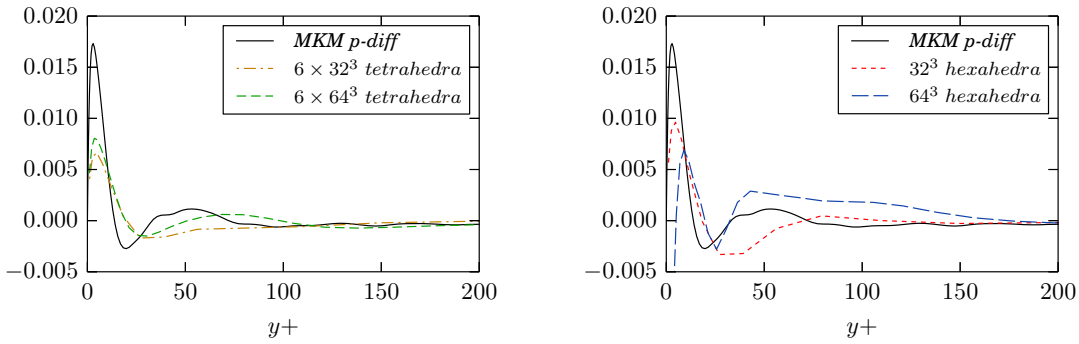


Figure 2.12: Channel flow – pressure diffusion for the Q-ASGS method.

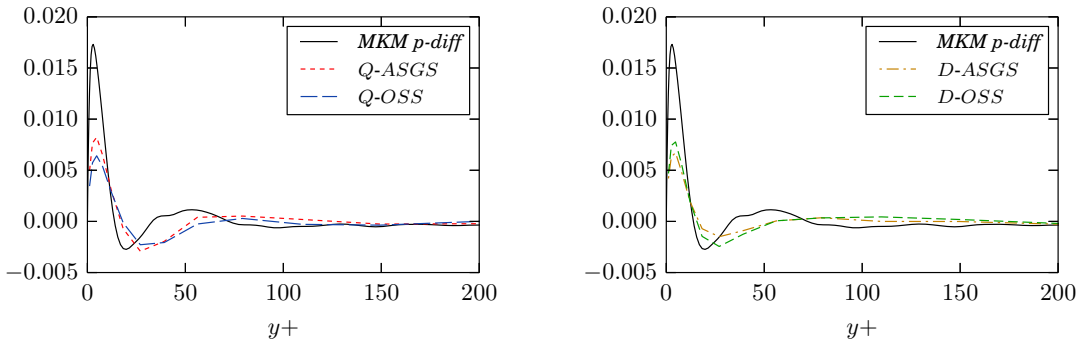


Figure 2.13: Channel flow – pressure diffusion in the 32^3 hexahedra case for the different small scale models.

The results obtained for term V, representing viscous diffusion, are presented in Fig. 2.14 for the different meshes used and in Fig. 2.15 for the different small scale models. As the other two diffusive terms, viscous diffusion transports energy towards the wall but, compared to the the previous terms, it acts closer to the wall in general.

This is consistent with the fact that viscous effects are predominant only in the smallest motions, which, for the turbulent channel flow, are significant only at a very close distance from the wall.

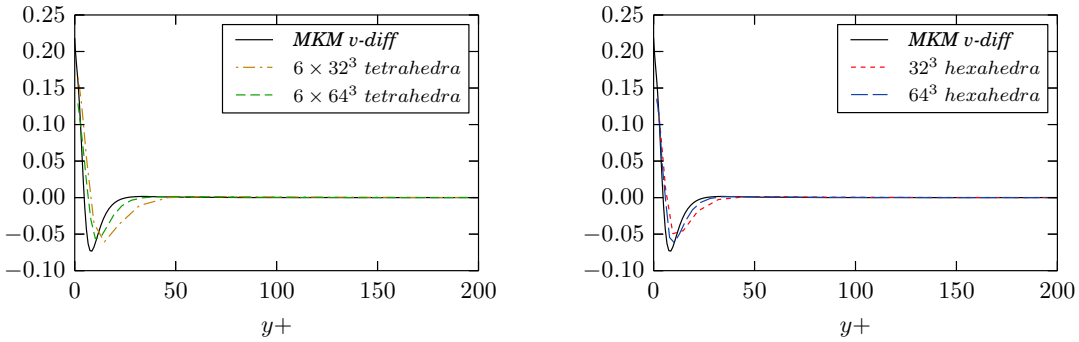


Figure 2.14: Channel flow – viscous diffusion for the Q-ASGS method.

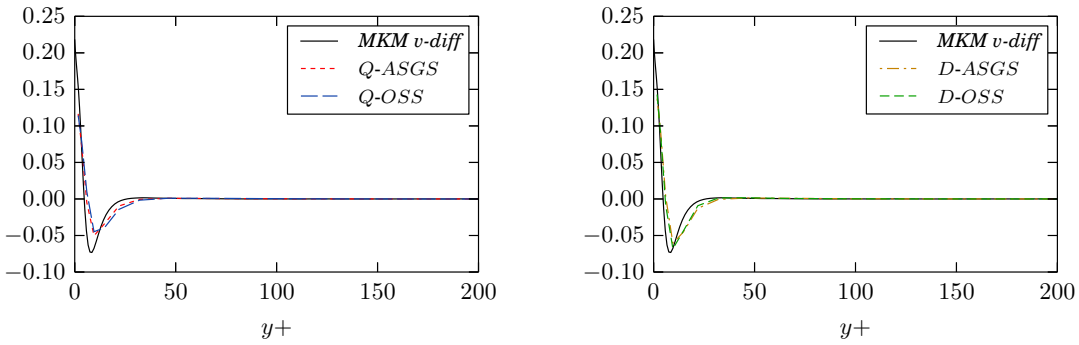


Figure 2.15: Channel flow – viscous diffusion in the 32^3 hexahedra case for the different small scale models.

The final term in Eq. (2.167), term VI, corresponds to viscous dissipation of turbulence kinetic energy. This term should be negative throughout the domain, as viscous dissipation is the only energy sink available in the problem, which can be verified in Fig. 2.16 for the different meshes and Fig. 2.17 for the different small scale models.

Viscous dissipation should predominantly occur close to the wall, where the smallest motions are concentrated, and we notice that our results follow this general trend, in agreement with DNS data. However, all our curves indicate a smaller (closer to zero) dissipation for any given distance to the wall, which seems to be generally in line with our interpretation of the average velocity results, indicating that dissipation is generally lower than expected.

In terms of the comparison between the different methods, the results obtained for dissipation are in agreement with the general trend observed for the previous terms. We notice that dynamic models, as shown in Fig. 2.17(b), provide results that are slightly closer to DNS measurements than quasi-static models, which in this case is clear in the small step displayed by the different curves close to the wall, and that the Q-ASGS results in Fig. 2.16(b), obtained using the pressure subscale, are slightly better than those in Fig 2.17(a), obtained by neglecting it.

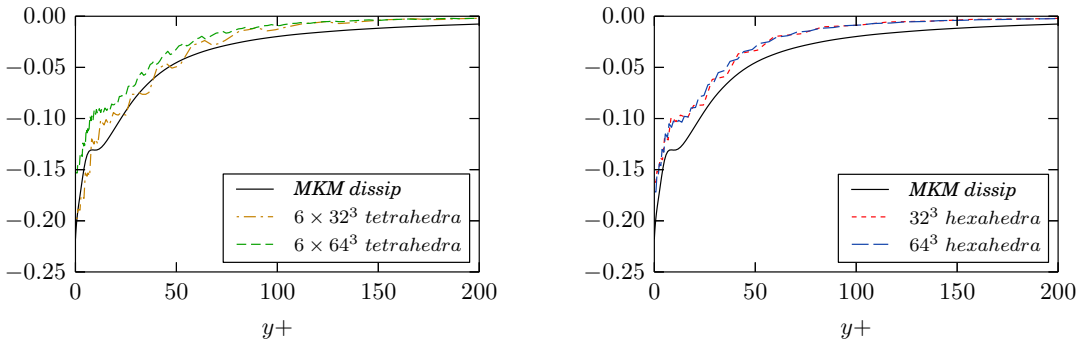


Figure 2.16: Channel flow – turbulence kinetic energy dissipation for the Q-ASGS method.

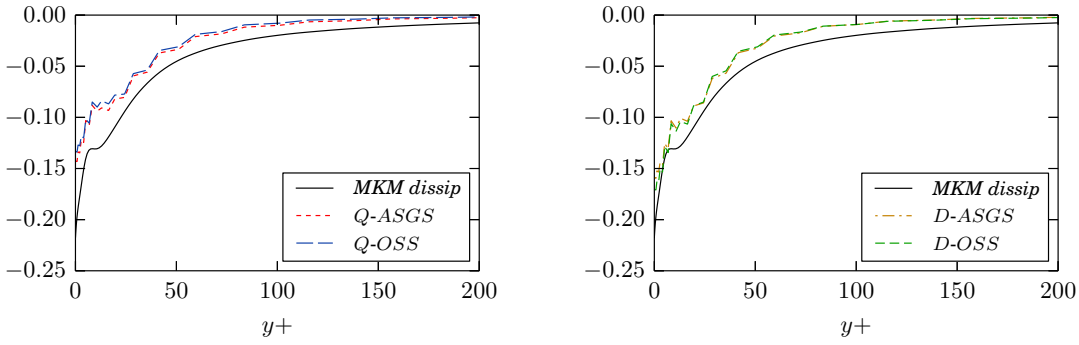
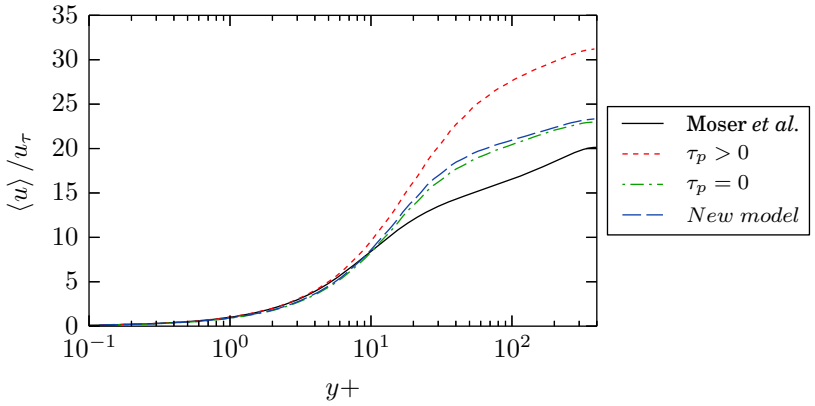


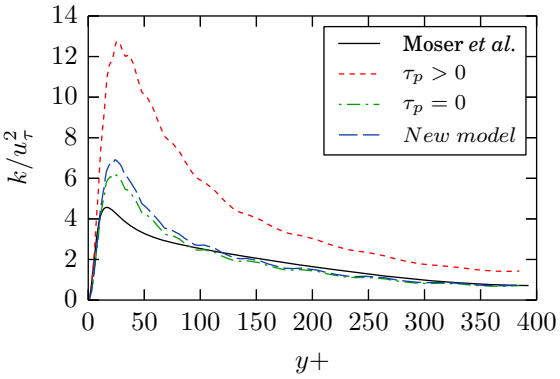
Figure 2.17: Channel flow – turbulence kinetic energy dissipation in the 32^3 hexahedra case for the different small scale models.

2.7.4 Effect of the proposed pressure subscale model

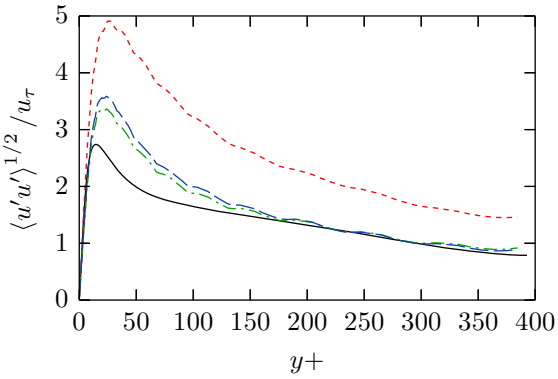
As a final test, we simulated the same problem using the alternative model for the pressure subscale proposed in Section 2.6 and the coarser tetrahedral grid, comprising 6×32^3 tetrahedra. We present the results in terms of the average stream-wise velocity



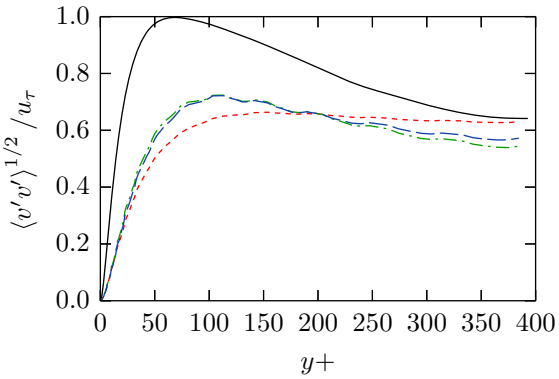
(a) Average stream-wise velocity



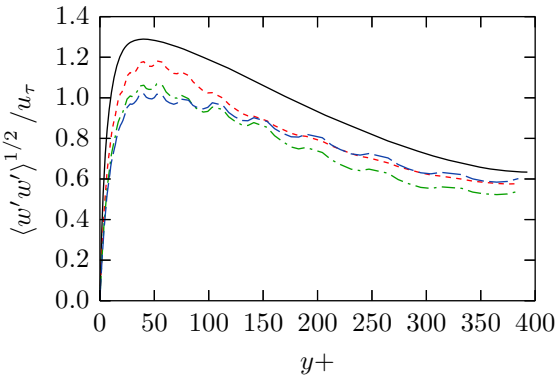
(b) Turbulence kinetic energy.



(c) Stream-wise velocity fluctuations $\langle u'u' \rangle$.



(d) Wall-normal velocity fluctuations $\langle v'v' \rangle$.



(e) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

Figure 2.18: Channel flow – stream-wise velocity average and velocity variances obtained using the proposed pressure small scale model and a 6×32^3 tetrahedra mesh.

profile and velocity variances in Fig. 2.18, where we compare them to DNS data and to the regular Q-ASGS formulation, either retaining ($\tau_p > 0$) or neglecting ($\tau_p = 0$) the pressure small scale. We observe that both neglecting the pressure small scale and using the proposed model result in a much closer agreement to DNS data compared to retaining τ_p. This allows us to obtain results that are much closer to those obtained using 32^3 hexahedra, which were presented in Fig. 2.4, while using a tetrahedral mesh.

2.8 Concluding remarks

We devoted the present chapter to introduce VMS stabilized formulations for the incompressible Navier-Stokes equations and to expose the arguments that have been used to relate them to LES methods for turbulence modeling. Besides the classical VMS approach, we have discussed dynamic subscale models and the possibility of using the last known value of the complete velocity $\boldsymbol{u}_h^{n+1,i} + \boldsymbol{u}_s^{n+1,i}$ as the linearized advective velocity on the convective term. These two modifications to the basic formulation allow us to provide a stronger theoretical justification to the use of VMS methods as a type of turbulence modeling, similar to LES but using a projection to the mesh instead of spatial filtering to introduce scale separation. We have implemented a finite element solver based on dynamic subscale formulations and used it to simulate the well-known benchmark of the turbulent channel flow at $\mathrm{Re}_\tau = 395$.

We have investigated the effect of using either tetrahedral or hexahedral meshes for the simulation and the use of different small scale models. Motivated by results found on the literature, we decided to neglect the pressure subscale in some of the tests, in the hope of obtaining more accurate solutions. While it is true that neglecting the small scale pressure results in a better agreement to DNS data in terms of the average streamwise velocity in the single direct comparison we have for this (the Q-ASGS model), we have also found that this choice results in a poorer prediction of the turbulence kinetic energy balance. We do not have a definitive answer to this apparent contradiction, but one possible explanation could be that the small scale pressure introduces an unexpected energy transfer mechanism, which we do not detect in our balance, since we are only measuring the (large scale part of) the terms in Eq. (2.167). Obviously, more tests are required before a definitive answer can be provided, and the first step would be repeating the analysis presented in Section 2.7 but now without neglecting the pressure subscale and measuring not only the large scale part of the energy balance, but also the contributions of the small scale velocities to the terms in Eq. (2.167).

This dependence of the results on the pressure small scale also motivated us to propose a new model for the pressure small scales, which we based on strengthening the enforcement of the incompressibility of the velocity solution via the use of an approximate interpolation space for the small scale pressure. While this new formulation produces improved results when compared to the classical approach on a tetrahedral mesh, we want to remark that the effect of the pressure small scale term seems to be

problem-dependent (this can be seen for example in [32]), which means that this should be understood as the starting point of a wider investigation and not a definitive result.

In the same sense, in proposing the formulation we have made some arbitrary decisions, such as the choice of a discontinuous linear interpolation space or the zero-average condition used to invert the Laplacian in Eq. (2.160). These are by no means the only possibilities, and it would be interesting to know the impact of this choice compared to other alternatives.

Chapter 3

A Finite Calculus stabilized finite element formulation for turbulent flows

3.1 Introduction

In the present chapter we introduce a new stabilized finite element formulation for the simulation of incompressible flow problems based on the Finite Calculus (FIC) approach [86, 87]. FIC is a general framework for the development of stabilized formulations, based on writing the balance equations of the problem for an arbitrarily small domain, instead of the usual point-wise partial differential equation (PDE). This results in a modified strong-form equation with additional terms that, once the problem is rewritten as a variational equation, have a stabilizing effect on the numerical formulation.

The FIC approach has been applied to incompressible flows at a range of Reynolds numbers in the past [91–93], but in the present document we intend to investigate the behavior of the formulation as an alternative to large eddy simulation (LES) in a finite element context, as we did for Variational Multiscale (VMS) based formulations in the previous chapter.

The main new feature of the FIC formulation presented here, compared to previous approaches, is the addition of a new dissipative term based on the velocity gradients. This term has an effect on the total dissipation introduced by the numerical formulation and will be shown to improve the accuracy of the solution for the turbulent flow examples considered.

We will start by presenting the general FIC approach in Section 3.2. This approach will be used to obtain a stabilized expression for the linear momentum balance in Section 3.3 and a stabilized continuity equation in Section 3.4. These two expressions will be combined to obtain a discrete formulation in Section 3.5. The presented formulation will then be used to simulate a turbulent channel in Section 3.6, the flow past a cylinder

in Section 3.7 and a solar collector in Section 3.8. Finally, some concluding remarks are presented in Section 3.9.

3.2 FIC formulation

Although our end goal is to apply the FIC formulation to the full Navier-Stokes equations, it is convenient to introduce it first on a simpler problem. We present the first order FIC balance following the approach of [88], by applying it to the $1D$ advection-diffusion equation.

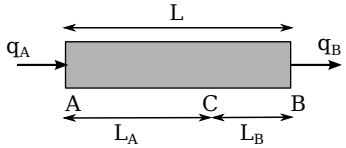


Figure 3.1: Fluxes in a $1D$ domain.

Consider a scalar quantity ϕ advected with a velocity u through the $1D$ domain shown in Fig. 3.1. The domain has a total length $L = x_B - x_A$ and diffusivity coefficient κ. The distribution of ϕ will be the solution of the convection-diffusion equation

$$-u\frac{\mathrm{d}\phi}{\mathrm{d}x} + \kappa\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 0 \qquad \text{in}\,\Omega = [x_A, x_B] \tag{3.1}$$

Furthermore, we can define the flux passing through a point P on the domain as

$$q_P = -u\phi + \kappa\frac{\mathrm{d}\phi}{\mathrm{d}x} \tag{3.2}$$

Consider the flux through the boundary of the domain. Given that the fluxes q_A, q_B entering and exiting it through its extremes must be in equilibrium, we can write

$$q_B - q_A = 0 \tag{3.3}$$

The basic premise of the FIC approach is to write Eq. (3.3) in terms of the flux through an arbitrary point C that lies in the interior of the domain. If the fluxes at A and B are expressed as a Taylor series expansion of the flux at C, we can state

$$q_A = q_C + L_A\frac{\mathrm{d}q}{\mathrm{d}x} + \frac{L_A^2}{2}\frac{\mathrm{d}^2q}{\mathrm{d}x} + O\left(L_A^3\right)$$
$$q_B = q_C - L_B\frac{\mathrm{d}q}{\mathrm{d}x} + \frac{L_B^2}{2}\frac{\mathrm{d}^2q}{\mathrm{d}x} + O\left(L_A^3\right)$$

If we introduce these definitions in Eq. (3.3) we can rearrange the resulting expression to obtain

$$\frac{\mathrm{d}q}{\mathrm{d}x} - \frac{L_B - L_A}{2}\frac{\mathrm{d}^2q}{\mathrm{d}x^2} = 0 \tag{3.4}$$

where we used that $L_B^2 - L_A^2 = (L_B + L_A)(L_B - L_A)$. In FIC formulations, the quantity $h = L_B - L_A$ is defined as the *characteristic length* of the problem. Introducing the definition of the flux in Eq. (3.4) and neglecting third order derivatives we recover the expression

$$-u\frac{\mathrm{d}\phi}{\mathrm{d}x} + \left(\kappa + \frac{uh}{2}\right)\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 0 \tag{3.5}$$

where we have neglected the spatial variation of u and κ.

As the position of point C in the balance domain of Fig. 3.1 is arbitrary, Eq. (3.1) holds for any point within the analysis domain. Comparing Eq. (3.5) to the pointwise balance equation Eq. (3.1), we see that, by enforcing the balance of fluxes on the finite-sized domain $[x_A, x_B]$, we introduce a modified diffusivity, which acts as an additional source of numerical diffusion as long as the characteristic size is chosen such that $uh > 0$. This has a stabilizing effect on the resulting finite element formulation. Note that, in contrast to most stabilization frameworks (such as the VMS formulation presented in the previous chapter), the stabilizing terms appear as a result of a modification of the original PDE and not from a manipulation of the variational form.

The procedure used here to obtain the FIC formulation for the $1D$ advection-diffusion equation can be extended to other problems and multiple dimensions, and is known in the FIC context as first order FIC balance in space. Defining the residual form of our problem as

$$r = -u\frac{\mathrm{d}\phi}{\mathrm{d}x} + \kappa\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} \tag{3.6}$$

we can rearrange the terms in Eq. (3.5) as

$$r - \frac{h}{2}\frac{\mathrm{d}r}{\mathrm{d}x} = 0 \tag{3.7}$$

where we are once more neglecting the spatial variation of u and κ and all third-order derivatives.

The same approach can be extended to multiple dimensions [87]. The vector form of the FIC equation reads

$$\boldsymbol{r} - \frac{h_j}{2}\frac{\partial \boldsymbol{r}}{\partial x_j} = 0 \qquad j \in \{1, n_d\} \tag{3.8}$$

where h_j represents the characteristic length in the j-th coordinate direction. We will use the notation $\boldsymbol{h}$ to denote the vector of characteristic lengths in the different coordinate directions.

This procedure can be directly applied to the momentum equation. We will follow a slightly different approach for the mass conservation equation, originally introduced in [90], which retains higher order terms in the FIC balance to obtain a second order expression.

3.3 Stabilized momentum equation

We present first the FIC stabilized form of the momentum equation. This equation was already introduced in the previous pages, but is stated in Eq. 3.9 for reference. Note that, as we did for the VMS formulation in Chapter 2, we follow the approach of [29] and use the skew-symmetric form of the convective term.

$$\rho\,\partial_t u_i + \rho\left(\frac{1}{2}u_k\frac{\partial u_i}{\partial x_k} + \frac{1}{2}\frac{\partial}{\partial x_k}\left(u_i u_k\right)\right) - \frac{\partial \sigma_{ij}}{\partial x_j} = f_i \qquad \text{in } \Omega\times[0,T) \quad i\in\{1,n_d\} \tag{3.9}$$

For a Newtonian fluid, the stress tensor $\boldsymbol{\sigma}$ can be expressed in terms of the rate of strain tensor $\boldsymbol{\varepsilon}$ as

$$\sigma_{ij} = 2\mu\left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3}\,\delta_{ij}\right) - p\,\delta_{ij} \tag{3.10}$$

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{3.11}$$

We can follow the procedure outlined in the previous section to develop FIC-based stabilized form of the momentum equation. We introduce Eq. (3.10) in Eq. (3.9) and rewrite it in residual form as

$$r_i^m = \rho\partial_t u_i + \rho\left(\frac{1}{2}u_k\frac{\partial u_i}{\partial x_k} + \frac{1}{2}\frac{\partial}{\partial x_k}\left(u_i u_k\right)\right) - 2\mu\frac{\partial}{\partial x_j}\left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3}\delta_{ij}\right) + \frac{\partial p}{\partial x_i} - f_i \tag{3.12}$$

Expressing the balance of linear momentum along each spatial direction i and following the argument of the previous section we can obtain the FIC balance statement for the momentum equation as

$$r_i^m - \frac{h_j}{2}\frac{\partial r_i^m}{\partial x_j} = 0 \quad i,j\in\{1,n_d\} \tag{3.13}$$

where h_j represents the length used to write the balance along the j-th coordinate direction. Note that, in principle, a different vector of characteristic lengths $\boldsymbol{h} = \{h_j\}$ can be used in the balance equation for each momentum component r_i^m.

We consider different possibilities to design $\boldsymbol{h}$. The first possibility is to define the characteristic lengths based on the finite element size along the streamlines of the flow, which results in a method similar to the SUPG formulation [57]. A second option is to base the characteristic length on the size of the element along the direction of the gradient of velocity. This acts as a source of additional diffusion, although the resulting formulation is not stable by itself. Finally, we consider the possibility of mixing both approaches by introducing a combination coefficient. Each approach will be presented in succession in the following pages.

3.3.1 Streamline diffusion formulation

Consider a characteristic length vector $\boldsymbol{h}_u$ aligned on the direction of the flow velocity

$$\boldsymbol{h_u} = h_u \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \tag{3.14}$$

where h_u is the projected length of a given element along the direction of flow, defined by the unit vector $\boldsymbol{u}/\|\boldsymbol{u}\|$. Using this expression, we can rewrite Eq. (3.13) as

$$r_i^m - \frac{h_u}{2\|\boldsymbol{u}\|} u_j \frac{\partial r_i^m}{\partial x_j} \tag{3.15}$$

We can use Eq. (3.15) as the starting point to write a stabilized formulation for the momentum equation. Multiplying by a test function w_i and integrating over the fluid domain we obtain

$$\int_\Omega w_i\, r_i^m - w_i\, \frac{h_u}{2\|\boldsymbol{u}\|} u_k \frac{\partial r_i^m}{\partial x_k}\, \mathrm{d}\Omega = 0 \tag{3.16}$$

It is convenient to integrate by parts the second term in Eq. (3.16). Note that, as the length h_u will be defined as a constant quantity on each element, the boundary integral that appears should be understood as an integral over elemental boundaries.

$$\int_\Omega w_i\, r_i^m\, \mathrm{d}\Omega + \int_\Omega \frac{h_u}{2\|\boldsymbol{u}\|} u_k \frac{\partial w_i}{\partial x_k}\, r_i^m\, \mathrm{d}\Omega - \sum_{(e)} \int_{\Gamma_e} \frac{h_u}{2\|\boldsymbol{u}\|} w_i\, (u_k n_k)\, r_i^m\, \mathrm{d}\Gamma = 0 \tag{3.17}$$

We have neglected the elemental boundary integrals appearing in Eq. (3.17) in the present work. In practice, this is similar to consider that the small scales vanish over element boundaries on VMS formulations. At this point, we introduce the definition of the residual Eq. (3.12) and its gradient in Eq. (3.17). This gives

$$\begin{aligned}
&\int_\Omega w_i\, \rho \left(\partial_t u_i + \frac{1}{2} u_k \frac{\partial u_i}{\partial x_k} \right) \mathrm{d}\Omega - \int_\Omega \frac{1}{2} u_k \frac{\partial w_i}{\partial x_k} \rho u_i \, \mathrm{d}\Omega \\
&\qquad + \int_\Omega 2\mu \frac{\partial w_i}{\partial x_j} \left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3} \delta_{ij} \right) \mathrm{d}\Omega - \int_\Omega \frac{\partial w_i}{\partial x_i}\, p \, \mathrm{d}\Omega \\
&+ \int_\Omega \frac{h_u}{2\|\boldsymbol{u}\|} u_k \frac{\partial w_i}{\partial x_k} \left(\rho \partial_t u_i + \rho \left(\frac{1}{2} u_k \frac{\partial u_i}{\partial x_k} + \frac{1}{2} \frac{\partial}{\partial x_k} (u_i u_k) \right) \right) \mathrm{d}\Omega \\
&\quad + \int_\Omega \frac{h_u}{2\|\boldsymbol{u}\|} u_k \frac{\partial w_i}{\partial x_k} \left(\frac{\partial p}{\partial x_i} - 2\mu \frac{\partial}{\partial x_j} \left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3} \delta_{ij} \right) - f_i \right) \mathrm{d}\Omega = \\
&\qquad \int_\Omega w_i\, f_i \, \mathrm{d}\Omega + \int_{\Gamma_N} w_i\, t_i \, \mathrm{d}\Gamma - \int_{\Gamma_N} \frac{1}{2} \rho w_i\, (u_k n_k)\, u_i \, \mathrm{d}\Gamma
\end{aligned} \tag{3.18}$$

where t_i represents the i-th component of the tractions imposed on the Neumann boundary Γ_N.

Although Eq. (3.18) was developed using a FIC based approach, the final expression is analogous to a SUPG stabilized formulation, with $h_u/2\,\|\boldsymbol{u}\|$ (which has dimensions of time) playing the role of the SUPG stabilization parameter τ.

3.3.2 Gradient diffusion formulation

An alternate approach to Eq. (3.14) is to measure the characteristic length in the direction of the gradient of the i-th component of velocity, $\nabla u_i = \partial u_i/\partial x_j$, given by

$$\boldsymbol{h_{g_i}} = h_{g_i}\frac{\nabla u_i}{\|\nabla u_i\|} \qquad \text{No sum on } i. \tag{3.19}$$

which, as before, can be used to write a FIC balance statement for each component of the momentum equation

$$r_i^m - \frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j}\frac{\partial r_i^m}{\partial x_j} \qquad \text{No sum on } i. \tag{3.20}$$

We can obtain a variational form of the FIC momentum balance equation given by Eq. (3.20) following the same procedure used for the streamline formulation. Multiplying by a test function w_i and integrating over the fluid domain gives

$$\int_\Omega w_i\left(r_i^m - \frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j}\frac{\partial r_i^m}{\partial x_j}\right)\mathrm{d}\Omega = \tag{3.21}$$

$$\int_\Omega w_i\, r_i^m\,\mathrm{d}\Omega - \int_\Omega w_i\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j}\frac{\partial r_i^m}{\partial x_j}\,\mathrm{d}\Omega = 0 \tag{3.22}$$

The first integral in Eq. (3.22) is identical to the first term of Eq. (3.17) and can be developed as in the previous section. We direct our attention towards the second term in Eq. (3.22), which can be integrated by parts as follows

$$\begin{gathered}
-\int_\Omega w_i\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j}\frac{\partial r_i^m}{\partial x_j}\,\mathrm{d}\Omega = \\
\int_\Omega \frac{\partial}{\partial x_j}\left(w_i\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j}\right) r_i^m - \int_\Omega \frac{\partial}{\partial x_j}\left(w_i\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j} r_i^m\right)\mathrm{d}\Omega = \\
\int_\Omega \frac{\partial w_i}{\partial x_j}\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j}\, r_i^m\,\mathrm{d}\Omega + \int_\Omega w_i\,\frac{\partial}{\partial x_j}\left(\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j}\right) r_i^m\,\mathrm{d}\Omega - \\
\int_\Omega \frac{\partial}{\partial x_j}\left(w_i\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_j} r_i^m\right)\mathrm{d}\Omega
\end{gathered} \tag{3.23}$$

From the three terms in the last equality of Eq. (3.23), only the first one will be kept. The second one is neglected as it involves either spatial derivatives of the characteristic length or second derivatives of velocity. The last term can be transformed into a boundary

integral using the divergence theorem, and is dropped for the same reasons we neglected the boundary terms in the streamline formulation. Finally, it is convenient to rewrite the remaining term as

$$\int_\Omega \frac{\partial w_i}{\partial x_j}\left(\frac{h_{g_i} r_i^m}{2\,\|\nabla u_i\|}\delta_{ik}\right)\frac{\partial u_k}{\partial x_j}\,\mathrm{d}\Omega \tag{3.24}$$

If we choose the characteristic length such that $h_{g_i} r_i^m > 0$, Eq. (3.24) describes the discrete version of a non-isotropic Laplacian, where the diffusivity for each coordinate direction is different. The diffusivity coefficient in this case is proportional to the magnitude of the finite element residual on each coordinate direction and exhibits a similar structure to that of a shock-capturing formulation, such as [24]. The numerical diffusion added on each direction is defined by the tensor D_{ij}^g:

$$D_{ij}^g = \frac{h_{g_i} r_i^m}{2\,\|\nabla u_i\|}\delta_{ij} \qquad \text{No sum on } i \tag{3.25}$$

Going back to Eq. (3.22), the weak form for the gradient diffusion formulation reads

$$\int_\Omega w_i\, r_i^m\;\mathrm{d}\Omega + \int_\Omega \frac{\partial w_i}{\partial x_j} D_{ik}^g \frac{\partial u_k}{\partial x_j}\,\mathrm{d}\Omega = 0 \tag{3.26}$$

It must be noted that the formulation of Eq. (3.26) by itself is not sufficient to stabilize convection-dominated flows in general. Therefore, we consider the possibility of combining the present approach with the streamline-based characteristic length.

3.3.3 Combined Approach

As a last possibility, we consider a combined approach including both the stabilizing terms of the streamline-diffusion formulation and the additional diffusion of the gradient formulation. The FIC expression for this case reads

$$r_i^m - \beta\frac{h_u}{2\,\|\boldsymbol{u}\|}u_j\frac{\partial r_i^m}{\partial x_j} - (1-\beta)\,\frac{h_{g_i}}{2\,\|\nabla u_i\|}\frac{\partial u_i}{\partial x_k}\frac{\partial r_i^m}{\partial x_k} \qquad \text{No sum on } i. \tag{3.27}$$

where $\beta \in [0,1]$ is a combination parameter.

The development of the combined formulation follows the steps of each of its components as shown in the previous pages. Therefore, only the final expression for the weak

form, obtained by combining Eq. (3.18) and Eq. (3.26), is given here:

$$\begin{aligned}
&\int_\Omega w_i\,\rho\left(\partial_t u_i + \frac{1}{2}u_k\frac{\partial u_i}{\partial x_k}\right)\,\mathrm{d}\Omega - \int_\Omega \frac{1}{2}u_k\frac{\partial w_i}{\partial x_k}\rho u_i\,\mathrm{d}\Omega \\
&\qquad + \int_\Omega 2\mu\,\frac{\partial w_i}{\partial x_j}\left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3}\,\delta_{ij}\right)\,\mathrm{d}\Omega - \int_\Omega \frac{\partial w_i}{\partial x_i}\,p\,\mathrm{d}\Omega \\
&+ \int_\Omega \beta\frac{h_u}{2\,\|\boldsymbol{u}\|}u_k\frac{\partial w_i}{\partial x_k}\left(\rho\partial_t u_i + \rho\left(\frac{1}{2}u_k\frac{\partial u_i}{\partial x_k} + \frac{1}{2}\frac{\partial}{\partial x_k}\left(u_i u_k\right)\right)\right)\,\mathrm{d}\Omega \\
&\qquad + \int_\Omega \beta\frac{h_u}{2\,\|\boldsymbol{u}\|}u_k\frac{\partial w_i}{\partial x_k}\left(\frac{\partial p}{\partial x_i} - 2\mu\,\frac{\partial}{\partial x_j}\left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3}\,\delta_{ij}\right) - f_i\right)\,\mathrm{d}\Omega \\
&\qquad\qquad + \int_\Omega \frac{\partial w_i}{\partial x_j}\left(1-\beta\right)D^g_{ik}\frac{\partial u_k}{\partial x_j}\,\mathrm{d}\Omega = \\
&\int_\Omega w_i\,f_i\,\mathrm{d}\Omega + \int_{\Gamma_N} w_i\,t_i\,\mathrm{d}\Gamma - \int_{\Gamma_N}\frac{1}{2}\rho w_i\left(u_k n_k\right)u_i\,\mathrm{d}\Gamma
\end{aligned} \tag{3.28}$$

with D^g_{ik} given by Eq. (3.25).

3.3.4 Definition of the stabilization parameters

Eq. (3.28) represents the basic formulation used for the momentum equation in the present chapter. To develop it, we introduced five free parameters (in $3D$): the velocity characteristic length h_u, one gradient characteristic length h_{g_i} along each coordinate direction $i = \{1, 2, 3\}$, and the combination parameter β, which must be defined before the method can be implemented.

Streamline diffusion characteristic length $\mathbf{h_u}$

The characteristic length for the streamline diffusion terms is defined from the size of the element in the direction of velocity $\boldsymbol{u}$. Defining the unit vector in the direction of velocity as $\boldsymbol{e}_u$ and representing the element edge joining nodes a and b with the vector $\boldsymbol{l}_{ab}$, the element length is given by

$$h_u = \max_{\text{edges}}\left\{\boldsymbol{e}_u \cdot \boldsymbol{l}_{ab}\right\} \qquad \text{with} \quad \boldsymbol{e}_u = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \tag{3.29}$$

This is shown graphically for triangles and quadrilaterals in Fig. 3.2, but the same procedure can be applied in $3D$ to define the elemental length using the edges of tetrahedra and hexahedra.

In practice, Eq. (3.29) is evaluated on the integration points of each element. In the case that velocity is (close to) zero on a given point for a given time step, $\boldsymbol{e_u}$ is undefined and this expression can not be used. If this happens, an average element length is used instead. The characteristic length we used in this case will be introduced in Eq. (3.41) for the stabilization of the mass equation.

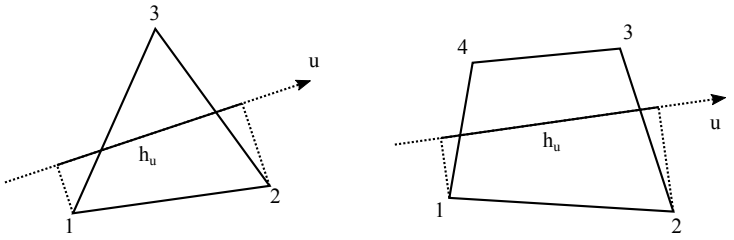


Figure 3.2: Definition of the element lenght h_u for triangles and quadrilaterals.

Gradient diffusion characteristic lengths $\mathbf{h_{g_i}}$

The characteristic element lengths for the gradient term are defined analogously to Eq. (3.29), but using the gradient of the i-th component of velocity to define the direction of projection. The characteristic element length to be used for the i-th coordinate direction is therefore defined as

$$h_{g_i} = \max_{edges} \{ \boldsymbol{e}_{g_i} \cdot \boldsymbol{l}_{ab} \} \qquad \text{with} \quad \boldsymbol{e}_{g_i} = \frac{\nabla u_i}{\|\nabla u_i\|} \qquad \text{No sum on } i. \tag{3.30}$$

Combination parameter $\boldsymbol{\beta}$

In principle, the combination parameter β could take any value in the range $\beta \in [0, 1]$. The limit case $\beta = 1$ results in the classical FIC formulation for the momentum equation, used for example in [88] or [93]. This formulation is very close to the SUPG stabilization, but uses the stabilization parameter of Eq. (3.16), derived from FIC principles. On the other end of the range, $\beta = 0$ implies using the gradient diffusion term exclusively and results in a formulation that is not numerically stable for convection-dominated problems. In the present work, we have found that values of $\beta \geq 0.5$ are typically needed for the problem to be stable for all flow regimes, while values in the range $0.7 \sim 0.9$ typically give the best results.

In addition to defining the value of β as a fixed quantity for the entire simulation, we have also experimented with the possibility of adjusting β dynamically using the local features of the flow. In some simulations we have set a local value for β depending on the directions of velocity and its gradient:

$$\beta_i = \max \left\{ 1 - \frac{u_k \left(\partial u_i / \partial x_k \right)}{\|\boldsymbol{u}\| \, \|\nabla u_i\|}, \beta_m \right\} \qquad \text{No sum on } i. \tag{3.31}$$

where β_m is a minimum value to prevent the loss of stability if the velocity becomes parallel to ∇u_i in some point. Note that the value of β that is obtained using Eq. (3.31) is different for each coordinate direction i. This means that, if this expression is used, both the streamline terms and the gradient terms in Eq. (3.28) are non-isotropic.

3.4 Stabilized mass balance equation

In order to obtain a stabilized formulation for the mass balance equation, the approach used in [90, 91] will be followed. A similiar approach was also applied in [89] for a quasi-incompressible fluid. We introduce the following notation for the mass balance residual:

$$r^c = \frac{\partial u_k}{\partial x_k} = \varepsilon_{kk} \tag{3.32}$$

which can be used to derive the second order FIC balance in space as

$$r^c + \frac{h_j^2}{12}\frac{\partial^2 r^c}{\partial x_j^2} = \varepsilon_{kk} + \frac{h_j^2}{12}\frac{\partial^2 \varepsilon_{kk}}{\partial x_j^2} = 0 \tag{3.33}$$

The expression of second order mass balance was originally used to obtain a stabilized formulation for incompressible flows in [90], where it was derived by expressing the balance of mass within a rectangular domain. In this reference, it is shown that Eq. (3.33) can be obtained by writing the velocities along the boundaries of the rectangle as a Taylor series expansion of the velocity on its center and retaining terms up to third order.

Now the problem consists in obtaining an expression for $\partial^2 \varepsilon_{kk}/\partial x_j^2$ that is useful for the calculation. To do so, we go back to the momentum balance as stated in Eq. (3.12). Assuming that we are in equilibrium, and therefore $r_i^m = 0$, and using the identity

$$\frac{\partial}{\partial x_k}(u_i u_k) = u_k \frac{\partial u_i}{\partial x_k} + u_i \frac{\partial x_k}{\partial x_k} = u_k \frac{\partial u_i}{\partial x_k} + u_i\, \varepsilon_{kk} \tag{3.34}$$

we can rearrange the terms in the momentum balance to read

$$\rho\, \partial_t u_i + \rho\, u_k \frac{\partial u_i}{\partial x_k} + \frac{1}{2}\rho\, \varepsilon_{kk} u_i - 2\mu \frac{\partial}{\partial x_j}\left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3}\delta_{ij}\right) + \frac{\partial p}{\partial x_i} - f_i = 0 \tag{3.35}$$

Moving all terms involving ε_{kk} to the same side of the equality we obtain

$$-\frac{\rho\, u_i}{2}\varepsilon_{kk} - \frac{2\mu}{3}\frac{\partial \varepsilon_{kk}}{\partial x_i} = \rho\, \partial_t u_i + \rho\, u_k \frac{\partial u_i}{\partial x_k} - 2\mu \frac{\partial \varepsilon_{ij}}{\partial x_j} + \frac{\partial p}{\partial x_i} - f_i = \hat{r}_i^m \tag{3.36}$$

where we introduced the notation of $\hat{r}_i^m$ for the right hand side of Eq. (3.36) for convenience. At this point it is helpful to write the first order FIC balance for the continuity equation

$$\varepsilon_{kk} - \frac{h_i}{2}\frac{\partial \varepsilon_{kk}}{\partial x_i} = 0 \tag{3.37}$$

and use it to express ε_{kk} in terms of its derivative in Eq. (3.36)

$$-\left(\frac{\rho\, u_j h_j}{4} + \frac{2\mu}{3}\right)\frac{\partial \varepsilon_{kk}}{\partial x_i} = \hat{r}_i^m \tag{3.38}$$

We can introduce Eq. (3.38) in Eq. (3.33) to write

$$r^c + \frac{h_i^2}{12}\frac{\partial}{\partial x_i}\left(\frac{\partial \varepsilon_{kk}}{\partial x_i}\right) = r^c - \frac{h_i^2}{12}\frac{\partial}{\partial x_i}\left(\left(\frac{\rho\, u_j h_j}{4} + \frac{2\mu}{3}\right)^{-1} \hat{r}_i^m\right) = 0 \tag{3.39}$$

Neglecting the spatial variation of the product $h_j u_j$, we take the coefficient that multiplies $\hat{r}_i^m$ out of the derivative, obtaining

$$r^c - \frac{h_i^2}{12}\left(\frac{\rho\, u_j h_j}{4} + \frac{2\mu}{3}\right)^{-1} \frac{\partial \hat{r}_i^m}{\partial x_i} = 0 \tag{3.40}$$

Eq. (3.40) expresses the basic FIC mass balance statement used in the present work. We will simplify it slightly by using an average characteristic length as done in [89], which allows us to combine the two coefficients in Eq. (3.40) in a single isotropic stabilization parameter τ_c

$$\tau_c = \left(\frac{3\rho\, \|\boldsymbol{u}\|}{h} + \frac{8\mu}{h^2}\right)^{-1} \tag{3.41}$$

where we use the norm of the velocity and the average element length h, which is calculated as the square root of the elemental area in $2D$ or the cubic root of the elemental volume in $3D$.

Using τ_c we can write the final FIC balance statement for the incompressibility equation as

$$r^c - \tau_c \frac{\partial \hat{r}_i^m}{\partial x_i} = 0 \tag{3.42}$$

We can multiply Eq. (3.42) by a test function q and integrate over the fluid domain Ω to obtain the weak form of the equation

$$\int_\Omega q\, r^c \,\mathrm{d}\Omega - \int_\Omega q \tau_c \frac{\partial}{\partial x_i}\left(\rho \partial_t u_i + \rho u_k \frac{\partial u_i}{\partial x_k} - 2\mu \frac{\partial}{\partial x_j}\varepsilon_{ij} + \frac{\partial p}{\partial x_i} - f_i\right) \mathrm{d}\Omega = 0 \tag{3.43}$$

It is convenient to integrate by parts the second integral in Eq. (3.43) to reduce the order of the derivatives involved. As in the momentum equation, the boundary terms resulting from this operation are neglected in the present work, obtaining the expression

$$\int_\Omega q \frac{\partial u_i}{\partial x_i} \,\mathrm{d}\Omega + \int_\Omega \frac{\partial q}{\partial x_i} \tau_c \left(\rho \partial_t u_i + \rho u_k \frac{\partial u_i}{\partial x_k} - 2\mu \frac{\partial}{\partial x_j}\varepsilon_{ij} + \frac{\partial p}{\partial x_i} - f_i\right) \mathrm{d}\Omega = 0 \tag{3.44}$$

Eq. (3.44) represents a stabilized formulation for the continuity equation, similar to that obtained in GLS formulations [56]. Note that the stabilization parameter τ_c, defined in Eq. (3.41), has the same structure as the classical SUPG or GLS characteristic time τ and the static version of the parameter τ_1 used for the VMS formulation in Chapter 2.

In addition to the formulation given by Eq. (3.44), we have also tested a variant involving the projection of $\hat{r}_i^m$. Consider the following modified version of Eq. (3.42)

$$\frac{\partial u_i}{\partial x_i} + \tau_c \frac{\partial}{\partial x_i}\left(\hat{r}_i^m - \pi_i\right) = 0 \tag{3.45}$$

where $\boldsymbol{\pi}$ represents the L_2 projection onto the finite element grid of $\hat{\boldsymbol{r}}^m$, that is to say, the solution of

$$\int_\Omega w_i \pi_i \,\mathrm{d}\Omega = \int_\Omega w_i \hat{r}_i^m \,\mathrm{d}\Omega \tag{3.46}$$

This formulation results in the following weak form, again neglecting boundary terms, which substitutes Eq. (3.44).

$$\int_\Omega q \frac{\partial u_i}{\partial x_i}\,\mathrm{d}\Omega + \int_\Omega \frac{\partial q}{\partial x_i}\tau_c \left(\rho \partial_t u_i + \rho u_k \frac{\partial u_i}{\partial x_k} - 2\mu \frac{\partial}{\partial x_j}\varepsilon_{ij} + \frac{\partial p}{\partial x_i} - f_i + \pi_i\right)\mathrm{d}\Omega = 0 \tag{3.47}$$

We will use Eq. (3.47) as the reference formulation in the following, with the understanding that any terms involving π_i can be dropped to recover the formulation without projections.

3.5 Finite element formulation

Combining the stabilized momentum equation given by Eq. (3.28) and that of Eq. (3.47) for the continuity equation we obtain the complete stabilized weak form of the problem, which we used to develop a finite element formulation.

In the present work we restrict ourselves to linear finite elements, using triangular and quadrilateral elements in $2D$ or tetrahedra and hexahedra in $3D$. This means that all terms involving second derivatives of velocity in Eqs. (3.28) and (3.47) will be neglected, as they are identically zero when using our interpolation. The full formulation, without second order terms, is given by

Momentum

$$\begin{aligned}
&\int_\Omega w_i\,\rho\left(\partial_t u_i + \frac{1}{2}u_k\frac{\partial u_i}{\partial x_k}\right)\mathrm{d}\Omega - \int_\Omega \frac{1}{2}u_k\frac{\partial w_i}{\partial x_k}\rho u_i\,\mathrm{d}\Omega \\
&\qquad + \int_\Omega 2\mu\,\frac{\partial w_i}{\partial x_j}\left(\varepsilon_{ij} - \frac{\varepsilon_{kk}}{3}\,\delta_{ij}\right)\mathrm{d}\Omega - \int_\Omega \frac{\partial w_i}{\partial x_i}\,p\,\mathrm{d}\Omega \\
&+ \int_\Omega \beta\frac{h_u}{2\,\|\boldsymbol{u}\|}u_k\frac{\partial w_i}{\partial x_k}\left(\rho\partial_t u_i + \rho\left(\frac{1}{2}u_k\frac{\partial u_i}{\partial x_k} + \frac{1}{2}\frac{\partial}{\partial x_k}\left(u_i u_k\right)\right)\right)\mathrm{d}\Omega \\
&+ \int_\Omega \beta\frac{h_u}{2\,\|\boldsymbol{u}\|}u_k\frac{\partial w_i}{\partial x_k}\left(\frac{\partial p}{\partial x_i} - f_i\right)\mathrm{d}\Omega + \int_\Omega \frac{\partial w_i}{\partial x_j}\left(1-\beta\right)D^g_{ik}\frac{\partial u_k}{\partial x_j}\,\mathrm{d}\Omega = \\
&\qquad \int_\Omega w_i\,f_i\,\mathrm{d}\Omega + \int_{\Gamma_N} w_i\,t_i\,\mathrm{d}\Gamma - \int_{\Gamma_N}\frac{1}{2}\rho w_i\left(u_k n_k\right)u_i\,\mathrm{d}\Gamma
\end{aligned} \tag{3.48}$$

Mass balance

$$\int_\Omega q\,\frac{\partial u_i}{\partial x_i}\,\mathrm{d}\Omega + \int_\Omega \frac{\partial q}{\partial x_i}\tau_c\left(\rho\partial_t u_i + \rho u_k\frac{\partial u_i}{\partial x_k} + \frac{\partial p}{\partial x_i} - f_i + \pi_i\right)\mathrm{d}\Omega = 0 \tag{3.49}$$

with

$$D^g_{ik} = \frac{h_{g_i} r_i^m}{2\left\|\nabla u_i\right\|}\delta_{ik} \qquad \tau_c = \left(\frac{3\rho\left\|\boldsymbol{u}\right\|}{h} + \frac{8\mu}{h^2}\right)^{-1} \qquad \beta \in [0,1]$$

3.5.1 Spatial discretization

We introduce a finite element discretization Ω_h of the problem domain Ω. Using this discrete representation, the problem variables $\boldsymbol{u}$ and p can be represented using a finite element interpolation as

$$\boldsymbol{u}_h = \sum_a^{n_n} \boldsymbol{N}_a\left(\boldsymbol{x}\right)\boldsymbol{u}_a \qquad p_h = \sum_a^{n_n} N_a\left(\boldsymbol{x}\right)p_a \tag{3.50}$$

where n_n represents the number of nodes in the finite element mesh, $\boldsymbol{u}_a$ and p_a are the variables evaluated at node a, $N_a\left(\boldsymbol{x}\right)$ is the standard linear finite element function associated to node a and

$$\boldsymbol{N}_a = \begin{bmatrix} N_a & 0 & 0 \\ 0 & N_a & 0 \\ 0 & 0 & N_a \end{bmatrix}$$

Furthermore, we introduce the notation $\boldsymbol{U}$, $\dot{\boldsymbol{U}}$ and $\boldsymbol{P}$ to indicate the vectors of nodal values for velocity, acceleration and pressure, respectively. Given that the variational form of the problem, represented by Eqs. (3.48) and (3.49), must hold for all admissible test functions and that the set of finite element shape functions $\{N_a\}$ constitutes a basis of the interpolation space, we can obtain a system of equations by imposing that the variational form of the problem must hold for each basis function N_a. This system can be expressed in matrix form as

$$\begin{bmatrix} \boldsymbol{M}+\boldsymbol{M}_K & \boldsymbol{0} \\ \boldsymbol{M}_D & \boldsymbol{0} \end{bmatrix}\begin{bmatrix} \dot{\boldsymbol{U}} \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{C}+\boldsymbol{K}+\boldsymbol{S}_K+\boldsymbol{D}_G & \boldsymbol{G}+\boldsymbol{S}_G \\ \boldsymbol{D}+\boldsymbol{S}_D & \boldsymbol{L} \end{bmatrix}\begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}+\boldsymbol{S}_F+\boldsymbol{T} \\ \boldsymbol{S}_Q+\boldsymbol{S}_\Pi \end{bmatrix} \tag{3.51}$$

The different matrices in Eq. (3.51) represent the discrete form of the terms in Eqs. (3.48) and (3.49). Each of them can be built by the assembly of elemental contributions. For an element containing N nodes, an elemental matrix $\boldsymbol{A}^e$ can be defined using $N \times N$ blocks $\boldsymbol{A}^e_{ab}$, where a and b are local node indices. The individual blocks for the different

matrices and vectors can be defined as

$$\boldsymbol{M}^e_{ab} = \int_{\Omega_e} \rho \boldsymbol{N}^T_a \boldsymbol{N}_b \, \mathrm{d}\Omega \tag{3.52}$$

$$\boldsymbol{C}^e_{ab} = \int_{\Omega_e} \rho \frac{1}{2} \left(\boldsymbol{N}^T_a u_k \frac{\partial \boldsymbol{N}_b}{\partial x_k} - \left(u_k \frac{\partial \boldsymbol{N}_a}{\partial x_k} \right)^T \boldsymbol{N}_b \right) \mathrm{d}\Omega \tag{3.53}$$

$$\boldsymbol{G}^e_{ab\,i} = - \int_{\Omega_e} \left(\frac{\partial \boldsymbol{N}_a}{\partial x_i} \right)^T N_b \, \mathrm{d}\Omega \tag{3.54}$$

$$\boldsymbol{D}^e_{ab\,i} = \int_{\Omega_e} N_a \frac{\partial \boldsymbol{N}_b}{\partial x_i} \, \mathrm{d}\Omega = - \left(\boldsymbol{G}^e_{ba\,i} \right)^T \tag{3.55}$$

$$\boldsymbol{F}^e_a = \int_{\Omega_e} \boldsymbol{N}^T_a \boldsymbol{f} \, \mathrm{d}\Omega \tag{3.56}$$

$$\boldsymbol{T}^e_a = \int_{\Gamma_N} \boldsymbol{N}^T_a \left(\boldsymbol{t} - \rho \frac{1}{2} \left(u_k n_k \right) \boldsymbol{N}_b \right) \mathrm{d}\Gamma \tag{3.57}$$

Introducing the strain rate-velocity matrix $\boldsymbol{B}_a$ for node a and the constitutive matrix $\boldsymbol{C}_\mu$

$$\boldsymbol{B}^T_a = \begin{bmatrix} \dfrac{\partial N_a}{\partial x} & 0 & 0 & \dfrac{\partial N_a}{\partial y} & 0 & \dfrac{\partial N_a}{\partial z} \\ 0 & \dfrac{\partial N_a}{\partial y} & 0 & \dfrac{\partial N_a}{\partial x} & \dfrac{\partial N_a}{\partial z} & 0 \\ 0 & 0 & \dfrac{\partial N_a}{\partial z} & 0 & \dfrac{\partial N_a}{\partial y} & \dfrac{\partial N_a}{\partial x} \end{bmatrix} \tag{3.58}$$

$$\boldsymbol{C}_\mu = \begin{bmatrix} 4\mu/3 & -2\mu/3 & -2\mu/3 & 0 & 0 & 0 \\ -2\mu/3 & 4\mu/3 & -2\mu/3 & 0 & 0 & 0 \\ -2\mu/3 & -2\mu/3 & 4\mu/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{3.59}$$

the viscosity term in Eq. (3.48) can be expressed in discrete form as the viscous stress matrix

$$\boldsymbol{K}^e_{ab} = \int_{\Omega_e} \boldsymbol{B}^T_a \boldsymbol{C}_\mu \boldsymbol{B}_b \, \mathrm{d}\Omega \tag{3.60}$$

The stabilization terms in Eq. (3.48) and Eq. (3.49) give rise to the following matrices

$$\boldsymbol{M}^e_{K\,ab} = \int_{\Omega_e} \rho\beta \frac{h_u}{2\,\|\boldsymbol{u}\|} \left(u_k \frac{\partial \boldsymbol{N}_a}{\partial x_k}\right)^T \boldsymbol{N}_b \,\mathrm{d}\Omega \tag{3.61}$$

$$\boldsymbol{M}^e_{D\,ab\,i} = \int_{\Omega_e} \rho\tau_c \left(\frac{\partial N_a}{\partial x_i}\right)^T \boldsymbol{N}_b \,\mathrm{d}\Omega \tag{3.62}$$

$$\boldsymbol{S}^e_{K\,ab} = \int_{\Omega_e} \rho\beta \frac{h_u}{2\,\|\boldsymbol{u}\|} \left(u_k \frac{\partial \boldsymbol{N}_a}{\partial x_k}\right)^T \left(u_l \frac{\partial \boldsymbol{N}_b}{\partial x_l}\right) \,\mathrm{d}\Omega \tag{3.63}$$

$$\boldsymbol{S}^e_{G\,ab\,i} = \int_{\Omega_e} \beta \frac{h_u}{2\,\|\boldsymbol{u}\|} \left(u_k \frac{\partial \boldsymbol{N}_a}{\partial x_k}\right)^T \frac{\partial N_b}{\partial x_i} \,\mathrm{d}\Omega \tag{3.64}$$

$$\boldsymbol{D}^e_{G\,ab} = \int_{\Omega_e} (1-\beta) \left(\frac{\partial \boldsymbol{N}_a}{\partial x_i}\right)^T D^g_{ik} \left(\frac{\partial \boldsymbol{N}_b}{\partial x_k}\right) \,\mathrm{d}\Omega \tag{3.65}$$

$$\boldsymbol{S}^e_{D\,ab\,i} = \int_{\Omega_e} \tau_c N_a \frac{\partial \boldsymbol{N}_b}{\partial x_i} \,\mathrm{d}\Omega \tag{3.66}$$

$$\boldsymbol{L}^e_{ab} = \int_{\Omega_e} \tau_c \left(\frac{\partial N_a}{\partial x_k}\right)^T \frac{\partial N_b}{\partial x_k} \,\mathrm{d}\Omega \tag{3.67}$$

$$\boldsymbol{S}^e_{F\,a} = \int_{\Omega_e} \beta \frac{h_u}{2\,\|\boldsymbol{u}\|} \left(u_k \frac{\partial \boldsymbol{N}_a}{\partial x_k}\right)^T \boldsymbol{f} \,\mathrm{d}\Omega \tag{3.68}$$

$$\boldsymbol{S}^e_{Q\,a} = \int_{\Omega_e} \tau_c \frac{\partial N_a}{\partial x_i} f_i \,\mathrm{d}\Omega \tag{3.69}$$

$$\boldsymbol{S}^e_{\Pi\,a} = -\int_{\Omega_e} \tau_c \frac{\partial N_a}{\partial x_i} \pi_i \,\mathrm{d}\Omega \tag{3.70}$$

In the following, Eq. (3.51) will be expressed using the compact notation

$$\tilde{\boldsymbol{M}} \begin{bmatrix} \dot{\boldsymbol{U}} \\ \boldsymbol{0} \end{bmatrix} + \tilde{\boldsymbol{C}} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} = \tilde{\boldsymbol{F}} \tag{3.71}$$

If projections are used in the stabilization of the incompressibility equation, an additional system has to be solved to determine the nodal values of the projection variables $\boldsymbol{\pi}$. The equations for the projection can be obtained from the discrete version of Eq. (3.46), resulting in

$$\boldsymbol{M}_\Pi \boldsymbol{\Pi} = \boldsymbol{R}_\Pi \tag{3.72}$$

where $\boldsymbol{\Pi}$ is the array of nodal values for the projection variables and

$$\boldsymbol{M}^e_{\Pi\,ab} = \int_{\Omega_e} \boldsymbol{N}_a^T \boldsymbol{N}_b \,\mathrm{d}\Omega \tag{3.73}$$

$$\boldsymbol{R}^e_{\Pi\,a} = \int_{\Omega_e} \boldsymbol{N}_a^T \hat{\boldsymbol{r}}^m \,\mathrm{d}\Omega \tag{3.74}$$

Note that the assembly of elemental contributions given by Eq. (3.73) results in a dense matrix. In practice, the system matrix $\boldsymbol{M}_\Pi$ in Eq. (3.72) is approximated by a diagonal mass matrix for efficiency.

3.5.2 Time integration and linearization

To solve the problem described by Eqs. (3.48) and (3.49), we first need introduce a time discretization to express the nodal accelerations $\dot{\boldsymbol{U}}$ in terms of the nodal velocities $\boldsymbol{U}$. As in Chapter 2, we use the Bossak scheme to obtain the time-discrete problem. Referring the reader to Section 2.5 for the details on the method, here we present only the final expression, which can be expressed as

$$\begin{aligned}\left(\frac{1-\alpha_B}{\gamma_N \Delta t}\tilde{\boldsymbol{M}} + \tilde{\boldsymbol{C}}\right)\begin{bmatrix}\boldsymbol{U}_{n+1}\\ \boldsymbol{P}_{n+1}\end{bmatrix} = \\ \tilde{\boldsymbol{F}} - \frac{1-\alpha_B}{\gamma_N \Delta t}\tilde{\boldsymbol{M}}\begin{bmatrix}\boldsymbol{U}_{n}\\ \boldsymbol{0}\end{bmatrix} + \left\{(1-\alpha_B)\left(\frac{1}{\gamma_N}-1\right)+\alpha_B\right\}\tilde{\boldsymbol{M}}\begin{bmatrix}\dot{\boldsymbol{U}}_{n}\\ \boldsymbol{0}\end{bmatrix}\end{aligned} \tag{3.75}$$

with $\alpha_B = -0.3$, $\Gamma_N = 1/2 - \alpha_B$ and

$$\dot{\boldsymbol{U}}_{\boldsymbol{n}} = \frac{1}{\gamma_N \Delta t}\left(\boldsymbol{U}_n - \boldsymbol{U}_{n-1}\right) - \left(\frac{1}{\gamma_N}-1\right)\dot{\boldsymbol{U}}_{\boldsymbol{n-1}} \tag{3.76}$$

The only step left to finalize the finite element solver is to introduce a linearization for Eq. (3.75). Both the system matrix and the right-hand side term contain sources of non-linearity in the form of terms that depend on the current values of the variables. This includes all terms involving the convective term $u_k \partial u_i / \partial x_k$, stabilization terms due to the dependence of the different stabilization coefficients on the local velocity u_i and the gradient diffusion term, which involves both the momentum residual r_i^m and the velocity gradients ∇u_i.

As in Chapter 2, we rewrite Eq. (3.75) in residual form and introduce a linearization so that the unknowns can be obtained by iteratively solving a linear system of equations. Defining the approximation to the value at time step $n+1$ after i non-linear iterations as $\boldsymbol{U}^i_{n+1}$, the residual form of Eq. (3.75) is given by

$$\begin{aligned}\boldsymbol{R}\left(\boldsymbol{U}^i_{n+1}, \boldsymbol{P}^i_{n+1}\right) = \tilde{\boldsymbol{F}} - \frac{1-\alpha_B}{\gamma_N \Delta t}\tilde{\boldsymbol{M}}\begin{bmatrix}\boldsymbol{U}_{n}\\ \boldsymbol{0}\end{bmatrix} + \\ \left\{(1-\alpha_B)\left(\frac{1}{\gamma_N}-1\right)+\alpha_B\right\}\tilde{\boldsymbol{M}}\begin{bmatrix}\dot{\boldsymbol{U}}_{n}\\ \boldsymbol{0}\end{bmatrix} - \left(\frac{1-\alpha_B}{\gamma_N \Delta t}\tilde{\boldsymbol{M}} + \tilde{\boldsymbol{C}}\right)\begin{bmatrix}\boldsymbol{U}^i_{n+1}\\ \boldsymbol{P}^i_{n+1}\end{bmatrix}\end{aligned} \tag{3.77}$$

Our problem now consists in finding $\boldsymbol{U}^{i+1}_{n+1}$, $\boldsymbol{P}^{i+1}_{n+1}$ such that $\boldsymbol{R}^{i+1}_{n+1} = \boldsymbol{0}$. As before, using Picard iterations we obtain the iterative scheme

$$-\left(\frac{1-\alpha_B}{\gamma_N \Delta t}\tilde{\boldsymbol{M}} + \tilde{\boldsymbol{C}}\right)\begin{bmatrix}\boldsymbol{\delta U^i}\\ \boldsymbol{\delta P^i}\end{bmatrix} = \boldsymbol{R}\left(\boldsymbol{U}^i_{n+1}, \boldsymbol{P}^i_{n+1}\right) \tag{3.78}$$

this problem is solved iteratively until convergence in terms of the increments $\boldsymbol{\delta U^i}$ and $\boldsymbol{\delta P^i}$ or the residual vector $\boldsymbol{R}\left(\boldsymbol{U}^i_{n+1}, \boldsymbol{P}^i_{n+1}\right)$.

3.5.3 Summary of the formulation

Starting from the weak form described by Eq. (3.48) and Eq. (3.49), we have introduced a finite element discretization in space and a time discretization based on the Bossak method. Additionally, a Picard linearization has been used to obtain a linear system of equations to be solved iteratively, given by Eq. (3.78). To summarize, the complete FIC solution procedure is presented in compact form as Algorithm 3.1.

Algorithm 3.1 FIC incompressible flow solver.

for n = 0 nsteps **do**
  $n = n + 1$,$t = t + \Delta t$
  while $\left\|\boldsymbol{R}^i_{n+1}\right\| \leq$ tol **do**
    $i = i + 1$
    for all elements **do**
      if Using dynamic procedure for β **then**
        Compute β_i according to Eq. (3.31)
      end if
      Evaluate local contributions using Eqs. (3.52)–(3.70).
      Assemble local contributions to the linear system of Eq. (3.78).
    end for
    Solve Eq. (3.78) for $\boldsymbol{\delta U^i}$, $\boldsymbol{\delta P^i}$.
    Update variables $\boldsymbol{U}^{i+1}_{n+1}$,$\boldsymbol{P}^{i+1}_{n+1}$.
    if Using projections **then**
      for all elements **do**
        Assemble projection problem using Eqs. (3.73) and (3.74).
      end for
      Obtain new values for the projection by solving Eq. (3.72).
    end if
  end while
  Calculate $\dot{\boldsymbol{U}}^{i+1}_{n+1}$ according to Eq. (3.76).
end for

This formulation has been implemented within the Kratos Multiphysics code [34], a software framework for the development of finite element solvers. The code is prepared to work in a parallel environment, as will be presented in Chapter 4. This has proved essential to perform the larger simulations in a reasonable time, which were run using the Gottfried cluster of the North-German Supercomputing Alliance (HLRN).

3.6 Turbulent channel flow

The flow in a plane turbulent channel is a classic turbulence benchmark and represents a challenging problem for LES formulations, due to the dependence of the vortex size to the distance to the wall [125]. It has been studied for a wide range of Reynolds numbers, but we direct our attention to the moderate value of $\mathrm{Re}_\tau = 395$. There is an extensive bibliography regarding this case, with a comprehensive set of statistical data obtained from direct numerical simulations by Moser *et al.* in [81] and different studies in which the problem was modelled using stabilized VMS-based formulations, both using classical finite elements such as in [45] or [32] and using isogeometric elements [9] or [3].

This Reynolds number is very convenient because it allows using a mesh size in the inertial subrange, even close to the wall, while keeping the computational cost under control. At higher Reynolds numbers, the number of elements required to have a grid size on the inertial subrange increases prohibitively and some type of wall model is usually preferred to reduce the required number of elements (see for example [100] or [17]).

The plane turbulent channel problem simulates a flow driven by a fixed pressure gradient between two parallel infinite walls. Defining the distance between the two walls as 2δ, the problem is formulated in terms of the wall friction τ_w and the friction velocity u_τ

$$\tau_w = -\delta \frac{\mathrm{d}P}{\mathrm{d}x} = \rho u_\tau^2 \quad \Rightarrow \quad u_\tau = \left(-\frac{\delta}{\rho} \frac{\mathrm{d}P}{\mathrm{d}x} \right)^{\frac{1}{2}} \tag{3.79}$$

where $\mathrm{d}P/\mathrm{d}x$ is the imposed pressure gradient. Using the definitions of Eq. (3.79), the turbulent channel problem can be characterized by the friction Reynolds number Re_τ, defined as

$$\mathrm{Re}_\tau = \frac{u_\tau \delta}{\nu} \tag{3.80}$$

which is set to $\mathrm{Re}_\tau = 395$ for the present simulation. The results are presented in terms of the dimensionless distance to the wall $y+ = u_\tau y/\nu$.

To perform the simulation at the desired Reynolds number, we have selected the following parameters:

$$\rho = 1\, Kg/m^3 \quad \nu = 1.472 \times 10^{-4}\, m^2/s \quad \delta = 1\, m \quad \frac{\mathrm{d}P}{\mathrm{d}x} = -3.372040 \times 10^{-3}\, N/m^2 \tag{3.81}$$

3.6.1 Problem definition

We model a domain defined by $[0, 2\pi] \times [-\delta, \delta] \times [0, 2\pi/3]$ in the stream-wise, wall-normal and cross-stream directions respectively, using the same domain as in [9]. Zero velocity Dirichlet conditions are assigned on the wall sides and periodic boundary conditions are used in the remaining directions.

The problem has been modeled using linear hexahedral and tetrahedral elements. In the first case, a grid of 64^3 elements has been used. Mesh nodes were placed regularly along the stream-wise and cross-stream directions while, in the wall-normal direction, a weighting function is used to move the nodes closer to the wall. The location y^i of the i-*th* node in the wall-normal direction is chosen as

$$y^i = \delta \frac{\tanh\left(w\left(\frac{2i}{n_{el}} - 1\right)\right)}{\tanh\left(w\right)} \qquad i \in [0, n_{el}] \tag{3.82}$$

with $w = 2.432$ for $n_{el} = 64$ and $w = 2.927$ for $n_{el} = 32$, chosen so that the first node in the mesh is always at a dimensionless distance to the wall $y+ = 1$. For the tetrahedral cases, a mesh with the same nodal positions is used, but each hexahedra is split into six tetrahedra.

The time step for the simulation is chosen as $\Delta t = 0.04\,s$ which, according to the analysis in [49], should be sufficient to reproduce the features of the flow. We use the expected average velocity profile as the initial condition, adding a random fluctuation to destabilize the solution. Once a fully turbulent flow develops, the flow is left to evolve until it reaches a statistically homogeneous solution. With this, different averages and correlations are calculated using the approach presented in Chapter 4. Statistical results were collected at the integration points of each element, averaging over time and over planes parallel to the walls.

We have performed multiple simulations using these settings to study the behaviour of different variants of the FIC formulation presented in the preceding pages. Note that all turbulent channel flow cases were run using the projections for the mass stabilization in Eq. (3.47).

3.6.2 Fixed combination parameter

The first simulations were performed using the FIC formulation with a fixed combination parameter, set to $\beta = 0.8$. The average stream-wise velocity $\langle u \rangle$, relative to the friction velocity u_τ, is shown in Fig. 3.3, compared to the DNS data of Moser *et al.* [81] for the same Reynolds number.

The velocity variances for the same simulation are shown in Fig. 3.4. In addition to the variances in each coordinate direction we also present the total turbulence kinetic energy, defined as

$$k = \frac{1}{2}\left(\langle u'u' \rangle + \langle v'v' \rangle + \langle w'w' \rangle\right) \tag{3.83}$$

Additionally, we measured the dissipation of average stream-wise linear momentum in the wall-normal direction. We know from studying the RANS momentum equation that the average shear stress in the xy plane can be written as (see for example [125]

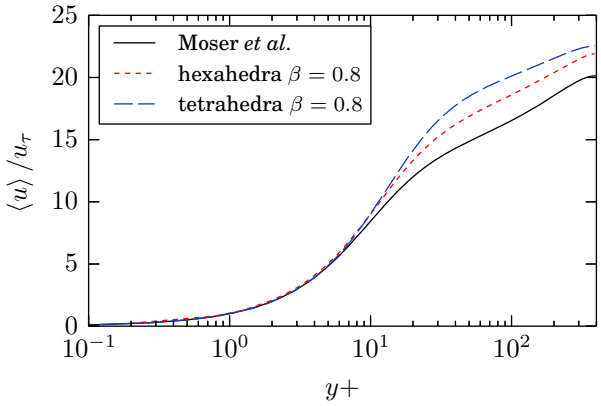


Figure 3.3: Channel flow – average stream-wise velocity profiles obtained using $\beta = 0.8$, compared to the DNS data of Moser *et al.* [81].

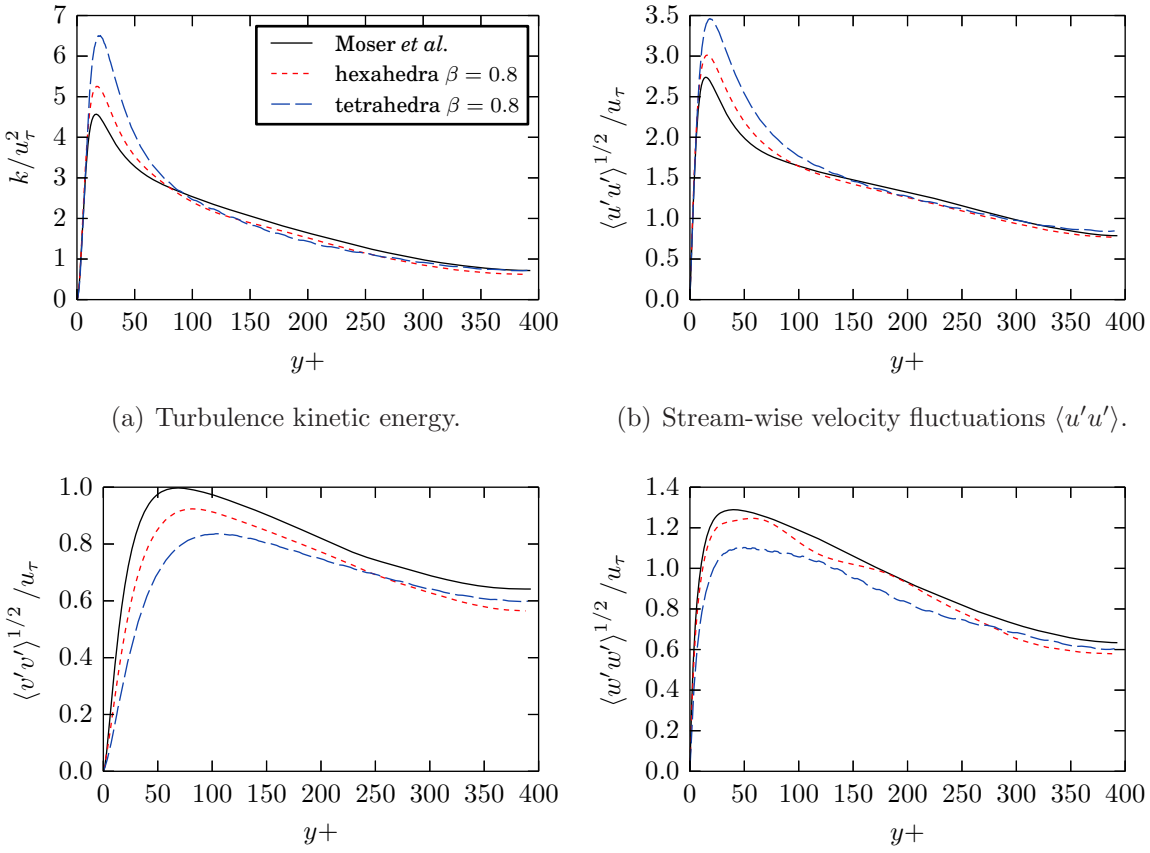
or [101])

$$\langle \tau_{xy} \rangle = \rho \langle u'v' \rangle + \mu \frac{\partial \langle u \rangle}{\partial y} = \tau_w \frac{y}{\delta} \tag{3.84}$$

where the first term of the middle equality represents the Reynolds stresses in the xy plane and the second the viscous dissipation due to the average velocity. This decomposition is shown in Fig. 3.5 for the simulation performed using hexahedral elements. As the addition of the two terms is close to the expected straight line, we consider that the flow is in statistical equilibrium.

We observe that the results obtained with linear hexahedra are closer to the expected values than those obtained with linear tetrahedra in all cases. This was to be expected, as hexahedra use trilinear shape functions, which define a richer interpolation than the linear functions used in tetrahedra.

In light of the fact that the formulation we are using has a free parameter, β, which is set *a priori*, we are interested in studying how the solution is dependent of its value. To investigate this, we simulated the problem with the same settings, changing only the value of β. The results obtained using a tetrahedral mesh with 64 divisions along each coordinate direction are shown in Fig. 3.6, where the statistics obtained with $\beta = 0.8$ and a tetrahedral mesh in the previous test, corresponding to the dashed curve in Fig. 3.3 and Fig. 3.4 are compared to those obtained with the same mesh and different values of β. It is observed that there is not a large amount of variation between the different cases, although the $\beta = 0.5$ case tends to display a lower level of velocity fluctuations in all directions. This suggests that small values of β, which give more weight to the gradient diffusion term, result in a solution that is more diffusive overall.



(a) Turbulence kinetic energy.

(b) Stream-wise velocity fluctuations $\langle u'u' \rangle$.

(c) Wall-normal velocity fluctuations $\langle v'v' \rangle$.

(d) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

Figure 3.4: Channel flow – turbulence kinetic energy and Reynolds stresses obtained using $\beta = 0.8$, compared to Moser *et al.* [81].

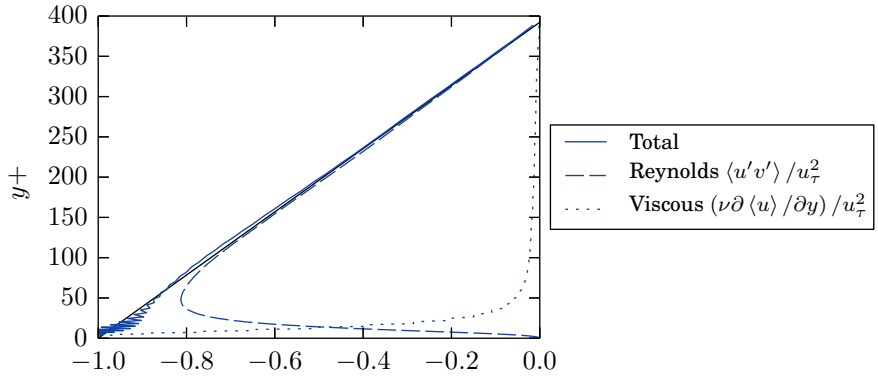
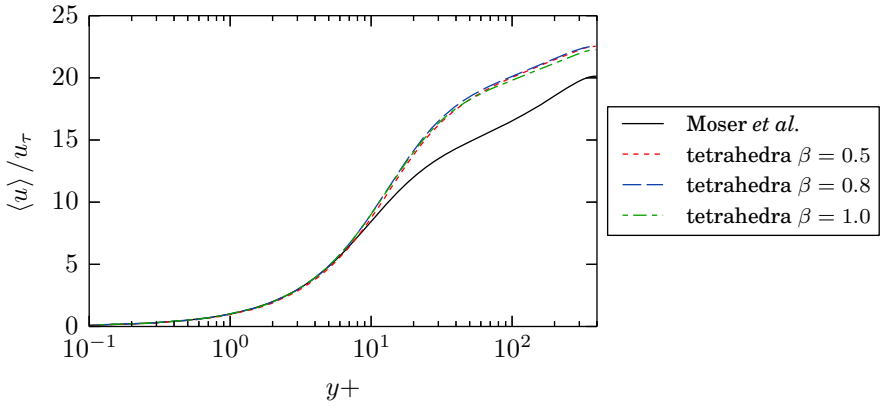
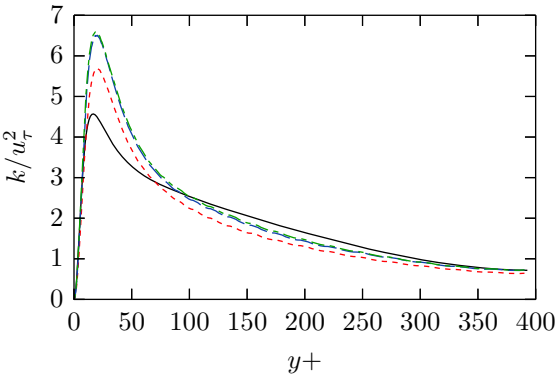


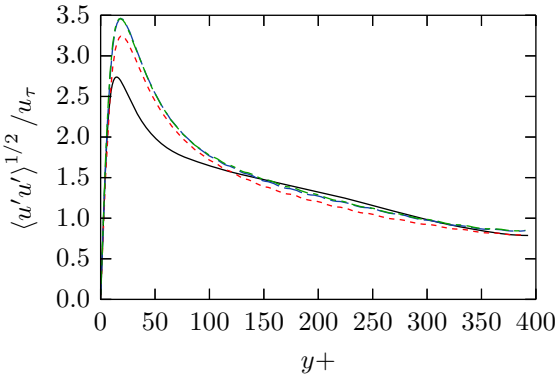
Figure 3.5: Channel flow – average stress $\langle \tau_x y \rangle$ profile obtained using hexahedra and fixed $\beta = 0.8$.



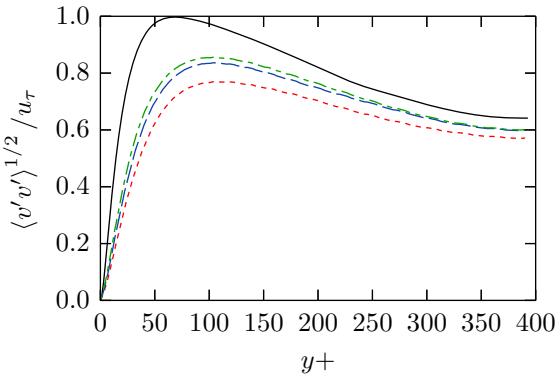
(a) Average stream-wise velocity



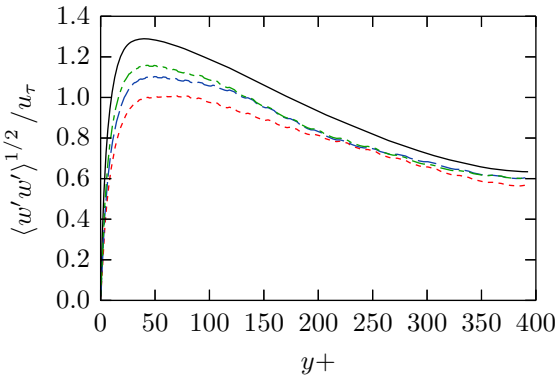
(b) Turbulence kinetic energy.



(c) Stream-wise velocity fluctuations $\langle u'u' \rangle$.



(d) Wall-normal velocity fluctuations $\langle v'v' \rangle$.



(e) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

Figure 3.6: Channel flow – velocity average and variances for a range of values of β, using tetrahedral meshes.

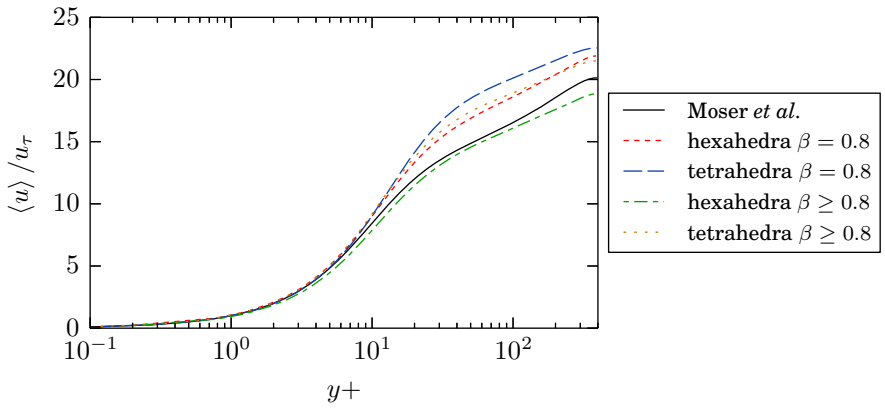


Figure 3.7: Channel flow – average stream-wise velocity profiles using a fixed or dynamic combination parameter.

3.6.3 Variable combination parameter

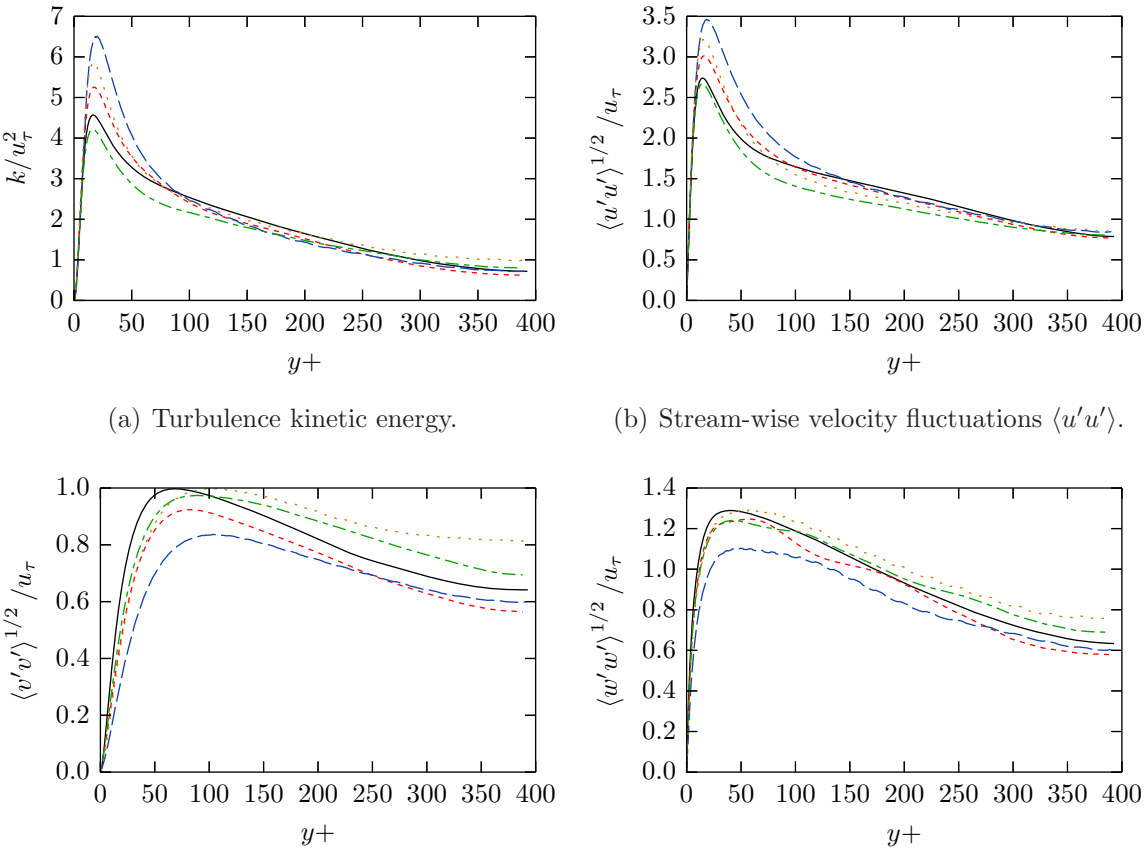
The next set of tests was performed using Eq. (3.31) to assign a value to the combination parameter β, while keeping the remaining simulation settings as in the fixed beta case. The minimum value of the coefficient was set to $\beta \geq 0.8$ for the first tests, which produced the average velocity distribution presented in Fig. 3.7 and the velocity variances shown in Fig. 3.8, where we compare them to to the results obtained for a fixed coefficient in previous simulations.

We can see in the figures that the average velocity profiles are generally lower to those obtained in the fixed β cases and closer to those obtained from DNS data. Conversely, the velocity fluctuations measured in the stream-wise direction are somewhat lower than in the fixed β simulations, resulting in a turbulence kinetic energy that is much closer to that obtained from DNS simulations.

As in the previous case, we are also interested in quantifying the impact that the choice of a limit value for the combination coefficient has in the obtained solution. To test its influence, we ran several simulations for tetrahedra (Fig. 3.9) and hexahedra (Fig. 3.10), changing the limit value for β. We consider that the results show minor variations depending on the choice of parameter, at least for large values of β.

We also studied the sensitivity of the solution to the choice of mesh size. The results obtained using a coarser mesh of 32^3 hexhahedra or 6×32^3 tetrahedra are compared in Fig. 3.11 to those obtained with the meshes used in the previous examples. As before, there is a clear difference in the behaviour of tetrahedra and hexahedra. In particular, it seems that the coarser tetrahedral mesh is insufficient to reproduce the features of the problem, resulting in a significantly larger average velocity.

As a final validation, we compared our results to those obtained using a GLS formulation on the same finite element mesh. The results of this test are shown in Fig. 3.12.



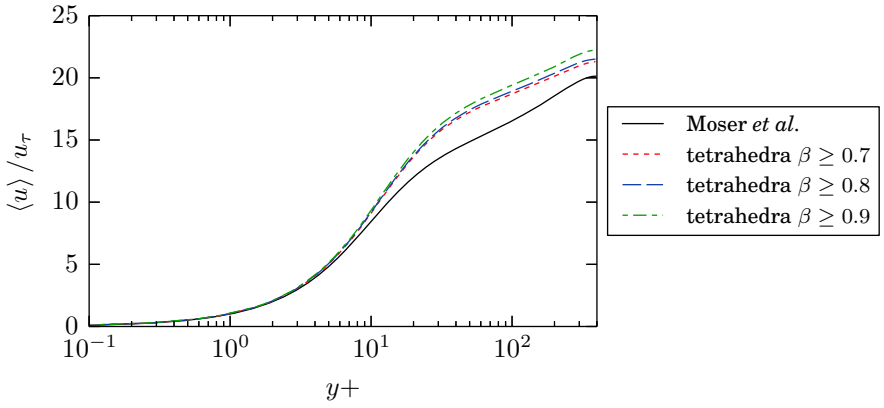
(a) Turbulence kinetic energy.

(b) Stream-wise velocity fluctuations $\langle u'u' \rangle$.

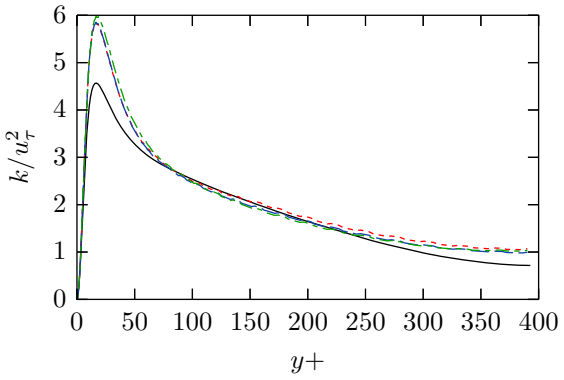
(c) Wall-normal velocity fluctuations $\langle v'v' \rangle$.

(d) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

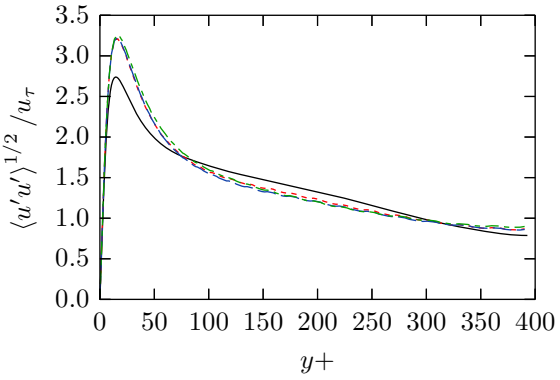
Figure 3.8: Channel flow – turbulence kinetic energy and Reynolds stresses using a fixed or dynamic combination parameter. See legend in Fig. 3.7.



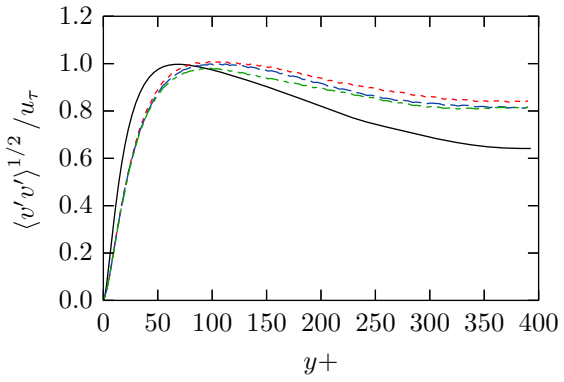
(a) Average stream-wise velocity



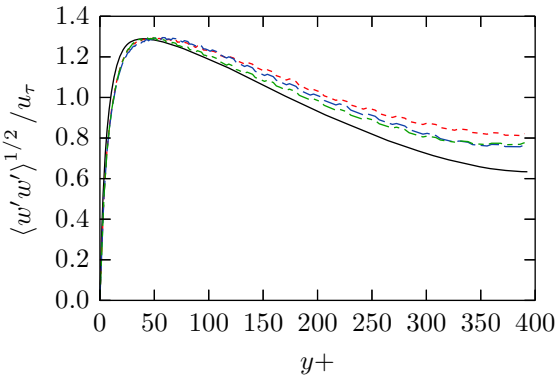
(b) Turbulence kinetic energy.



(c) Stream-wise velocity fluctuations $\langle u'u' \rangle$.

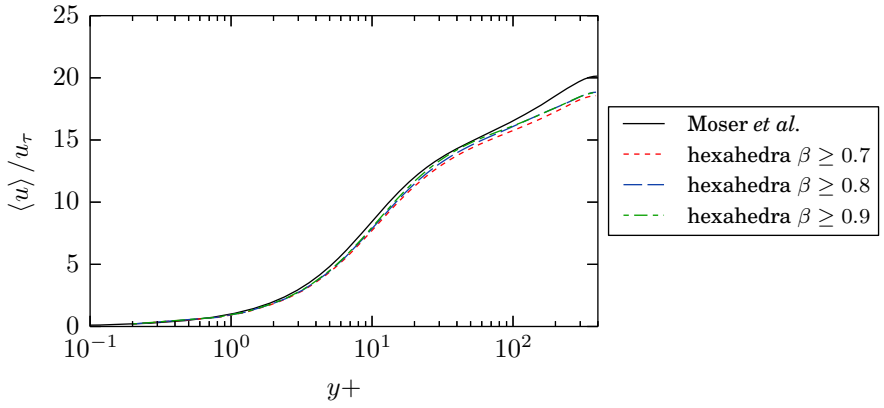


(d) Wall-normal velocity fluctuations $\langle v'v' \rangle$.

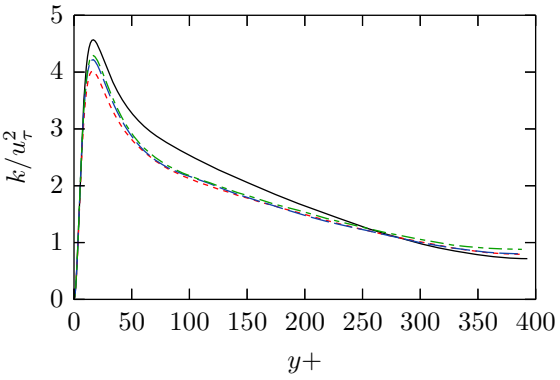


(e) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

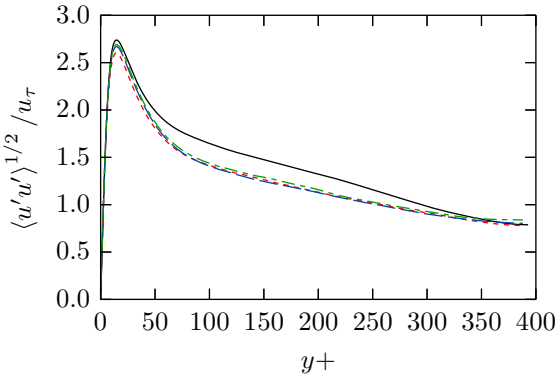
Figure 3.9: Channel flow – velocity average and variances obtained using linear tetrahedra and different limits for the dynamic combination parameter.



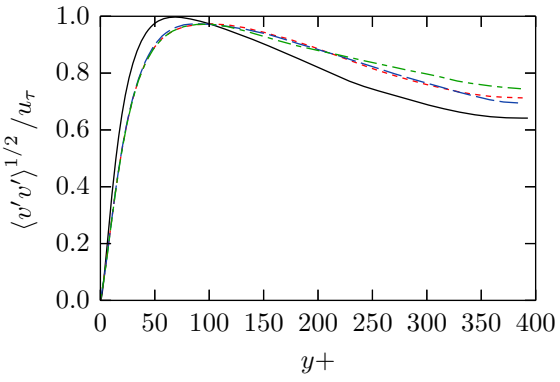
(a) Average stream-wise velocity



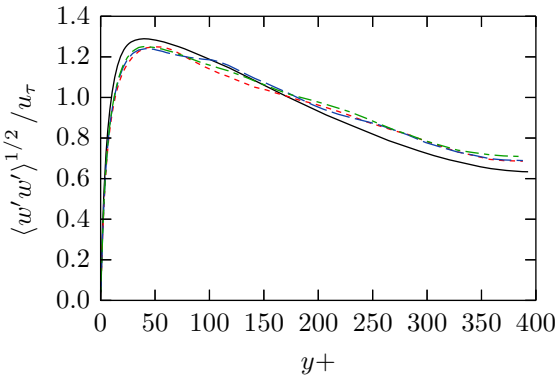
(b) Turbulence kinetic energy.



(c) Stream-wise velocity fluctuations $\langle u'u' \rangle$.

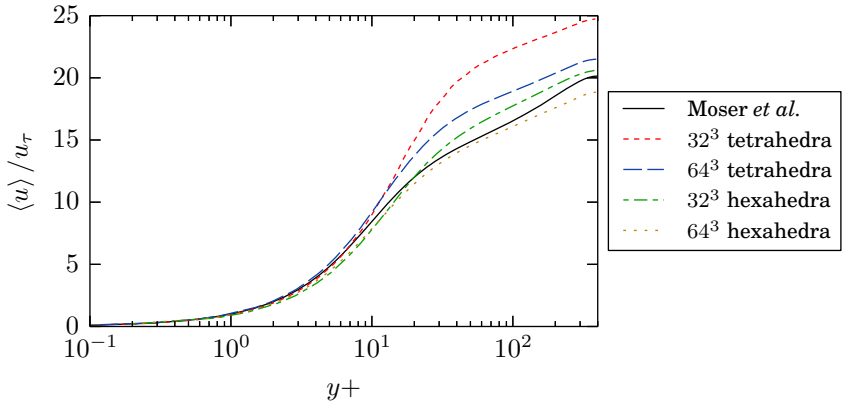


(d) Wall-normal velocity fluctuations $\langle v'v' \rangle$.

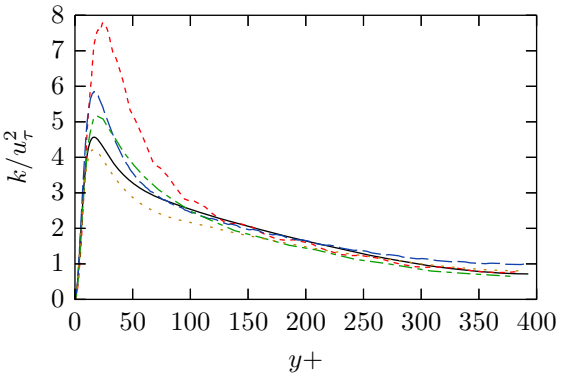


(e) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

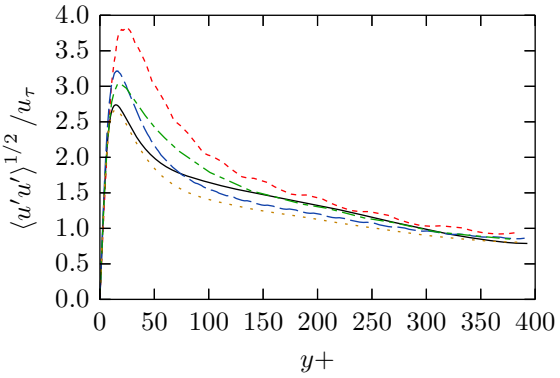
Figure 3.10: Channel flow – velocity average and variances obtained using linear hexahedra and different limits for the dynamic combination parameter.



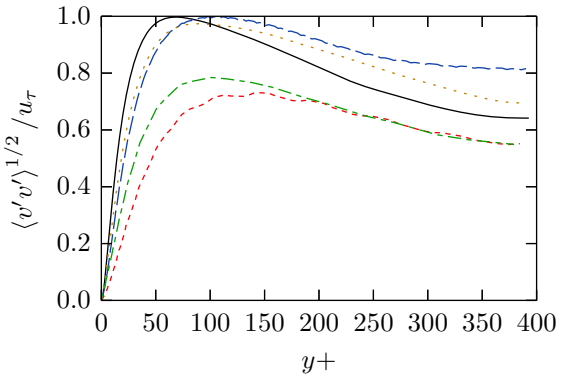
(a) Average stream-wise velocity



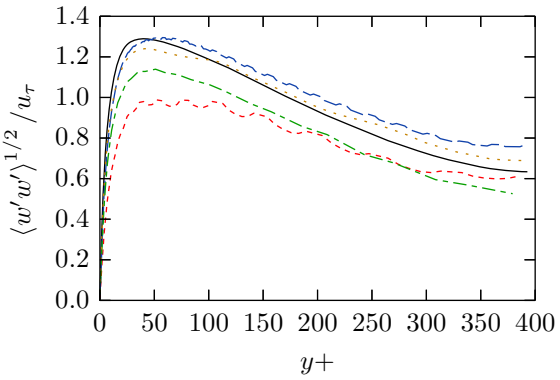
(b) Turbulence kinetic energy.



(c) Stream-wise velocity fluctuations $\langle u'u' \rangle$.

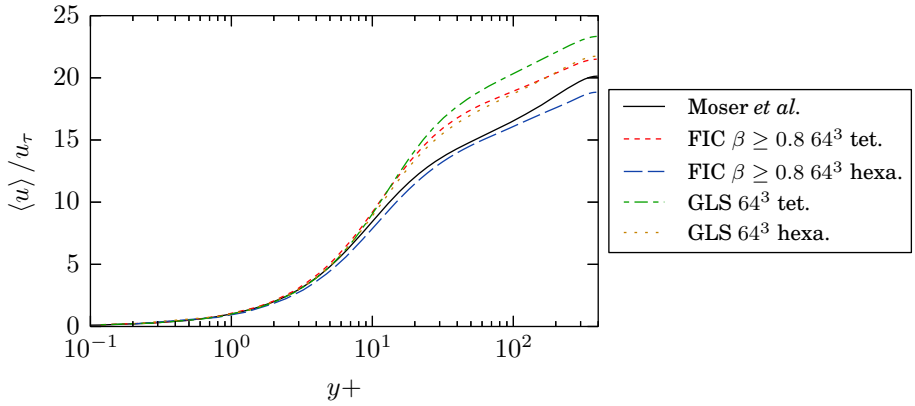


(d) Wall-normal velocity fluctuations $\langle v'v' \rangle$.

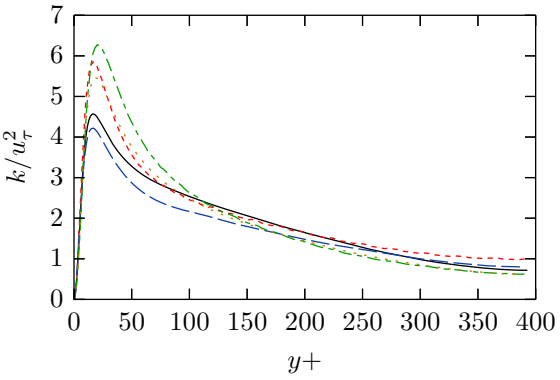


(e) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

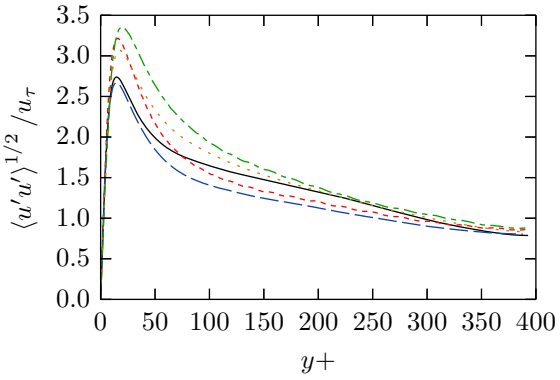
Figure 3.11: Channel flow – velocity average and variances obtained using different grid sizes (all results with dynamic $\beta \geq 0.8$).



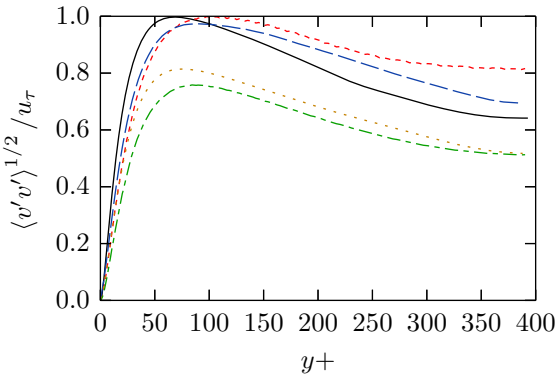
(a) Average stream-wise velocity



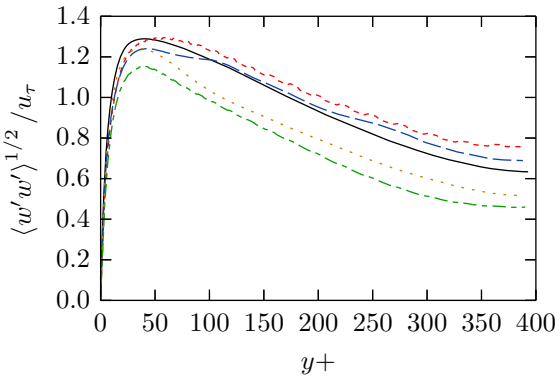
(b) Turbulence kinetic energy.



(c) Stream-wise velocity fluctuations $\langle u'u' \rangle$.



(d) Wall-normal velocity fluctuations $\langle v'v' \rangle$.



(e) Cross-stream velocity fluctuations $\langle w'w' \rangle$.

Figure 3.12: Channel flow – velocity average and variances obtained using FIC with a dynamic combination coefficient $\beta \geq 0.8$ or GLS.

3.6.4 Summary of the results

To conclude the analysis of the turbulent channel flow at $\mathrm{Re}_\tau = 395$ test case we summarize the main results. A first general observation is that hexahedra produce much more accurate solutions than tetrahedra for a given mesh size. A difference in accuracy was expected, as the interpolation obtained using hexahedra is richer, but we have found it to be large in our mesh studies. An observation of Fig. 3.11 suggests that 6×64^3 tetrahedra are required to obtain results comparable to 32^3 hexahedra, which represents 48 times more elements in total. In spite of this, we will continue to use tetrahedra in the following examples, due to their flexibility in meshing complex geometries.

As the formulation we are proposing has a free parameter, the combination parameter β, we wanted to study the sensitivity to its value. In the cases where the parameter was fixed throughout the simulation, the general trend is to obtain lower variances the smaller the coefficient, which corresponds to giving more weight to the gradient diffusion term (see Fig. 3.6). This suggests that the gradient diffusion term introduces a significant amount of numerical dissipation, producing more homogeneous solutions. However, it must be noted that the average velocity profile is much less sensitive to the choice of parameter, producing very similar results in all cases.

If the combination parameter is set on each element according to the local weighting function of Eq. (3.31), the obtained velocity profiles are generally lower than those obtained with a fixed parameter and measured variances show a better agreement with the DNS data, specially in the $\langle u'u' \rangle$ correlation, which is the larger contribution to the total turbulence kinetic energy (see Fig. 3.8). In this case, the only external parameter is the minimum admissible value of β, but our experiments show that the results are not very sensitive to this parameter, at least if it is large enough, as can be observed in Fig. 3.9 for tetrahedra and Fig. 3.10 for hexahedra.

Finally, we compared our approach to using a standard GLS stabilization, which, for linear elements, is equivalent to the Q-ASGS formulation presented in Chapter 2. The results, shown in Fig. 3.12, suggest that the formulation we propose results in a closer approximation to DNS data than the reference for a given mesh size. We propose two reasons a justification for this result. On one hand, we introduce a new term, the gradient diffusion, which has been shown to introduce an additional source of dissipation. On the other, the formulation we propose, unlike GLS, does not include a *div-div* term, which was shown in [32] and in our own results in the previous chapter to have a negative impact in the turbulent channel case.

Based on the results obtained in this set of tests, we conclude that setting the combination parameter locally, limited to a minimum value of $\beta \geq 0.8$, is the variant that better approximates the reference solution. As a result, we will adopt this formulation for the remaining cases.

3.7 Flow around a cylinder

The flow over a circular cylinder is a classical problem in CFD simulations, which has been studied extensively, both experimentally and numerically, as can be verified for example in the review of [82]. Flow over circular cylinders exhibits a variable behavior depending on the Reynolds number, due to the different vortex shedding mechanisms that develop on the wake [69]. As a benchmark example for the FIC formulation, we studied the case corresponding to a Reynolds number $\mathrm{Re} = 3900$ based on the diameter of the cylinder and the inflow velocity. This case was studied experimentally in [94] and numerically in [66] or [12]. In this section, we will use a numerical set up similar to that of [66] and compare our results to those presented in that reference.

We simulated the flow over a cylinder with dimensionless diameter $D = 1$ using a domain of $30\,D \times 30\,D$, centered on the cylinder, in the plane normal to the cylinder's axis, and $W = 3.14\,D$ in the span-wise direction. The simulation is run for 160 dimensionless time units (made dimensionless using the inflow velocity U_∞ and the diameter D) with a dimensionless time step $\delta t = 0.1$, which should provide sufficient resolution to capture the main vortex shedding frequency.

The domain for the problem is presented in Figure 3.13. Consider the axes x, y and z, aligned in the stream-wise, cross-stream and span-wise directions respectively, and let u, v and w, be the components of the flow velocity in each of the three coordinate directions. We are interested in measuring the velocity history in selected planes in the wake of the cylinder, chosen to coincide with those reported in the reference.

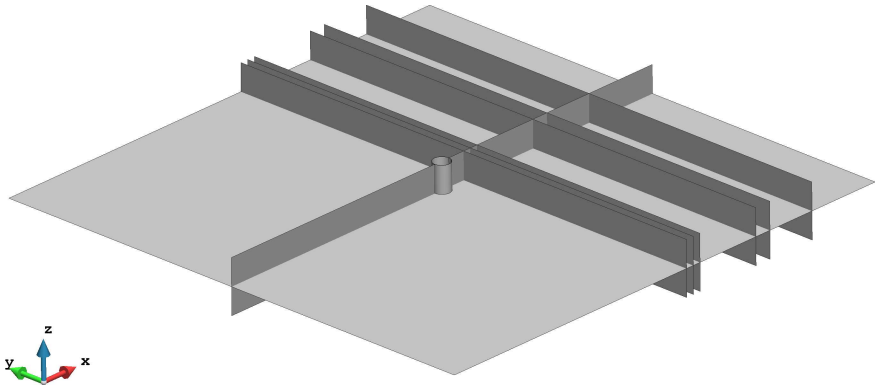


Figure 3.13: Flow around a cylinder – simulation domain and measurement planes.

Linear tetrahedral elements have been used to mesh the domain, with sizes ranging between $0.03\,D$ near the cylinder to $0.5\,D$ in the far regions. This resulted in a total of 1.74 million nodes and 10 million elements. The velocity is fixed to a constant $U_\infty = 1\,m/s$ for the inlet and to zero on the cylinder surface. Periodic boundary conditions have been used in the span-wise direction, while a no-penetration condition $v = 0$ has been imposed in the far sides on the cross-stream direction. The combination

parameter for the FIC formulation is set locally for each element using the dynamic formulation, with a limit value of $\beta \geq 0.8$.

The instantaneous stream-wise velocity field on the central x–y plane at the end of the simulation is shown in Fig. 3.14. The velocities on the x–z plane for the same time instant are shown in Fig. 3.15. From the stream-wise velocity component u in Fig 3.15(a), the formation of a recirculation zone just after the cylinder can be observed. Similarly, in Fig. 3.15(b), which shows the instantaneous cross-stream component of velocity v, the alternating direction of the velocity suggests the formation of a vortex trail.

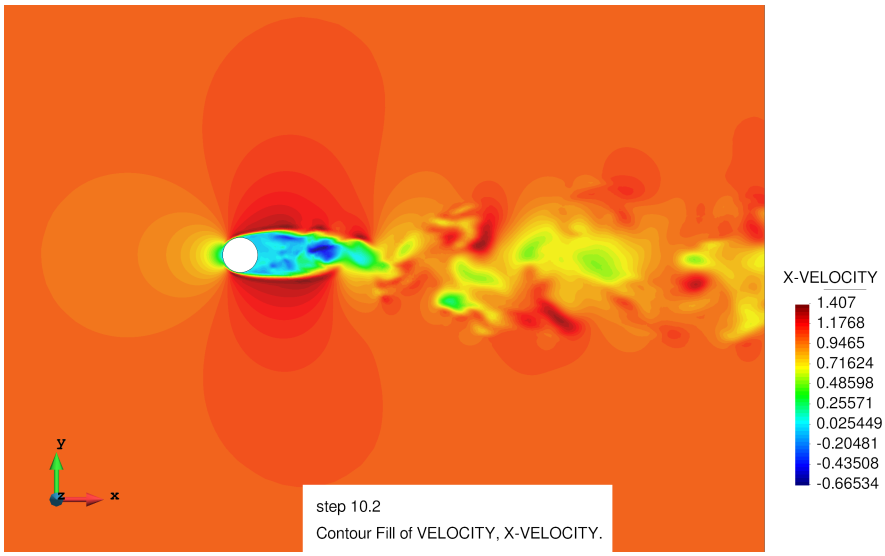
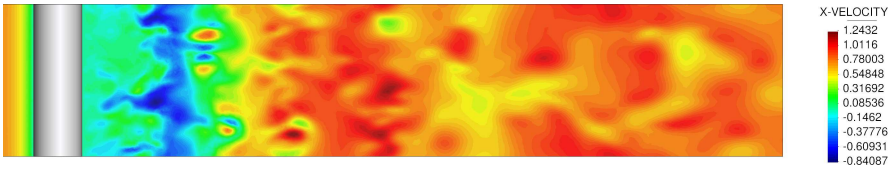


Figure 3.14: Flow around a cylinder – instantaneous stream-wise velocity u on the x–y midplane.

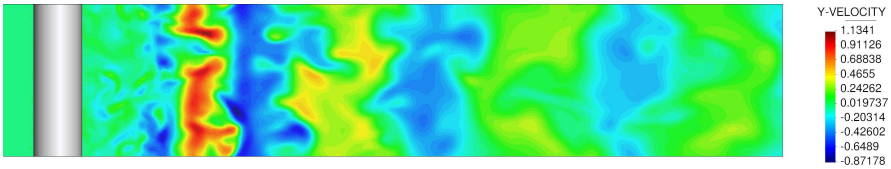
While the instantaneous velocity distributions give us a qualitative idea of the flow, we are interested in studying the statistics of the cylinder wake, which can give us a more quantitative idea of the quality of the simulation. We computed the average velocity in the stream-wise and cross-stream directions on different y–z sections (that is, normal to the mean flow) on the wake of the cylinder. The results in terms of averages, compared to those of [66], are shown in Fig. 3.16 for the near wake and in Fig. 3.17 for the far wake.

Observing the results, we can see that we obtain a close agreement with the reference, although the formation of the wake is slightly delayed when compared to the reference. This can be seen by considering that the average velocity defect on the wake should start as a deep U-shaped trough just behind the cylinder, where the average flow in the recirculation zone is very small or negative on average, and become wider and shallower (closer to the inflow velocity U_∞) as the wake develops. In general, our profiles are below the expected curve.

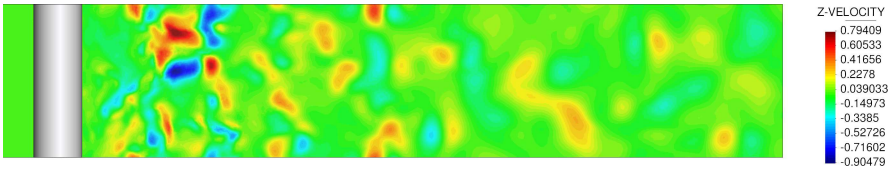
In addition, we have also computed the variance of the stream-wise velocity $\langle u'u' \rangle$,



(a) Stream-wise velocity u.

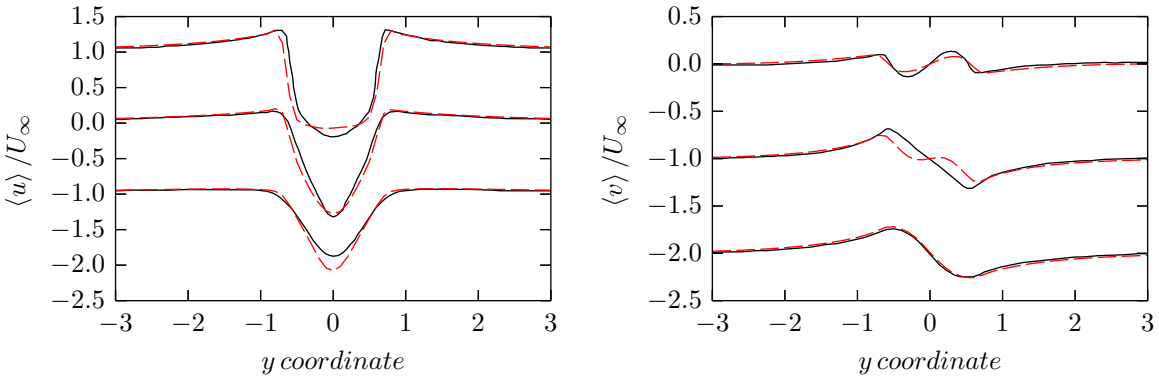


(b) Cross-stream velocity v.



(c) Span-wise velocity w.

Figure 3.15: Flow around a cylinder – instantaneous velocities on the x–z midplane.



(a) Stream-wise velocity u.

(b) Cross-stream velocity v.

Figure 3.16: Flow around a cylinder – average velocities in the near wake. Reference (——), present work (– – –). From top to bottom: $x/D = 1.06$, $x/D = 1.54$, $x/D = 2.02$.

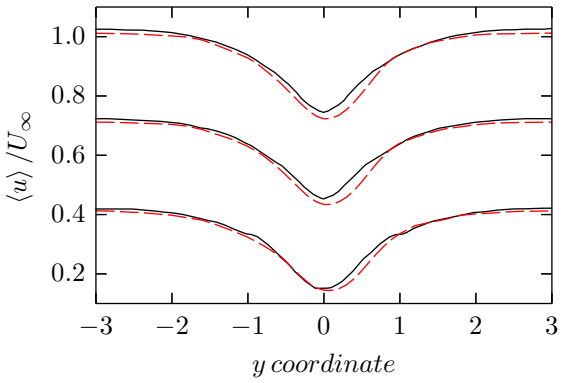


Figure 3.17: Flow around a cylinder – average stream-wise velocity u in the far wake. Reference (——), present work (– – –). From top to bottom: $x/D = 6.0$, $x/D = 7.0$, $x/D = 10.0$.

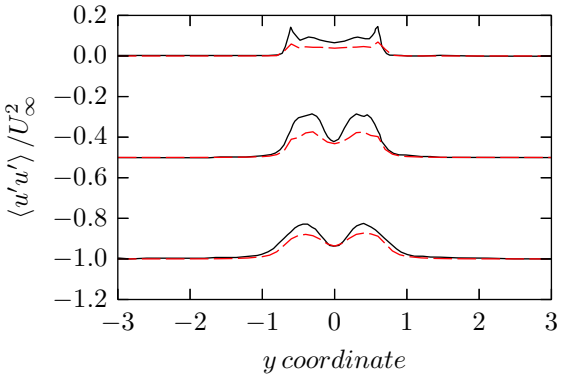


Figure 3.18: Flow around a cylinder – $\langle u'u' \rangle$ correlation in the near wake. Reference (——), present work (– – –). From top to bottom: $x/D = 1.06$, $x/D = 1.54$, $x/D = 2.02$.

shown in Fig. 3.19(a) for the near wake and Fig. 3.18 for the far wake. The obtained variance is smaller in general than expected, which suggests a smoother flow and a more diffusive solution. The cross-correlation $\langle u'v' \rangle$ is shown in Fig. 3.19(b) on the same planes where it was reported in the reference. This result shows some irregularity, in our solution as well as in the reference, which might be reduced with a longer simulation time.

Finally, we have computed the drag coefficient C_D and the Strouhal number St, which represents the dimensionless vortex shedding frequency, as given

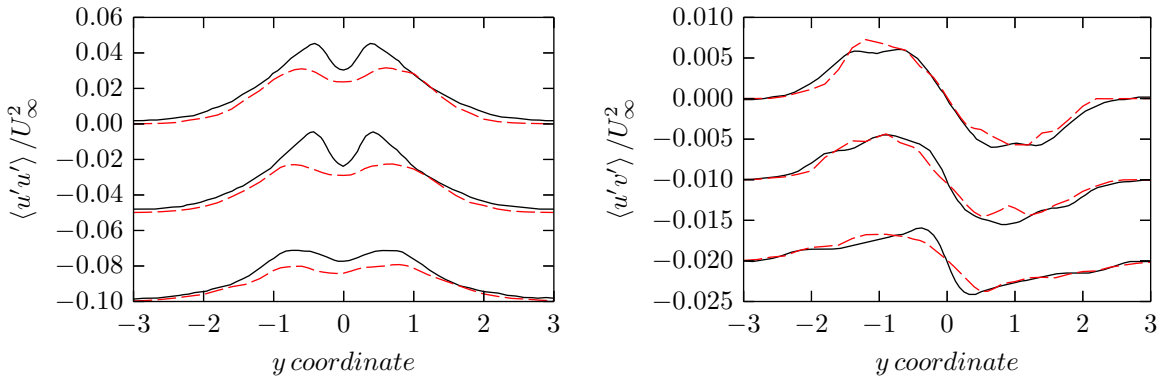
$$C_D = \frac{\langle R_x \rangle}{\frac{1}{2}\rho U_\infty D W} \qquad \mathrm{St} = \frac{f\, D}{U_\infty} \tag{3.85}$$

where $\langle R_x \rangle$ is the average force applied by the fluid on the surface of the cylinder in the stream-wise direction and f is the vortex-shedding frequency on the cylinder tail, which we have calculated from the lift force history $R_y(t)$. The results, compared to those reported in [66], are reported in Table 3.1.

Analysis	C_D	St
Present simulation	1.09	0.217
Numerical [66]	1.04	0.210
Experimental (reported in [66])	0.99 ± 0.05	0.215 ± 0.005

Table 3.1: Flow around a cylinder – flow parameters.

The results obtained for this case show in general good agreement to reference values in terms of the average solution. Variances, while in qualitative agreement with



(a) $\langle u'u' \rangle$ correlation.

(b) $\langle u'v' \rangle$ correlation.

Figure 3.19: Flow around a cylinder – velocity correlations in the far wake. Reference (——), present work (– – –). From top to bottom: $x/D = 6.0$, $x/D = 7.0$, $x/D = 10.0$.

the expected results in terms of spatial distribution, tend to be underestimated. This suggests that the obtained solution has larger dissipation than required, smoothing out peaks in the fluctuating solution. It would be interesting to repeat the same simulation with hexahedra or with a finer tetrahedral grid, to see which fraction of this dissipation is due to mesh resolution or to the formulation itself.

3.8 Flow around a solar collector

As a final example we wanted to test the capabilities of the formulation when applied to an industrial problem. For this, we simulated the wind flow around a parabolic trough solar collector and compared it to experimental data. A parabolic trough is an array of parabolic mirrors that concentrate solar rays in their focus, where the solar energy is collected and used to operate a steam turbine generator. The mirrors can be large structures (the one we are studying is a parabola with a 5 meter aperture) and are susceptible to damage due to strong winds. As such, there is interest in studying the wind load over the mirror, both numerically and experimentally.

We are using one of such experiments as a reference, where a 1:25 scale model of a single mirror was placed in a wind tunnel. This experiment was performed for Abengoa Research, which has given us access to the data, and was reported in [6, 75]. Note that, as requested by the company, we are providing all our results in scaled form to protect intellectual property.

Due to the larger scale of the model compared to the previous examples, we will not attempt to reproduce the full boundary layer and the Werner-Wengle wall model [131] will be used to introduce the equivalent wall friction close to solid surfaces. Additionally, as we are trying to reproduce a wind tunnel experiment with a turbulent incoming

flow, we need to generate a time-dependent inlet condition which bears a statistical resemblance to a real wind signal (see for example [73] or [107]).

The geometry of the collector is shown in Fig. 3.20 at full scale. To reproduce the different configurations of the collector as it rotates over its axis to follow the sun, the seven cases shown in Fig. 3.21 modifying the pitch of the mirror in increments of 30°. The average wind is always assumed to reach the collector frontally, so that the mirror presents the maximum possible area to the flow.

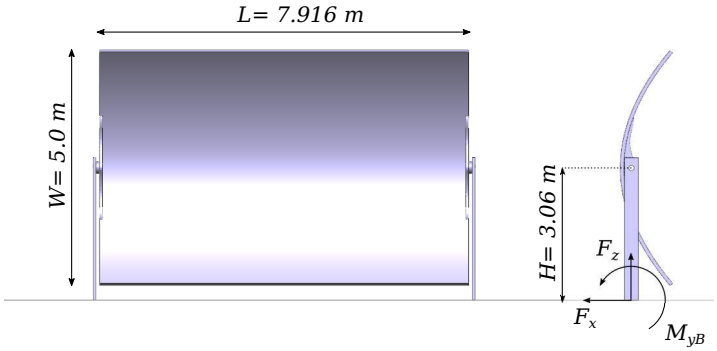


Figure 3.20: Parabolic collector dimensions (full scale).

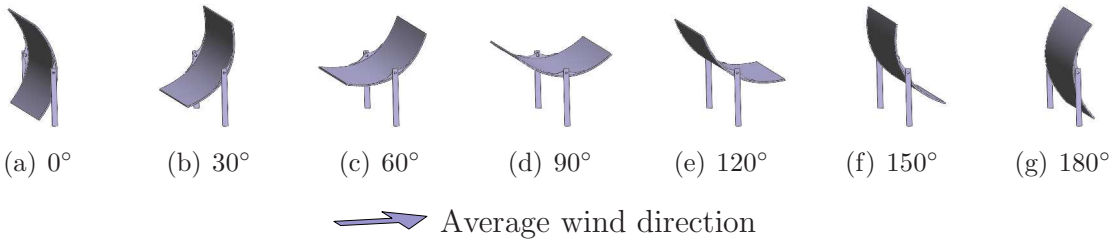


Figure 3.21: Solar collector – pitch angles considered in the simulation. Wind blows from the left.

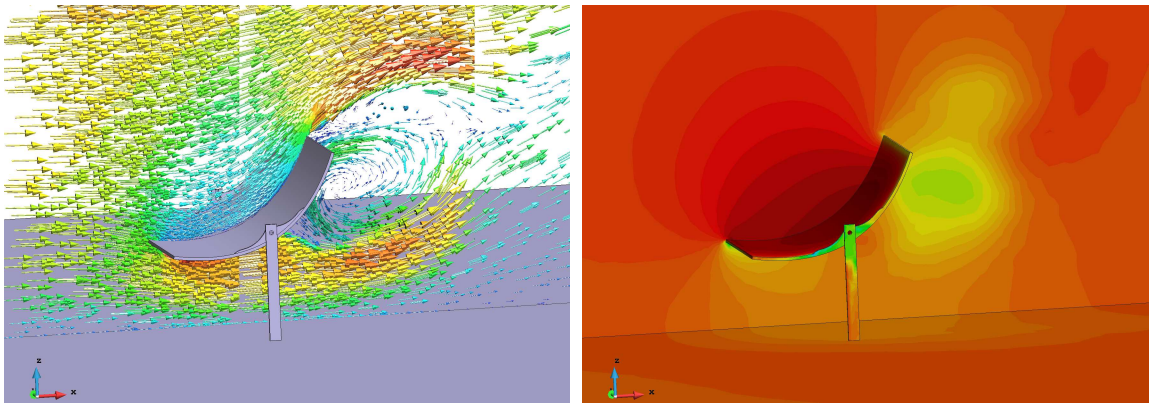
We simulated the collector at model scale (1:25), reproducing the wind tunnel experiment. The collector was placed on a fluid domain of $3 \times 2 \times 1.4\,m$ at model scale in the stream-wise, cross-stream and vertical directions, respectively, at $5W = 1\,m$ from the inlet.

The incoming wind is generated using the model of [73] to follow a Kaimal wind spectrum [62] with a reference stream-wise velocity $U_{ref} = 6.26\,m/s$ at $0.4\,m$ (corresponding to $10\,m$ at full scale) and a wall roughness $z_0 = 0.0012\,m$ ($z_0 = 0.03\,m$ at full scale). The air is assumed to have density $\rho = 1.225\,Kg/m^3$ and viscosity $\nu = 1.4604 \times 10^{-5}\,m^2/s$, resulting on a Reynolds number Re ≈ 85000 calculated using U_{ref} and W.

The calculation was performed with an unstructured tetrahedral mesh, with mesh sizes ranging from $h = W/50 = 0.004\,m$ close to the mirror surface to $h = 0.32\,W =$

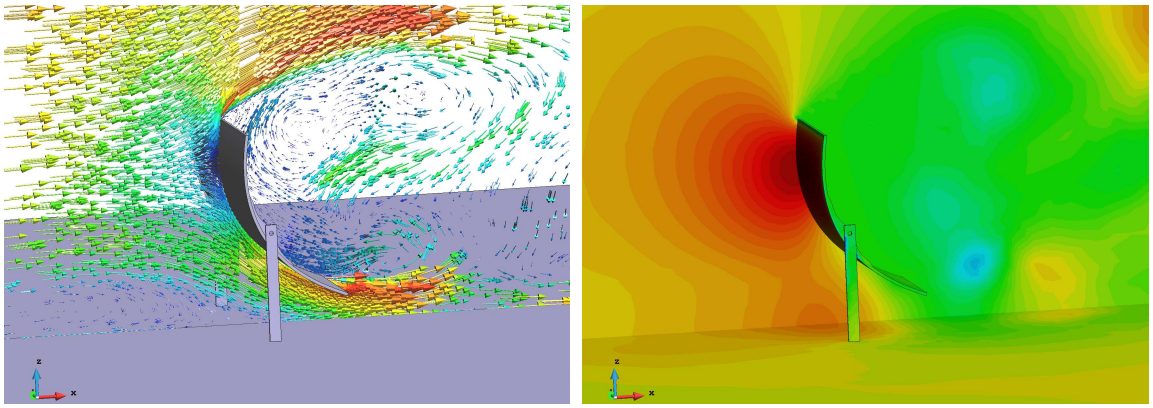
$0.064\,m$ on the far regions. This resulted in approximately 1.5×10^5 nodes and 8×10^5 elements for each simulation.

Instantaneous distributions of velocities and pressures close to the collector are shown in Fig. 3.22 for the case with pitch angle 60° and in Fig. 3.23 for the case of 150° as an example of the results. As can be seen in the figures, a vortex trail develops due to the presence of the collector.



(a) Velocity vectors. (b) Pressure contours.

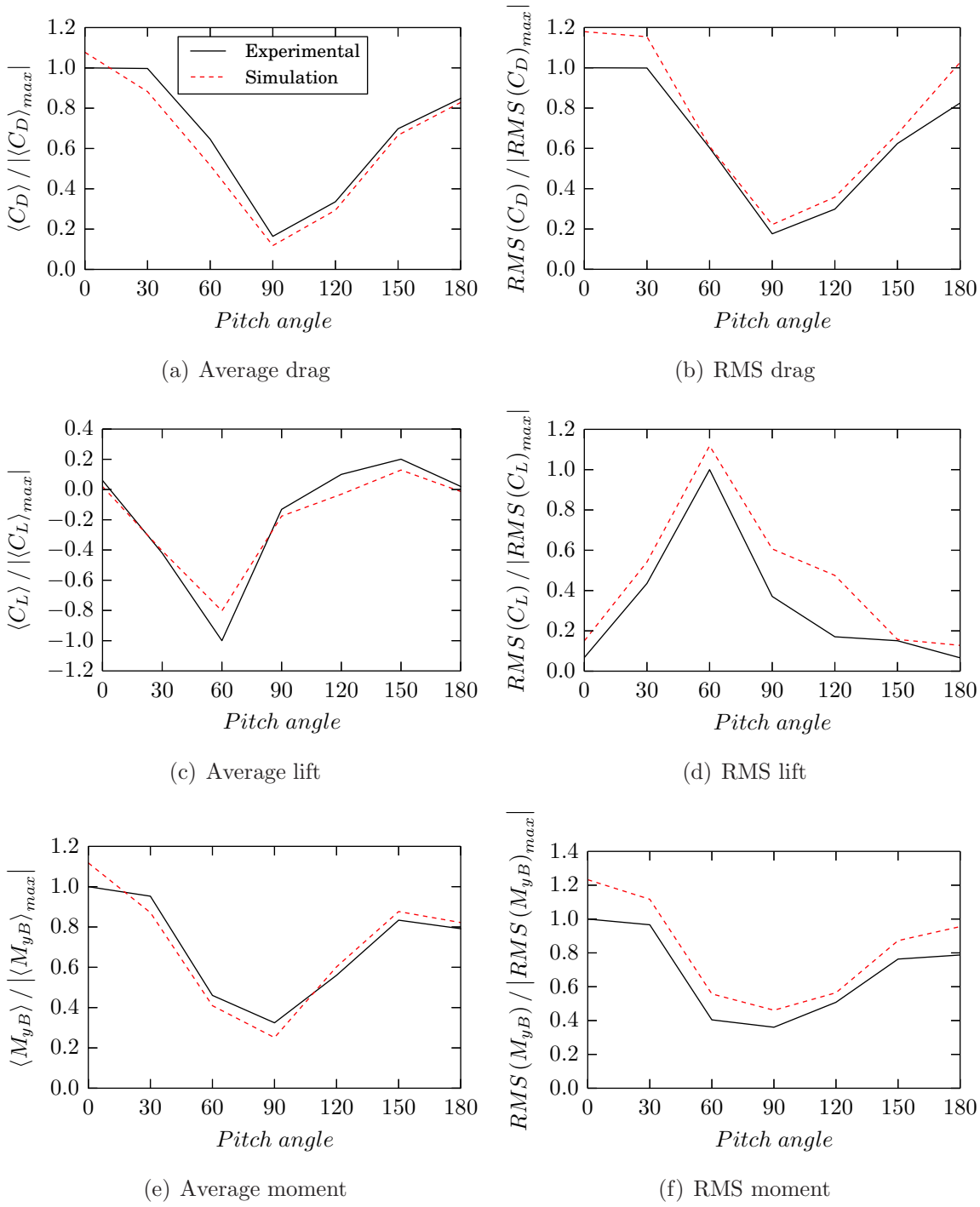
Figure 3.22: Solar collector – instantaneous velocity and pressure fields for pitch angle 60°.



(a) Velocity vectors. (b) Pressure contours.

Figure 3.23: Solar collector – instantaneous velocity and pressure fields for pitch angle 150°.

The simulation is performed for a sufficiently long time for the flow to become statistically steady and obtain a data set on this regime so that meaningful statistics can be obtained. The forces on the mirror are integrated at each time step and recorded



(a) Average drag

(b) RMS drag

(c) Average lift

(d) RMS lift

(e) Average moment

(f) RMS moment

Figure 3.24: Solar collector – average reactions.

over time. These results are used to compute the drag and lift coefficients and the moment M_{yB}, calculated around an axis placed on the base of the structure, as shown in Fig. 3.20. The reactions are made dimensionless using U_{ref} and the collector dimensions W, L as follows

$$C_D = \frac{\langle R_x \rangle}{0.5\rho U_{ref}^2 W L} \qquad C_L = \frac{\langle R_z \rangle}{0.5\rho U_{ref}^2 W L} \qquad C_M = \frac{\langle M_{yB} \rangle}{0.5\rho U_{ref}^2 W^2 L} \tag{3.86}$$

the resulting statistics for the different pitch angles considered are compared to the experiment in Fig. 3.24. The results are presented relative to the maximum value in each experimental curve as requested by Abengoa.

As was the general trend in the previous example, we achieve good accuracy in terms of the averages, while the results in terms of fluctuations (the RMS in this case) tend to be less accurate. In any case, we consider that this case serves as a proof of concept, showing that the formulation has potential to solve problems of practical interest in engineering, with complex geometries and variable inlet conditions.

3.9 Summary and conclusions

In the present chapter we have introduced a new FIC-based formulation for incompressible flows. The main features of our method are the presence of a new term on the momentum equation, which introduces an additional non-isotropic dissipation in the direction of the velocity gradients, and a stabilized formulation for the mass equation which is based on a second order FIC balance in space and represents an incompressible Eulerian version of the method presented for quasi-incompressible flows in [89].

This method has a free parameter in the combination coefficient β that defines the relative weights of the *classical* streamline diffusion and the new gradient diffusion term in the stabilization of the momentum equation. We have proposed a way to define this coefficient dynamically, with the intent of improving the results and reducing the dependence of the solution on the free parameter.

We have tested the method with several application examples. The first example presented is the turbulent channel flow at $\mathrm{Re}_\tau = 395$, where we have tested the different variants of the method. The main conclusions we extracted from the simulations can be summarized as follows:

- Hexahedral meshes provide significantly better results than tetrahedra, with 32^3 hexahedra producing results of comparable quality to 6×64^3 tetrahedra.
- The method is not very sensible to the free parameter β, at least for values of $\beta \geq 0.5$, for either the fixed or dynamic β variants.
- Using the dynamic formula for the β coefficient results in a better approximation to the DNS curves, although the problem to be solved becomes more non-linear.

Finally, the method was then applied to large turbulent flow simulations, first the flow around a cylinder and then an industrial-scale simulation of the wind flow over a parabolic solar collector. In this last case, due to the larger Reynolds number, the method was used in combination with the Werner-Wengle wall model to introduce the right dissipation on the walls without reproducing the full boundary layer.

The results show that the presented method is capable of reproducing the features of the flow and shows promise in the application of the method real world examples of industrial interest.

Chapter 4

Parallel implementation

4.1 Introduction

We have already remarked that one of the reasons why turbulent flow problems are challenging is that they involve fluid motions with very different characteristic sizes and, as a result, their numerical simulation requires very fine discretizations, both in space and in time. Even with the introduction of turbulence modeling, LES simulations for problems of practical interest in engineering require significant computational resources and are well beyond the range of what currently can be calculated in a reasonable time with a desktop computer.

Currently, the overwhelming majority of High Performance Computing (HPC) machines are distributed memory clusters (see for example the TOP500 list of the most powerful computer systems [118]), in which many individual processors work in parallel on different parts of the problem to be solved. Such machines are organized as groups of interconnected calculation nodes, where each node contains one or several processors and a block of memory. In this calculation framework, there exists a very clear distinction between local data, stored in the node's memory, which is readily available to the processor, and non-local data, stored in a different node, which has to be requested to that node requires many more computation cycles to access.

In the context of finite element simulations, large problems can be solved using a distributed memory approach by dividing the model domain into parallel subdomains, each containing a fraction of the finite element mesh, and assigning each of them to an individual processor. This requires careful design of the calculation software to solve the global problem while minimizing the exchange of information between the individual subdomains.

In this sense, a part of the programming work presented here represents a contribution to an ongoing work in our research group to improve the performance of Kratos Multiphysics in distributed HPC clusters. This chapter will be devoted to presenting

some of the adaptations required for distributed memory simulations and to test the parallel performance of the resulting implementation.

We will focus our presentation on some aspects of parallelization that have required the most attention to achieve the goals of the present work. The first of these is the division of the original finite element mesh into parallel subdomains and the generation of a communication strategy to efficiently exchange information between them. To present this, we briefly describe in Section 4.2 how distributed data is organized in the code and how it can be shared among processes, while Section 4.3 deals with the generation of this distributed data from the complete problem.

The second aspect that will be considered is the parallel efficiency of the complete solver. We describe some details of the assembly and solution of the problem's linear system that require attention in a distributed memory context in Section 4.4 and evaluate the parallel performance of the complete solver in Section 4.5.

Finally, as we have seen in Chapters 2 and 3, in turbulent flow analysis the quantities of interest are frequently statistical results, obtained from spatial and/or temporal averaging. In the context of large data sets and HPC, the efficient calculation of statistical results requires careful consideration. This will be discussed in Section 4.6, where the approach we have followed will be described.

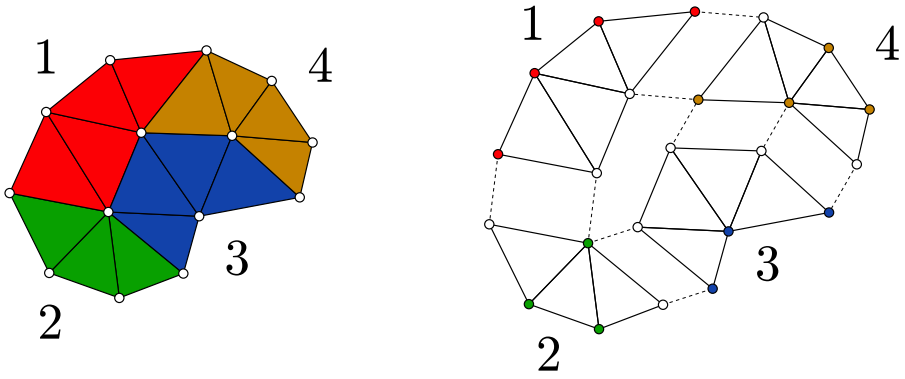
Some final thoughts and future lines of improvement are presented in Section 4.7.

4.2 Distributed memory model

While there are multiple strategies to distribute data for finite element problems, the approach of Kratos Multiphysics consists in dividing the mesh into discrete subdomains, such that each individual element is assigned to a given processor, as shown for an example problem in Fig. 4.1(a). In this approach, some of the mesh nodes will appear in two or more different subdomains, as they belong to elements assigned to different partitions. Such nodes will be called *interface nodes*.

It is critical to the solution of the problem that nodal data for interface nodes is consistent across the different partitions, which requires its synchronization given points of the solution procedure. To simplify this exchange of information, we found it convenient to also assign nodes to a given partition, which holds the reference values for nodal data. From the point of view of a calculation process, nodes which belong to its partition are defined as *local nodes*, while interface nodes which are known but are assigned to a different partition are called *ghost nodes*. An example of such partition is shown in Fig. 4.1(b), where local nodes are represented with a full circle in the color of the partition and ghost nodes are represented with empty circles. In the same figure, dashed lines represent inter-process communications which will be required during the solution procedure.

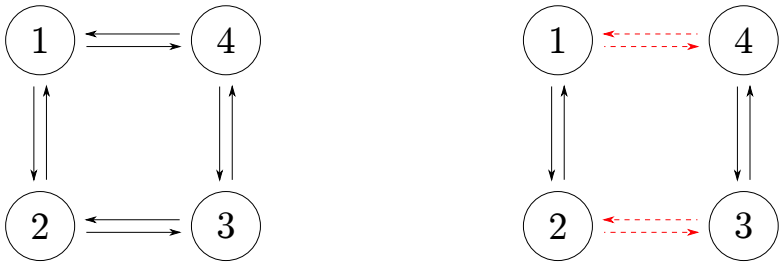
In a communication strategy, it is important to define not only which partitions have



(a) Element partition. (b) Node partition.

Figure 4.1: Division into subdomains.

to share information with each other but also the order in which such communication is done. For example, the communication graph for the partition of Fig. 4.1 is shown in Fig. 4.2(a).



(a) All links. (b) Optimal.

Figure 4.2: Communication patterns for the partition of Fig. 4.1.

The simplest approach would be transferring data in order, starting from one of the pairs and exchanging data one by one. This approach would have poor parallel performance as, while a given pair of processes exchanges information, the others are idle. Furthermore, the total number of exchanges required grows rapidly as the number of processes is increased, resulting in a significant fraction of the total calculation time spent waiting for communication.

A more efficient approach is to try to group communications in a way that processes are never idle, such as in Fig. 4.2(b), considering one first stage where process 1 communicates with 2 and 3 with 4, followed by a second phase where process 1 communicates with 4 and 2 with 3. Reducing the number of total communication stages and, in this case, keeping all processes busy at all times.

In the case of Kratos Multiphysics, an existing code which had to be adapted for

parallel simulations, the existing data structure had to be adapted to work in a distributed memory context. While we won't go into the details of the Kratos structure here (the interested reader is directed to [33, 35]), we will just say that all model data is stored in an entity known as `ModelPart`, which contains the definition of the mesh and the associated nodal or elemental data. In a parallel context, each process defines its own `ModelPart`, which in this case contains all elements in the subdomain assigned to that process and all nodes (both local and ghost) required to perform the simulation.

Data transfer across processes was achieved by adding an additional component in the `ModelPart`, the `Communicator`, which handles parallel communication [34]. This object stores the communication strategy (which process will I communicate with on each stage) and the lists of local and ghost nodes to be exchanged in each stage. When data transfer is required, the `Communicator` collects the nodal data to send on each stage and exchanges it with the corresponding process using Message Passing Interface (MPI) calls [74].

One of the advantages of this approach is that it allows to reuse a very significant part of the code between serial and parallel simulations. To do so, the code is programmed with the parallel implementation in mind, with all required data transfer performed through `Communicator` functions. Then, two different `Communicator` classes are implemented, one for parallel executions that works as defined above, and another for serial runs, which does nothing. With this approach, we have been able to significantly reduce code duplication, minimizing the maintenance problems related to keeping up to date a serial and a parallel version of the same code.

4.3 Partitioning of input data

Once we have a data structure that can be used to hold and communicate distributed finite element data we need to be able to, given a simulation domain, generate a partition in as many subdomains as parallel processes will be used in the simulation and a communication strategy to transfer data between them. In Kratos Multiphysics, the usual situation is to have a single input file containing the entire simulation mesh, typically generated with GiD [22]. The partitioning procedure in this case constitutes one last step of the pre-process of the problem, generating separate inputs for each simulation domain.

The partitioning process should generate balanced partitions, that is, all partitions should contain a similar number of elements and nodes, to ensure that the computational load is homogeneous for the different processors. A poorly balanced partition means that the processes with lighter work loads have to wait for the other to finish, resulting in longer total calculation times. In addition, the partition should minimize the amount of communication required relative to the total calculation time. This is sometimes expressed by saying that the computational cost of the simulation is propor-

tional to the number of elements, that is to say, to the *volume* of the partition, while the communication costs are proportional to its *surface*. Therefore, a good partitioning algorithm should minimize the surface to volume ratio for the partition.

We use a the capabilities provided by the METIS library [63], which generates a partition using a multi-level approach. Starting from a graph of nodal connectivities, multi-level methods are based in generating progressively coarser graphs by grouping close nodes together and generating an optimal partition of the coarsest graph. The partition is then refined by undoing the coarsening. Such approaches are considered to provide good quality partitions for unstructured meshes.

As mentioned, the approach of METIS is based on the nodal graph, and provides a partition of the nodes. While there are some capabilities in METIS to divide the elements, they work for homogeneous meshes, composed of a single type of element[1]. This was insufficient for our needs as, due to the way boundary integrals are implemented in Kratos Multiphysics, the meshes in the problems we are simulating are typically heterogeneous. A $3D$ simulation, for example, may include tetrahedral volume elements, triangular faces used to apply Neumann boundary terms or wall laws, for example, and point-to-point links to apply periodic boundary conditions. All of these have to be taken into account when building the nodal graph and the latter in particular has a significant impact in the final partition, as it typically indicates a relation between two nodes that are geometrically far away.

Element partitioning is achieved by following some simple rules. First, if all nodes in the element belong to the same partition, the element is automatically assigned to that partition. If its nodes are assigned to different partitions, then the element is preferentially assigned to the partition that owns most of the nodes, but keeping into account load balance: if that partition already contains more elements than the others, the element will be assigned to one of the partitions that own the remaining nodes.

Once this initial distribution in done, partitions are checked again for isolated nodes. Due to the way the element distribution procedure works, it can happen that some domain has a node but no elements that contain it. This situation would generate unnecessary communication, as the partition would have to store and update the reference data for a node it never uses, so those nodes are reassigned to a partition that needs them.

With the partition complete, the next step is to generate a communication strategy. To achieve this, we start by building the domain graph, stored as a matrix where term a_{ij} is non-zero if there is an interface between partitions i and j. For example, the graph for the domain of Fig. 4.1 is

[1]More recent versions of METIS do provide the possibility of partitioning heterogeneous meshes, but at the time of implementation we were limited by the relatively old versions available in the clusters we had access to.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{4.1}$$

The strategy to organize the communications is known as *coloring*, as the problem is equivalent to assigning a color to all edges in the graph such that no two edges of the same color reach the same node. Once this is done, communication is achieved by transferring data across all interfaces of the same color at the same time. To perform the coloring, we use a simple approach, always assigning the first available color as they are needed, presented as Algorithm 4.1. This has been shown to be sub-optimal, as in the worst case it can use up to twice as many colors as the number of partitions. However, in our experience, obtained solutions are not usually as bad. A more concerning issue is that it is significantly more computationally expensive than an approach such as [76], which would produce optimal strategies.

Algorithm 4.1 A simple coloring procedure.

```
1: Set N_p to the number of partitions.
2: Initialize matrix of colors C with zeros.        ▷ Maximum size of C is N_p × 2 N_p
3: Set N_c = 0.
4: for i in [1, N_p] do
5:     for j in [i + 1, N_p] do
6:         if a_ij is not 0 then                    ▷ a_ij as given by Eq. (4.1)
7:             for k in [1, 2 N_p] do
8:                 if c_ik is 0 then                ▷ Use first available color
9:                     c_jk = i
10:                    c_ik = j
11:                    if k > N_c then
12:                        N_c = k
13:                    end if
14:                    break    ▷ Once a color is assigned, skip to next communication
15:                end if
16:            end for
17:        end if
18:    end for
19: end for
20: return N_c                                      ▷ Number of colors used
```

For the partition of Fig. 4.1, this procedure results in the following color matrix, where each column corresponds to a color and unused columns have been omitted:

$$\boldsymbol{C} = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 4 & 2 \\ 3 & 1 \end{bmatrix} \tag{4.2}$$

If we take each row of $\boldsymbol{C}$ as a parallel process and each column as a stage in the communication procedure, c_{ij} indicates which partition we have to communicate with at a given step. In this case, we recover the communication pattern of Fig. 4.2(b). Note that, for more complex cases, we may obtain a communication pattern where some processors are idle for some of the communication steps. If this happens, the corresponding positions in $\boldsymbol{C}$ contain a -1.

The final step in the procedure is to write each partition and the associated communication data (local and ghost nodes and the communication strategy) to separate input files, one for each simulation process. Once this is done, the parallel solution procedure can start.

Taking into account all steps, the partition procedure can summarized as:

1. Read nodal connectivities and call METIS to partition the node graph.
2. Distribute elements.
3. Check each partition for isolated nodes, and move these to other partitions.
4. Use a coloring procedure to generate a communication strategy.
5. Write partition data to separate input files, one for each parallel process.

4.4 Distributed solution

Once the problem domain is partitioned and a communication strategy has been determined the model is prepared for a parallel simulation. Compared to a serial solution, a distributed memory simulation introduces additional complications in the procedure. First, as the finite element mesh was divided into subdomains, the system matrix will also be distributed, with the contribution from each individual element being computed in the process that holds it. More importantly, once the system matrix has been assembled, it has to be solved using a distributed algorithm which can take full advantage of the parallel environment. In fact, the choice of a scalable parallel solver has a crucial impact on the parallel performance of the code. Finally, once the problem has been solved, the updated values for the variables are synchronized to ensure that nodal variables are consistent across the processors.

In terms of the parallel finite element assembly procedure, Kratos Multiphysics relies on the Trilinos library [52] to construct and manage the system of linear equations that

has to be solved on each iteration. In particular, Trilinos' Epetra package provides an implementation of distributed memory sparse matrices and vectors that can be used to hold the system data and communicate it to other processes when needed.

Finite element matrices can typically be stored in a sparse format, since most of the terms in the system matrix are zero. Therefore, just as in the serial case, before constructing the system matrix and vector we need to determine the sparsity pattern and allocate the required memory. This is done based on the nodal graph: given that each matrix row (and column) corresponds to a given degree of freedom in the problem, an entry will be non-zero only if an element exists in the mesh connecting the nodes that the row and column degrees of freedom are associated to. Since the matrix implementation in Kratos Multiphysics is row-based, each row is considered local to the processor that owns the node associated to that row's degree of freedom. With this information we can generate the data structure that will hold the system data, which is ultimately an instance of Trilinos' `Epetra_FECrsMatrix` for the system matrix and an `Epetra_FEVector` for the right hand side vector.

The construction of the sparse data structure is a relatively time-consuming operation, but can be done once the first time the matrices have to be constructed and reused in subsequent iterations, as long as the mesh connectivity pattern is preserved. Note that this is true for the examples in the present chapter, as well as those in Chapters 2 and 3, but will not be the case for the adaptive mesh refinement simulations of Chapter 5.

Once the data structure is ready, we iterate over all elements in the model, calculating the elemental contribution to the problem and assembling it into the system matrix and vector. However, since each row receives contributions from all elements that contain the associated node, parallel communication is required to account for contributions coming from elements associated to a different processor. To minimize the amount of parallel communication required, the global matrix is assembled locally at first to add the contributions from all local elements. Once this is done, rows that receive contributions from multiple subdomains are assembled by adding the contributions from each and communicating the result to other partitions.

Once the system is solved, we encounter the inverse problem: the new nodal values have to be communicated from the process that stores that particular row to all processes that have a copy of the corresponding node. Again, this requires a communication step.

The (efficient and scalable) parallel solution of the system is, by itself, a different and very challenging problem, which will only be briefly presented here. Given the size of the systems to be solved, direct methods are not applicable due to their memory requirements and computational cost. A more adequate choice is to use parallel implementations of Krylov methods [108], which are very efficient solvers but are not designed as parallel algorithms and tend to require an increasing number of iterations to converge as more partitions are introduced. Alternatively, a variety of algorithms designed for parallelism exist, including Algebraic Multigrid (AMG) algorithms [119, 128], which in-

troduce a hierarchy of increasingly coarser versions of the original problem and use the coarser solutions to accelerate the convergence of the finer problem, solution strategies based on domain decomposition techniques, in which each domain is solved separately and its effect on its neighbors is felt through boundary terms and deflation techniques, which combine an aggressive coarsening with domain decomposition.

In Kratos Multiphysics, the usual approach has been to rely on the linear solvers provided by the Trilinos library, which includes both Krylov solvers on its Aztec package [129] and Multilevel algorithms through the ML [43] library. This approach was used for example to simulate the benchmarks presented in [34]. In the following pages we will use a different linear solver in combination with the incompressible flow formulation presented in Chapter 2. This solver is provided by the AMGCL library [37, 38], currently under development, and is based on combining a deflated approach where the coarse problem is given by the MPI partition with an AMG solver within each subdomain.

4.5 Benchmark cases

To evaluate the performance of the parallel implementation of our solver we have performed some simulations of large test cases. Note that, unlike in the remainder of this work, here we are more interested in the performance of the algorithm than in the quality of the solution. Therefore, we simulate very short time spans, mainly to sample the computational costs associated to the procedure, and the resulting solutions can not be considered as an accurate representation of the real flow. These examples were simulated in the Gottfried cluster of the North-German Supercomputing Alliance (HLRN).

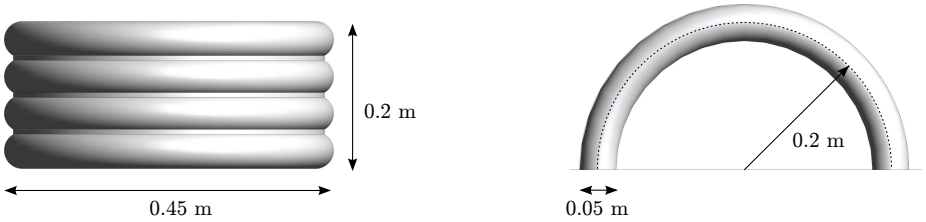
4.5.1 Flow around an inflatable structure

The first test case is adapted from a real simulation mesh courtesy of the uLites[2] project team. This project was concerned with the design and construction of inflatable structures subjected to wind loads and included the comparison of numerical and wind tunnel experiments on the designed structure. Here we take one of the simulation meshes, representing the geometry of a rigid scale model that was used in wind tunnel tests and simulate the flow around it using the Q-ASGS formulation described in Chapter 2.

The model, presented in Figure 4.3, is composed of four cylindrical tubes with a diameter of 0.05 m folded into a semicircle with a radius of 0.2 m, measured from the tube centerline. The simulation domain, shown in Figure 4.4, measures $2.6 \times 1.4 \times 1$ m in the streamwise, cross-stream and vertical directions, respectively, and the inlet is placed at 0.8 m of the axis of the module.

[2]Ultra-lightweight structures with integrated photovoltaic solar cells: design, analysis, testing and application to an emergency shelter prototype. EC project within the Seventh Framework Programme, Grant No. 314891.

The incoming flow is modeled as air with density $\rho = 1.25\,\mathrm{Kg/m^3}$ and viscosity $\mu = 1.875 \times 10^{-5}\,\mathrm{Pa \cdot s}$ flowing at $5.5\,\mathrm{m/s}$, which corresponds to a Reynolds number of approximately 7.3×10^4 relative to the radius of the module. Velocity is fixed to zero on the floor and the surface of the module, while a no-penetration condition is used on the sides of the calculation domain. Note that, given the mesh resolution, a wall law would be a more adequate boundary condition for the solid boundaries, but we will not use it here in order to concentrate our attention on the fluid solver.



(a) Top view. (b) Side view.

Figure 4.3: Inflatable structure model – geometry of the module.

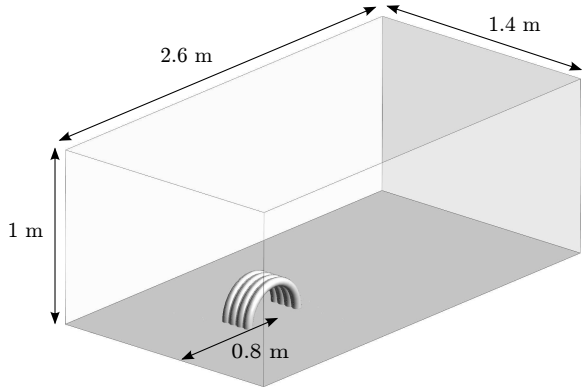


Figure 4.4: Inflatable structure model – dimensions of the simulation domain.

We define two different simulation meshes. The first is an unstructured tetrahedral composed of around 760000 nodes and 4 million elements, with sizes ranging from $0.0021\,\mathrm{m}$ close to the surface of the module to $0.067\,\mathrm{m}$ in the far regions. The second is obtained by the bisection of the element edges of the first, obtaining 8 elements from each original tetrahedron, and contains a total of 5.8 million nodes and 35 million tetrahedra. The flow is simulated for five time steps, using a step size $\delta t = 0.001\,\mathrm{s}$ for the 4 element mesh and $\delta t = 0.0005\,\mathrm{s}$ for the 35 million element case, halved to maintain the same Courant-Friedrichs-Lewy (CFL) number.

The time required to perform different parts of the procedure for the coarser mesh is summarized in Table 4.1. The solution of five complete time steps required a total of

between 32 and 36 system solutions, which took between 530.2 and 82.88 seconds of wall clock time when going from 96 to 768 parallel processes. We notice that the simulation time is dominated by the solve phase, which is more expensive than the finite element assembly in all cases.

MPI processes		96	192	384	768
Simulation time	Assembly time (s)	40.6	21.75	12.13	17.67
	Solution time (s)	530.2	191.47	96.83	82.88
	Num. of iterations	36	36	33	32
First iteration	Assembly time (s)	1.88	2.23	3.05	12.94
	Assembly speedup	1	0.84	0.62	0.15
	Solution time (s)	15.02	5.87	3.29	2.89
	Solution speedup	1	2.56	4.57	5.2
Average iteration	Assembly time (s)	1.11	0.56	0.28	0.15
	Assembly speedup	1	1.98	3.9	7.25
	Solution time (s)	14.72	5.3	2.92	2.58
	Solution speedup	1	2.78	5.04	5.7
Last time step	Solution time (s)	41.3	13.06	7.39	6.57
	Num of iterations	3	3	3	3
	Average solution (s)	13.77	4.35	2.46	2.19
	Solution speedup	1	3.16	5.59	6.29
Global speedup		1	3.03	5.41	6.35

Table 4.1: Inflatable structure test – measured times and parallel speedup for the 4 million element mesh.

We also observe that the first iteration of the finite element assembly procedure takes more time than subsequent iterations, since it includes the calculation of the sparsity pattern for the system matrix and the allocation of the required memory. We notice that, far from showing parallel scalability, the time spent in the first iteration is in fact increased as more processors are used. We consider this to be consistent with the fact that the set-up phase requires abundant parallel communication to determine the shape and ownership of each matrix row, which takes more time as more processes are involved. In any case, we have not considered the time of the first iteration in calculating averages, since it involves more operations than the rest.

Finally, we remark that we have considered the last time step separately from the rest. The cause for this is that, for some of the cases, the first few system solutions stop due to the solver reaching the maximum allowed iterations instead of fulfilling the convergence requirements. This is due to the initial condition being a bad first guess of an equilibrium solution, which means that in the first few iterations the system is harder to solve and the algorithm fails to achieve the desired error tolerance.

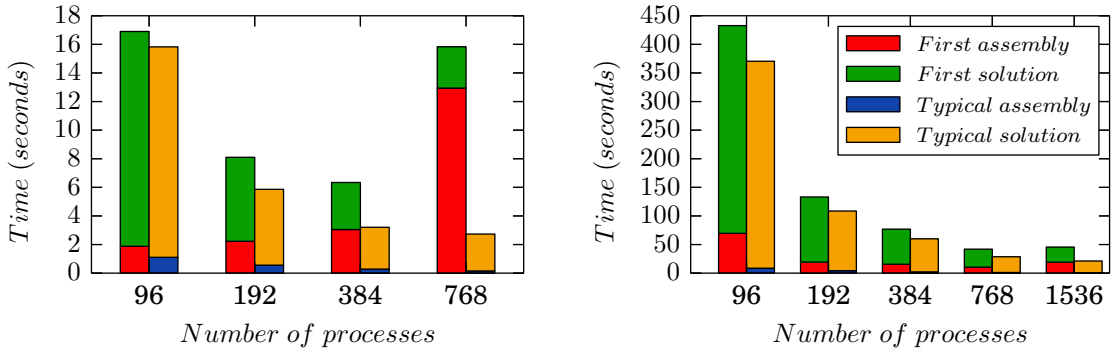
We feel that comparing converged and non-converged iterations skews the scalability results, since we want to measure the time required to achieve a solution of a given quality, not the time it takes to perform an arbitrary number of iterations. For this reason we measured the time required to solve the last time step, where all system solutions converged in all cases, and present it separately for the different cases. Likewise, the values given as global speedup in the results have been computed considering one average system assembly plus the average time taken in solving a single linear system on the last time step.

The times measured for the finer test case, using a 35 million element mesh, are presented in Table 4.2. The behavior of the method is qualitatively the same, and the same remarks about the additional cost of the first iteration are applicable.

MPI processes		96	192	384	768	1536
Simulation time	Assembly time (s)	250.04	109.6	59.5	31.7	30.55
	Solution time (s)	7962.48	2304.2	1217.11	555.29	418.51
	Num. of iterations	22	22	21	20	20
First iteration	Assembly time (s)	69.53	19.32	15.53	10.45	19.18
	Assembly speedup	1	3.6	4.48	6.65	3.63
	Solution time (s)	363.24	113.85	61.39	31.47	26.4
	Solution speedup	1	3.19	5.92	11.54	13.76
Average iteration	Assembly time (s)	8.6	4.3	2.2	1.12	0.6
	Assembly speedup	1	2	3.91	7.69	14.36
	Solution time (s)	361.87	104.3	57.79	27.57	20.64
	Solution speedup	1	3.47	6.26	13.13	17.53
Last time step	Solution time (s)	1084.26	264.21	160.62	70.57	52.37
	Num of iterations	3	3	3	3	3
	Avg. solution (s)	361.42	88.07	53.54	23.52	17.46
	Solution speedup	1	4.1	6.75	15.36	20.7
Global speedup		1	4.01	6.64	15.02	20.49

Table 4.2: Inflatable structure test – measured times and parallel speedup for the 35 million element mesh.

The time costs of a single solution iteration are represented graphically in Fig. 4.5, where the first iteration is compared to a typical iteration, calculated using the average time for the build phase and the last time step average for the solve phase. Here again we see a different behavior for the first and a typical iteration, due to the cost of determining the structure of the system matrix and allocating the required memory. Once this is done, in subsequent iterations the computational cost is driven by the system solution, which takes considerably more time in all cases.



(a) 4 million element mesh.

(b) 35 million element mesh.

Figure 4.5: Inflatable structure test – time costs of a single iteration.

To measure the parallel performance of the procedure we have measured the strong scalability of the algorithm, that is, we have compared the required taken to solve the same problem with an increasing number of processors. This is presented graphically in Fig. 4.6 for the 4 million element case. We can observe that the scalability behavior is basically driven by the system solution, which dominates the solution cost. The parallel efficiency of the finite element assembly is close to linear, taking 7.25 times less time when using eight times more processors, although parallel performance degrades when the number of elements per processor is low. This is to be expected, since the communication cost increases when using more processors while the individual work loads decrease as the number of elements per processor is reduced, which means that the total cost for large enough numbers of processors is dictated by communication operations.

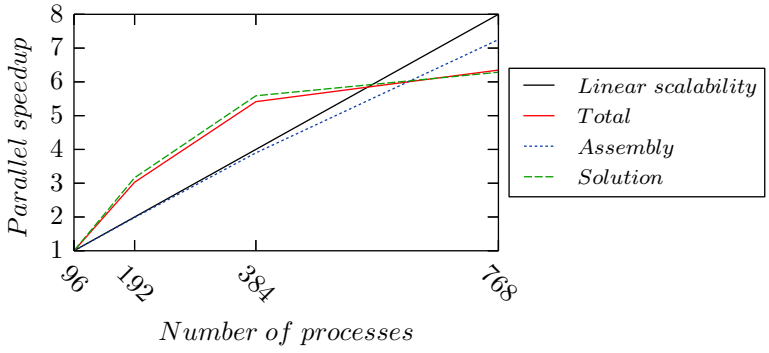


Figure 4.6: Inflatable structure test – parallel speedup for the 4 million elements.

The behavior of the solution phase is possibly more unexpected. The costs reported in Table 4.1 correspond to a better than linear scaling, which is a consequence of the

design of the AMGCL solver, which requires less iterations for larger numbers of processors. In any case, we observe that parallel performance is good for smaller numbers of processors but stops being worthwhile for the last test, since doubling the number of processors from 384 to 768 only produces a modest reduction in calculation costs. Again, this is consistent with communication dominating the total cost of the procedure.

The same strong scalability study can be done for the 35 million elements grid, producing the results presented in Fig. 4.7, and yields results that are qualitatively similar to those obtained for the 4 million element case. Here, however, the larger problem size means that we can use more processes before communication costs dominate, obtaining better scalability results for a given number of processors.

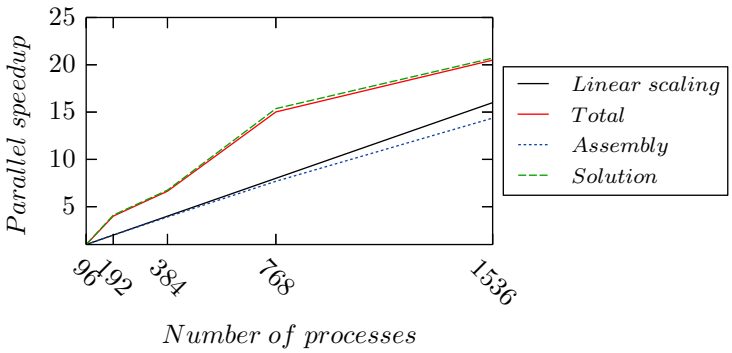


Figure 4.7: Inflatable structure test – parallel speedup for the 35 million elements.

Finally, we can use the results we have to estimate the weak scalability of the algorithm. Weak scalability is calculated by comparing the cost of solving a problem with a given number of processors to that of solving a larger problem with a proportionally larger number of processors. In our case, the cost of assembly is proportional to the number of elements while the cost of the system solution is proportional of the number of degrees of freedom and therefore to the number of nodes. Refining the problem, we obtained a mesh that contains exactly 8 times more elements while multiplying the number of nodes by approximately 7.6. We used this fact to estimate the weak scalability by comparing the time used to solve the 4 million element problem with the time required to perform the same operation using 8 times more processors in the 35 million element mesh. This is presented in Table 4.3, where we can see that, while the build phase does scale weakly, the fine problem takes up to four times as long to solve for the fine mesh.

Time ratio (fine problem/coarse problem)	t_{768}/t_{96}	t_{1536}/t_{192}
Average assembly time ratio	1.01	1.072
Typical solution time ratio	1.709	4.01

Table 4.3: Inflatable structure test – weak scalability estimate.

4.5.2 Flow around a race car

The second example we used to benchmark the solver is the flow around a race car, using the geometry shown in Fig. 4.8, adapted from [31]. The car is 5 m long, 1.8 m wide and 1 m high and it is placed within a fluid domain measuring $13.2 \times 6 \times 3.1$ m in the streamwise, cross-stream and vertical directions respectively. The problem being simulated consists in a flow of air ($\rho = 1.25\,\mathrm{Kg/m^3}$, $\mu = 1.875 \times 10^{-5}\,\mathrm{Pa \cdot s}$) at 1 m/s, which corresponds to a Reynolds number of 1.67×10^5.

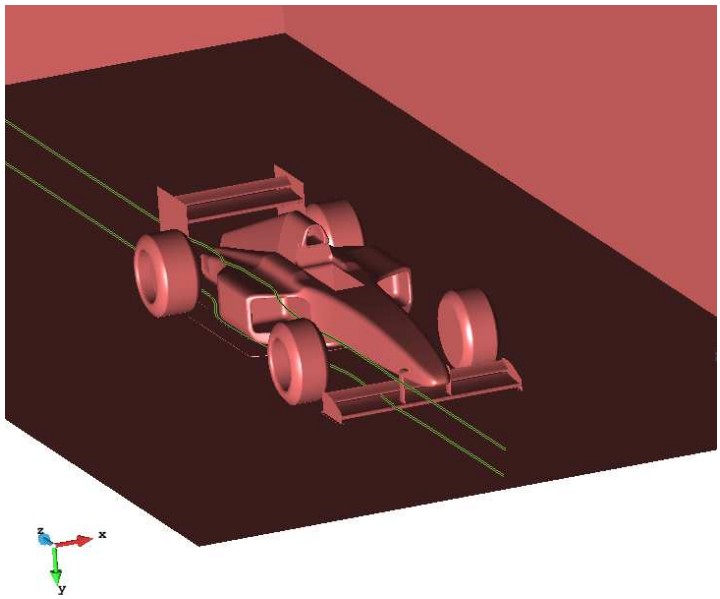


Figure 4.8: Race car test – geometrical model.

As in the previous case, the mesh is not designed for high-quality simulation (this would require a boundary layer mesh or at least a wall law) but to test the parallel capabilities of the solver. We test two different meshes: the first one contains 2.3 million nodes and 12.7 million tetrahedral elements with sizes ranging between 20 mm close to the surface of the car and 0.9m in the far regions, while the second one is obtained by the edge refinement of the first, which is composed of 17.7 million nodes and 102 million elements. Some details of the finest mesh used in the tests can be observed in Fig. 4.9.

Again, five time steps are simulated, using a time step of $\delta t = 10^{-4}$ s. Between 192 and 3072 processes were used for the tests. A summary of the times and iteration counts measured is presented in Table 4.4 for the 12 million element mesh. As in the previous case, we observe a different behavior for the first iteration, due to the fact that it includes the time required to build and allocate the data structure of the system matrix and vector. In this case, the solution required less iterations to converge to the desired tolerance and, unlike in the previous case, no convergence problems were detected in the solution of the linear system. As a result, no distinction will be made between the

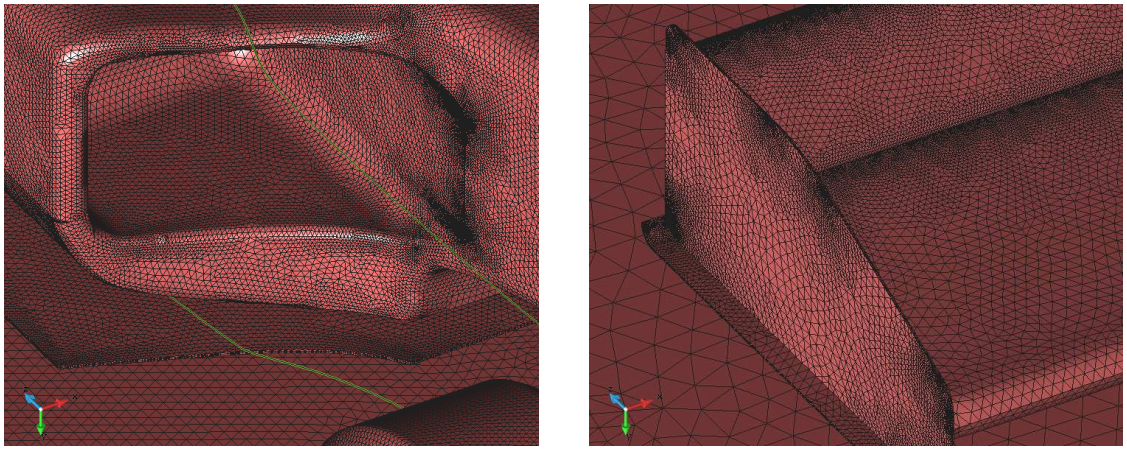


Figure 4.9: Race car test – details of the finer mesh.

results on the last time step and those on the previous ones. However, we do note that the first iteration requires significantly more time to solve the system than subsequent ones, specially for low process counts. As before, we consider this a consequence of the fact that the initial condition used is a bad approximation to an equilibrium solution.

MPI processes		192	384	768	1536	3072
Simulation time	Assembly time (s)	22.06	9.55	18.19	40.53	3.56
	Solution time (s)	258.86	103.65	38.8	33.43	66.05
	Num. of iterations	12	12	12	12	12
First iteration	Assembly time (s)	6.9	1.93	14.15	38.46	1.49
	Assembly speedup	1	3.58	0.49	0.18	4.63
	Solution time (s)	52.07	23.91	4.4	3.57	6.28
	Solution speedup	1	2.18	11.83	14.59	8.29
Average iteration	Assembly time (s)	1.38	0.69	0.37	0.19	0.19
	Assembly speedup	1	1.99	3.75	7.32	7.32
	Solution time (s)	18.8	7.25	3.13	2.71	5.43
	Solution speedup	1	2.59	6.01	6.93	3.46
Global speedup		1	2.54	5.77	6.95	3.59

Table 4.4: Race car test – measured times and parallel speedup for the 12 million element mesh.

The parallel efficiency, in terms of strong scalability, is presented in graphical form in Fig. 4.10. The finite element procedure shows good scalability up to 1536 processors, where increasing 8 times the number of processors reduces the assembly time by a factor of 7.32. This simulation, which corresponds to around 8 thousand elements per

process, gives us an upper limit to the scalability of the current implementation since, from this point on, increasing the number of processes does not result in a reduction of the assembly time. The system solution follows a similar trend, since in this case the solution time is actually increased when the number of processors is doubled from 1536 to 3072.

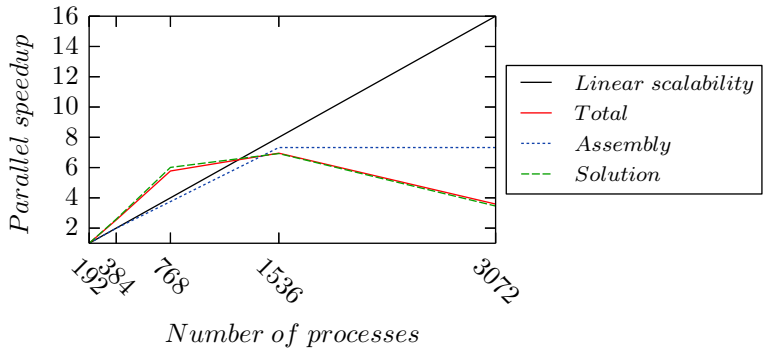


Figure 4.10: Race car test – parallel speedup for the 12 million elements.

The same test was simulated using the 102 million element mesh and the measured times are reported in Table 4.5. Some snapshots of the obtained velocity field are shown in Fig. 4.11, to provide an idea of the mesh resolution and the nature of the solution after five time steps. We observe that, with a larger model, we do not reach the limit of the parallel implementation as in the previous case, although the parallel performance does drop slightly for the cases with more processors.

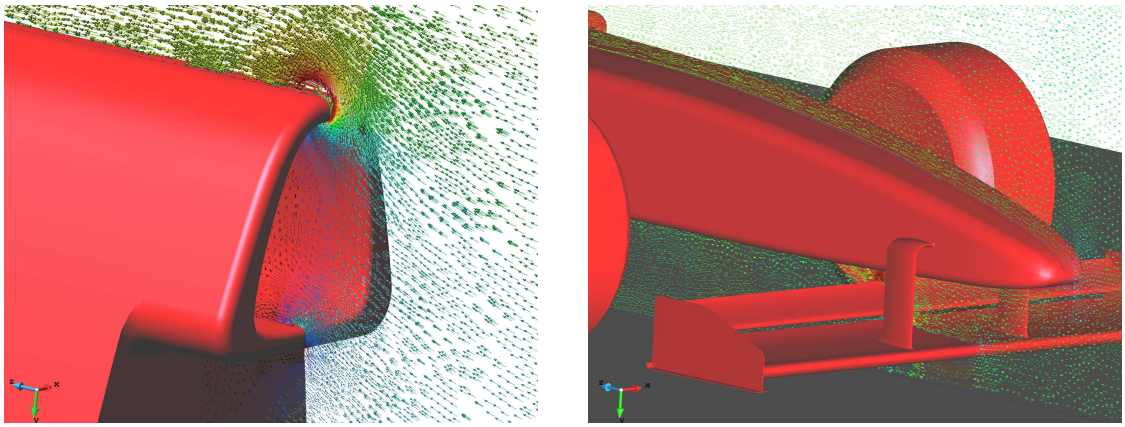


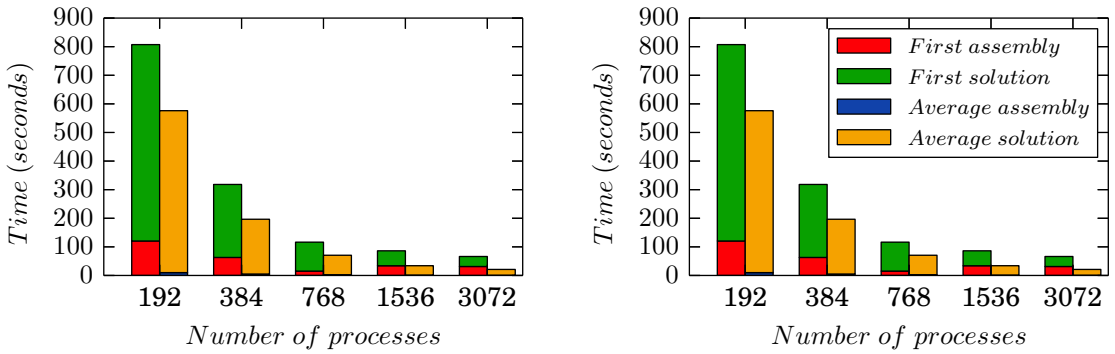
Figure 4.11: Race car test – details of the velocity solution on the finer mesh.

The relative costs of the finite element assembly and solution phases are shown in Fig. 4.12. As in the inflatable structure test, we see that the solution represents takes the lion's share of the computational time and will have the most impact on the overall

MPI processes		192	384	768	1536	3072
Simulation time	Assembly time (s)	250.13	128.56	50.13	51.84	41.92
	Solution time (s)	8047.23	2745.18	972.09	476.86	298.76
	Num. of iterations	14	14	14	14	14
First iteration	Assembly time (s)	120.5	63	15.28	33.97	31.32
	Assembly speedup	1	1.91	7.89	3.55	3.85
	Solution time (s)	686.76	255.24	101.17	52.25	35
	Solution speedup	1	2.69	6.79	13.14	19.62
Average iteration	Assembly time (s)	9.97	5.04	2.68	1.37	0.82
	Assembly speedup	1	1.98	3.72	7.25	12.23
	Solution time (s)	566.19	191.53	67.99	32.66	20.29
	Solution speedup	1	2.96	8.33	17.33	27.91
Global speedup		1	2.93	8.15	16.93	27.3

Table 4.5: Race car test – measured times and parallel speedup for the 102 million element mesh.

performance of the code. Again, we see that the first solution step has a significantly increased cost due to the additional operations and poor scalability, although the drop in parallel performance is not so severe as in the previous example.



(a) 12 million element mesh.

(b) 102 million element mesh.

Figure 4.12: Race car test – time costs of a single iteration.

The strong scalability results of the 102 million element case are presented in graphical form in Fig. 4.13. As was already mentioned, we do not observe the same drop in parallel performance as in the smaller model, due to the fact that here we have 8 times more elements per process. The assembly phase shows good performance up to the largest test, where the reduction in time is less than would correspond to the increase in

processors, but the overall performance continues to be dictated by the system solution costs. The system solution shows good parallel performance in all cases.

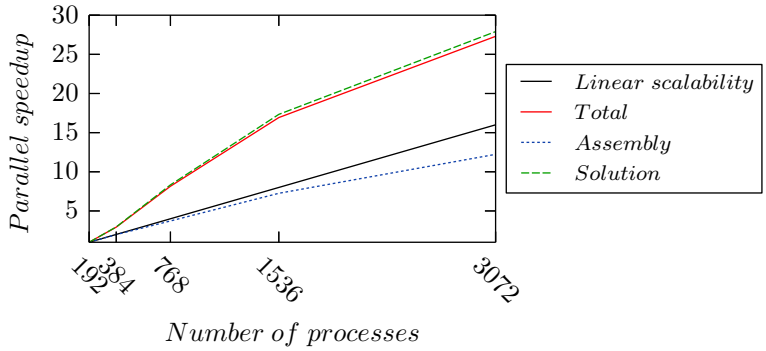


Figure 4.13: Race car test – parallel speedup for the 102 million elements.

We also estimated the weak scalability in this case, keeping in mind that it is an optimistic estimate for the solve phase, since the increase in the number of processors is slightly larger than the increase in the number of nodes (which changes by a factor of 7.7 in this case). The results are presented in Table 4.6 and coincide qualitatively with what was observed for the inflatable structure case. The build phase shows good weak scalability, close to one (although it drops somewhat in the 384 vs. 3072 process comparison), while the solver has a poorer performance.

Time ratio (fine problem/coarse problem)	t_{1536}/t_{192}	t_{3072}/t_{384}
Average assembly time ratio	0.997	1.177
Typical solution time ratio	1.737	2.799

Table 4.6: Race car test – weak scalability estimate.

4.6 Computation of statistical data on a parallel environment

In the present work it has often been necessary to record statistics of turbulent flows. Sometimes this has been challenging, due to the amount of data to be analysed and to the parallel environment in which the simulations have been run, which has made necessary the synchronization of data across processes.

To understand the problems posed by the calculation of statistical results, consider for example the unbiased estimator of the variance σ^2 of a random variable x obtained

using n samples, given by

$$\sigma_n^2 = \frac{1}{n-1}\sum_k^n (x_k - \mu_n)^2 \tag{4.3}$$

where μ_n is the estimate of the mean of x obtained using n samples,

$$\mu_n = \frac{1}{n}\sum_k^n x_k \tag{4.4}$$

The naive approach to the computation of the variance of x involves two loops over the data series: a first one using Eq. (4.4) to obtain μ_n and a second one to evaluate Eq. (4.3). To do so, one should store the entire dataset and analyse it as a post-process. Take as an example the turbulent channel flow simulations using linear hexahedral elements. To record statistical data on this problem, the different values of interest were sampled on the integration points of the mesh. Using second order Gaussian integration we have 8 Gauss points per element. With up to 64^3 hexahedra, we obtain $2^{21} \approx 2.1\times10^6$ samples per variable and time step. Keeping in mind that we want to calculate statistics involving all three velocity components, pressures and their gradients, storing the entire data set soon becomes very expensive in terms of memory.

Even if that data was effectively collected, consider that we used it to compute plane averages. In a parallel environment, the integration points on a given plane will likely lie on multiple processes, so the entire data set for the plane has to be gathered on a single process in order to evaluate Eq. (4.3).

When calculating the variance, one might compute it without storing the entire data series by manipulating Eq. (4.3) to obtain

$$\sigma_n^2 = \frac{n}{n-1}\left(\frac{1}{n}\sum_k^n x_k^2 - \mu_n^2\right) \tag{4.5}$$

which can be calculated in a single pass, thus avoiding the need to store the data series. Unfortunately, this expression runs into numerical roundoff errors if $\sigma_n^2 \ll \mu_n^2$. This is easy to see if one considers that in this case, $x_k - \mu_n$ can be expected to be a small number for each individual term in the summation of Eq. (4.3), while both $\sum_k^n x_k^2$ and μ_n^2 will be large positive numbers, so their difference might have too few significant digits[3].

Instead, we have followed the approach of [130], which provided formulas to compute the mean and variance of a data set given the means and variances of two sub-sets of the data. A similar procedure is presented for third and fourth order moments in [126] and generalized to arbitrary order moments in [97]. These formulas allowed us to calculate statistical results in a single pass, storing (and communicating across processes when needed) only a partial result for each statistic.

[3]There are ways around this issue. For example, one could use an auxiliary variable $\hat{x}_k = x_k - \hat{\mu}$, where $\hat{\mu}$ is a crude approximation of the mean, to reduce roundoff error.

4.6.1 Mean and variance

Consider a set of n samples $\mathcal{S}_n$ divided into two arbitrary subsets $\mathcal{S}_a$, $\mathcal{S}_b$, containing n_a and n_b elements respectively, such that $\mathcal{S}_n = \mathcal{S}_a \bigcup \mathcal{S}_b$ and $\mathcal{S}_a \bigcap \mathcal{S}_b = \emptyset$. The means of $\mathcal{S}_n$, $\mathcal{S}_a$ and $\mathcal{S}_b$ are related by

$$\sum_k^n x_k = \sum_i^{n_a} x_i + \sum_j^{n_b} x_j$$

$$n\mu_n = n_a\mu_a + n_b\mu_b$$

$$n\mu_n = (n - n_b)\,\mu_a + n_b\mu_b = n\mu_a + n_b\,(\mu_b - \mu_a)$$

defining $\delta_{ba} = \mu_b - \mu_a$ we can write

$$\mu_n = \mu_a + \frac{n_b}{n}\delta_{ba} \tag{4.6}$$

Eq. (4.6) can be introduced in Eq. (4.3) to obtain a single pass formula for variances. Introducing the intermediate result $M_{2,n} = (n-1)\,\sigma_n^2$,

$$M_{2,n} = \sum_k^n (x_k - \mu_n)^2 = \sum_k^{n_a} (x_k - \mu_n)^2 + \sum^b n_k\,(x_k - \mu_n)^2$$

$$= \sum_k^{n_a} \left(x_k - \mu_a + \frac{n_b}{n}\delta_{ba}\right)^2 + \sum^b n_k \left(x_k - \mu_b + \frac{n_a}{n}\delta_{ab}\right)^2 \tag{4.7}$$

Now we can develop the first term on the last line

$$\sum_k^{n_a} \left((x_k - \mu_a)^2 + \left(\frac{n_b}{n}\delta_{ba}\right)^2 - 2\,(x_k - \mu_a)\,\frac{n_b}{n}\delta_{ba}\right)$$

$$= \sum_k^{n_a} (x_k - \mu_a)^2 + n_a \left(\frac{n_b}{n}\delta_{ba}\right)^2 - 2\sum_k^{n_a} (x_k - \mu_a)\,\frac{n_b}{n}\delta_{ba}$$

$$= M_{2,n_a} + n_a \left(\frac{n_b}{n}\delta_{ba}\right)^2$$

where we have used the definition of μ_a to say $\sum_k^{n_a} (x_k - \mu_a) = 0$ in the last step. The same reasoning can be used on the second term on the last line of Eq. (4.7) to obtain

$$\sum^b n_k \left(x_k - \mu_b + \frac{n_a}{n}\delta_{ab}\right)^2 = M_{2,n_b} + n_b \left(\frac{n_a}{n}\delta_{ab}\right)^2$$

Going back to Eq. (4.7), we can use the fact that $\delta_{ab}^2 = \delta_{ba}^2$ to write

$$M_{2,n} = M_{2,n_a} + n_a \left(\frac{n_b}{n}\delta_{ba}\right)^2 + M_{2,n_b} + n_b \left(\frac{n_a}{n}\delta_{ab}\right)^2$$

$$M_{2,n_a} + M_{2,n_b} + (n_a + n_b)\,\frac{n_a n_b}{n^2}\delta_{ba}^2$$

from here we can obtain the desired expression, given in [19], which is

$$M_{2,n} = M_{2,n_a} + M_{2,n_b} + \frac{n_a n_b}{n}\delta_{ba}^2 \tag{4.8}$$

In practice, it is convenient to store the sum of all terms as an intermediate result instead of the mean. Defining $M_{1,n} = n\mu_n = \sum_k^n x_k$, Eq. (4.8) can be rewritten as

$$M_{1,n} = M_{1,n_a} + M_{1,n_b} \tag{4.9}$$

$$M_{2,n} = M_{2,n_a} + M_{2,n_b} + \frac{1}{n_a n_b n}\left(n_a M_{1,n_b} - n_b M_{1,n_a}\right)^2 \tag{4.10}$$

It is useful to particularize this expression for the case $n_a = n$, $n_b = 1$, which corresponds to adding a new measurement, x_{n+1}, to the data set

$$M_{1,n+1} = M_{1,n} + x_{n+1} \tag{4.11}$$

$$M_{2,n+1} = M_{2,n} + \frac{1}{n(n+1)}\left(nx_{n+1} - M_{1,n}\right)^2 \tag{4.12}$$

We can keep track of as many partial results as needed by storing n, $M_{1,n}$ and $M_{2,n}$ for each subset and using Eqs. (4.11) and (4.12) to update them. Once the calculation is finished and, for parallel simulations, any distributed data is gathered, the partial results can be combined using Eqs. (4.9) and (4.10) and the mean and variance of the complete record is recovered by

$$\mu_n = \frac{1}{n}M_{1,n} \qquad \sigma_n^2 = \frac{1}{n-1}M_{2,n} \tag{4.13}$$

4.6.2 Covariances

The present approach was extended to covariances in [97]. The covariance of two random variables x, y with means μ^x, μ^y can be estimated from n realizations of (x, y) as

$$c_n^{xy} = \frac{1}{n-1}\sum_k^n \left(x_k - \mu_n^x\right)\left(y_k - \mu_n^y\right) \tag{4.14}$$

which again requires a previous estimate of the mean of each variable.

We can obtain a single pass formula for the covariance using a similar procedure to that followed for variances in the previous section. We define the intermediate result $C_{2,n}^{xy} = (n-1)\, c_n^{xy}$ and use Eq. (4.6) to write μ_n^x and μ_n^y in terms of the partial results in Eq. (4.14). After rearranging the resulting expression, we obtain

$$C_{2,n}^{xy} = C_{2,n_a}^{xy} + C_{2,n_b}^{xy} + \frac{1}{n_a n_b n}\left(n_a M_{1,n_b}^x - n_b M_{1,n_a}^x\right)\left(n_a M_{1,n_b}^y - n_b M_{1,n_a}^y\right) \tag{4.15}$$

which can be particularized for the case where $n_a = n$, $n_b = 1$ as

$$C_{2,n+1}^{xy} = C_{2,n}^{xy} + \frac{1}{n(n+1)}\left(nx_{n+1} - M_{1,n}^x\right)\left(ny_{n+1} - M_{1,n}^y\right) \tag{4.16}$$

4.6.3 Third order statistics

The third order central moment of a distribution can be estimated from n samples as

$$s_x = \frac{1}{n}\sum_k^n (x_k - \mu_n)^3 \tag{4.17}$$

It can also be calculated in a single pass by means of the following formula, taken from [126]:

$$M_{3,n} = M_{3,n_a} + M_{3,n_b} + \frac{n_a n_b (n_a - n_b)}{n^2}\delta_{ba}^3 + \frac{3}{n}(n_a M_{2,n_b} - n_b M_{2,n_a})\,\delta_{ba} \tag{4.18}$$

In the present work, we have not used third order central moments, but the turbulence energy budget involves correlations between three different variables (two velocity fluctuations and a gradient). Triple correlations can be estimated using the general expression

$$c_n^{xyz} = \frac{1}{n}\sum_k^n (x_k - \mu_n^x)(y_k - \mu_n^y)(z_k - \mu_n^z) \tag{4.19}$$

which again can be transformed into a single-pass formula by using Eq. (4.6) to write the different means that appear in Eq. (4.19) in terms of the partial means in each subset. Rearranging the terms and introducing the partial result $C_{3,n}^{xyz} = nc_n^{xyz}$ produces the final expression

$$\begin{aligned}
C_{3,n}^{xyz} &= C_{3,n_a}^{xyz} + C_{3,n_b}^{xyz} \\
&+ \frac{n_a - n_b}{n_a^2 n_b^2 n^2}\left(n_a M_{1,n_b}^x - n_b M_{1,n_a}^x\right)\left(n_a M_{1,n_b}^y - n_b M_{1,n_a}^y\right)\left(n_a M_{1,n_b}^z - n_b M_{1,n_a}^z\right) \\
&+ \frac{1}{n_a n_b n}\left(n_a M_{2,n_b}^{xy} - n_b M_{2,n_a}^{xy}\right)\left(n_a M_{1,n_b}^z - n_b M_{1,n_a}^z\right) \\
&+ \frac{1}{n_a n_b n}\left(n_a M_{2,n_b}^{yz} - n_b M_{2,n_a}^{yz}\right)\left(n_a M_{1,n_b}^x - n_b M_{1,n_a}^x\right) \\
&+ \frac{1}{n_a n_b n}\left(n_a M_{2,n_b}^{zx} - n_b M_{2,n_a}^{zx}\right)\left(n_a M_{1,n_b}^y - n_b M_{1,n_a}^y\right)
\end{aligned} \tag{4.20}$$

Using Eq. (4.20), we can evaluate any triple correlation in a single pass, provided that we also keep track of the pairwise covariances and individual averages of each variable involved in the calculation. Fortunately, in the present work we were already interested in these quantities, as they appear in other terms of the turbulence energy budget, and this did not suppose an additional cost.

Finally, the particular case $n_a = n$, $n_b = 1$ results in the following simplified expres-

sion

$$C^{xyz}_{3,n+1} = C^{xyz}_{3,n} + \frac{n-1}{n^2\left(n+1\right)^2}\left(nx_{n+1} - M^x_{1,n}\right)\left(ny_{n+1} - M^y_{1,n}\right)\left(nz_{n+1} - M^z_{1,n}\right) - \frac{1}{n\left(n+1\right)}\left[M^{xy}_{2,n}\left(nz_{n+1} - M^z_{1,n}\right) + M^{yz}_{2,n}\left(nx_{n+1} - M^x_{1,n}\right) + M^{zx}_{2,n}\left(ny_{n+1} - M^y_{1,n}\right)\right] \quad (4.21)$$

4.7 Concluding remarks and future work

The parallelization of an existing code is a challenging task and requires significant efforts, some of which we presented here. As a general remark, it is important to design the code to be parallel from the start, choosing algorithms that require a minimum amount of parallel communication by design, rather than just parallelizing existing algorithms. This is particularly important in the choice of linear solvers, both because of the large impact the solution has in the total cost of the problem and because *traditional* approaches such as direct or Krylov solvers tend to parallelize poorly due to their communication requirements. We saw another example of this in the techniques for the calculation of flow statistics the algorithms of Section 4.6, where a specially designed technique is much more appropriate than the naive approach.

From the point of view of the parallel design of the code, Kratos Multiphysics currently uses blocking communication operations, in which execution of the code stops while data is being transferred between processors. Newer versions of MPI allow non-blocking communication, in which the data to be sent is defined and calculations can continue while communication goes on in the background. This could allow significant time savings for all calculations that can be written in a way where sent data is not immediately required in the receiving process.

Another addition that is being considered is the use of hybrid parallelization, based on combining a distributed approach such as the one presented here with a shared memory implementation. This could be potentially useful in machines such as the Gottfried cluster we used for our tests, where each calculation node contains multiple processors with a shared memory. In this situation, hybrid parallelization would allow us to use MPI for inter-node communication and simpler shared memory parallelization for the processors within the node.

Finally, we have devoted some time to describe our approach to partitioning the calculation mesh in Section 4.3. While this effort is essential for the initial subdivision of the domain, it is not sufficient when the calculation mesh changes during the problem, something that will happen in Chapter 5. When this happens, an approach for dynamic re-balancing of the problem would be required, reassigning nodes and elements to ensure

that the computational load on each process remains homogeneous during the entire simulation.

We want to remark that the techniques and results presented in this chapter represent some recent developments in the parallelization of the Kratos Multiphysics FEM framework and are a part of an ongoing work. In this sense, they should be understood as a snapshot of some recent developments rather than a finished result. This is particularly true for the numerical results presented in Section 4.5 as AMGCL, the solver library we are using, is still under active development.

Chapter 5

Adaptive mesh refinement for turbulent and viscoplastic flows

5.1 Introduction

One of the main advantages of finite elements is the possibility of using unstructured meshes, adapting the element size to the features of the problem and using finer resolutions near regions of interest. This requires a certain degree of familiarity with the problem being solved, as an insufficient mesh resolution can lead to missing important features such as boundary layers or sharp changes in the solution. In this sense, it would be desirable to have the ability to adaptively modify the mesh during the simulation, ensuring that its resolution is sufficient to represent the solution with a given accuracy at all times.

The first problem related to this strategy is how to quantify the error in the solution. One popular approach to this question are *a posteriori* error estimation techniques (see for example [1, 2, 61, 96]), where the computed solution itself is used to assess its accuracy. We explore one of such methods, closely related to the VMS stabilized formulations introduced in Chapter 2, which was originally presented in [50] for convection-diffusion problems and has been applied to the Navier-Stokes equations in [51, 106].

Once we are able to estimate the error and its distribution across the domain, we can modify the calculation grid to increase the overall accuracy. Many different mesh modification strategies have been proposed since the pioneering work in the late eighties [70, 98, 110], involving for example local mesh modifications [133], movement of mesh nodes through the solution of an auxiliary elasticity problem [20], mesh morphing techniques [112] or edge stretching [48]. However, mesh refinement, and meshing in general, is a particularly challenging problem in a distributed memory setting, as maintaining mesh quality over the entire domain and ensuring that the mesh is conforming is a very non-local problem, requiring communication across the different parallel subdomains.

We propose a mesh refinement algorithm based on edge division that was designed with parallel performance as its main goal, requiring as little information that is not local to each element as possible.

Our mesh refinement strategy was originally conceived as a complement to the incompressible fluid solver of Chapter 2, with the goal of increasing the reliability of the solution and optimizing the number of elements for the simulation of large turbulent flow problems with meshes in the LES range. However, during the development of the present work, an opportunity appeared to use the same approach in the simulation of viscoplastic fluids.

Viscoplastic fluids are a class of non-Newtonian fluids that are rigid unless the applied shear stress is larger than a threshold, known as the yield stress. We concentrate our attention on Bingham fluids in which, once the material starts to flow, shear stress varies linearly with strain rate increments. Adaptive mesh refinement techniques, involving full remeshing of the domain, have been used in the past to simulate plane and axisymmetric Bingahm fows [104, 105, 109] and also to simulate depth-integrated $3D$ free surface flows using a shallow water approximation [13]. We will apply our approach to the simulation of both plane and fully tri-dimensional cases.

The present chapter is organized as follows: first, the complete refinement procedure and its parallel implementation are presented in Section 5.2, followed by their application to incompressible flows in Section 5.3. Section 5.4 introduces the solver used for the simulation of non-Newtonian flows, which is applied to the simulation of Bingham fluids in Section 5.5. Some final remarks and conclusions are given in Section 5.6.

5.2 Adaptive refinement strategy

The refinement procedure we use is based on creating new elements by edge division. New nodes are inserted on edge mid-points of elements where the solution is considered to have a too large error. Once the new nodes have been added, the mesh is adapted to include the new nodes by subdivision of the existing elements. This approach involves several clearly differentiated components, acting in succession:

- An *a posteriori* error estimator to identify regions of the model that require refinement.
- A global refinement procedure that identifies the elemental edges to be divided and creates the new nodes.
- A local refinement technique to divide existing elements according to the patterns defined by the new edges and nodes.

In some examples we have considered one additional step, using local mesh improvement techniques to minimize the distortion of the elements in the new mesh.

It is important to note that the error estimation and the mesh refinement technique are independent of each other. This would allow us to combine the error estimator we present with a completely different refinement strategy, such as a full remeshing using Delaunay triangulation. Conversely, the only input required by the mesh refinement strategy is knowing which elements must be divided. This means that it could be used in combination with a different error estimator. For example, by choosing an estimator not tied to VMS stabilization, we could use the same refinement technique in combination with the FIC solver presented in Chapter 3.

5.2.1 Error estimation

The error estimator we use is based on the ideas presented by Hauke *et al.* in [50] for fluid transport problems. The same approach was applied to the Navier-Stokes equations in [51] and in [106], where part of the material that constitutes this chapter was originally presented. This estimator is motivated by the scale separation introduced in VMS formulations, which we presented in Chapter 2 for Newtonian flows and will be used again in Section 5.5 to stabilize viscoplastic flows.

Considering that the large scale variables represent the part of the solution reproduced by the finite element mesh, and small scale variables represent the part that is not resolved, the later can be understood as a measure of the local error in the solution. Once the large scale flow has been solved, the small scale can be evaluated on chosen points in the domain and its magnitude can be used as an *a posteriori* error estimator, identifying regions where the mesh is too coarse.

Following this argument, we define the following estimate of the error in element e, to be used in VMS stabilized formulations

$$I\left(e\right) = \left\|\boldsymbol{u}_s\right\|_{\Omega_e} = \left(\int_{\Omega_e} \boldsymbol{u}_s \cdot \boldsymbol{u}_s \,\mathrm{d}\Omega\right)^{\frac{1}{2}} \tag{5.1}$$

In the present chapter we are mainly concerned with the static versions of the ASGS and OSS methods, which will be briefly recalled. The small scale model for the static ASGS formulation was defined in Chapter 2 as

$$\begin{aligned} \boldsymbol{u}_s &= \tau_1 \boldsymbol{R}^m\left(\boldsymbol{u}_h, p_h\right) \\ &= \tau_1 \left(\boldsymbol{f} - \rho\partial_t \boldsymbol{u}_h - \rho\frac{1}{2}\left(\boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla \cdot \left(\boldsymbol{u}_h \otimes \boldsymbol{u}_h\right)\right) - \nabla \cdot 2\mu\nabla^s \boldsymbol{u}_h - \nabla p_h\right) \end{aligned} \tag{5.2}$$

while the corresponding model for the static OSS method was introduced as

$$\begin{aligned}\boldsymbol{u}_s &= \tau_1 \left(\boldsymbol{R}^m \left(\boldsymbol{u}_h, p_h \right) - \boldsymbol{\Pi}^m \right) \\ &= \tau_1 \left(\boldsymbol{f} - \rho \partial_t \boldsymbol{u}_h - \rho \frac{1}{2} \left(\boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla \cdot \left(\boldsymbol{u}_h \otimes \boldsymbol{u}_h \right) \right) - \nabla \cdot 2\mu \nabla^s \boldsymbol{u}_h - \nabla p_h - \boldsymbol{\Pi}^m \right)\end{aligned} \tag{5.3}$$

Replacing the small scale velocities in Eq. (5.1) by their definition we obtain the final expression for the estimator. Using Eq. (5.3), the OSS estimator can be expressed as

$$I\left(e\right) = \left(\int_{\Omega_e} \tau_1^2 \left\| \boldsymbol{R}^m - \boldsymbol{\Pi}^m \right\|^2 \, \mathrm{d}\Omega \right)^{\frac{1}{2}} \tag{5.4}$$

On the other hand, for the ASGS formulation, we use the definition for the small scales given by Eq. (5.2). This leads to an expression that is equivalent to dropping the projection $\boldsymbol{\Pi}^m$ from Eq. (5.3).

To perform a refinement step we evaluate the error estimate $I(e)$ on each element and identify those elements where the estimate is larger than a pre-defined tolerance. These elements are then split according to the algorithm described in the following section.

It is worth noting that multiple variants of a subscale based error estimator were presented in [50]. The version presented here corresponds to the case where no elemental boundary integrals are included in the indicator. We chose to do so to keep consistency with the actual small scale model we use in the calculations. The boundary terms in question appear when keeping the elemental boundary integrals in the derivation of the stabilized VMS formulation. Since we neglected these terms in our derivation (as was shown in Chapter 2), we will not take them into account here.

5.2.2 Mesh refinement strategy

The mesh refinement strategy we use is based on the local subdivision of existing triangles or tetrahedra by edge division. The procedure was designed to require only a minimal amount of information that is not local to the elements to be refined, in order to simplify its implementation in a distributed memory environment. The main idea of the refinement algorithm is that, once an element has been identified as a candidate for refinement, it is divided into new elements created by splitting the edges of the original element in half. Neighboring elements are then refined by splitting some of their edges as required to maintain consistency. The main steps of this procedure are shown graphically in Figure 5.1 and can be described as follows:

1. Iterate over mesh elements, evaluating the error estimator of Eq. (5.1).

2. If the error estimate is larger than a predefined tolerance in a given element, mark it and its edges as *needing refinement* (Fig. 5.1(a)).

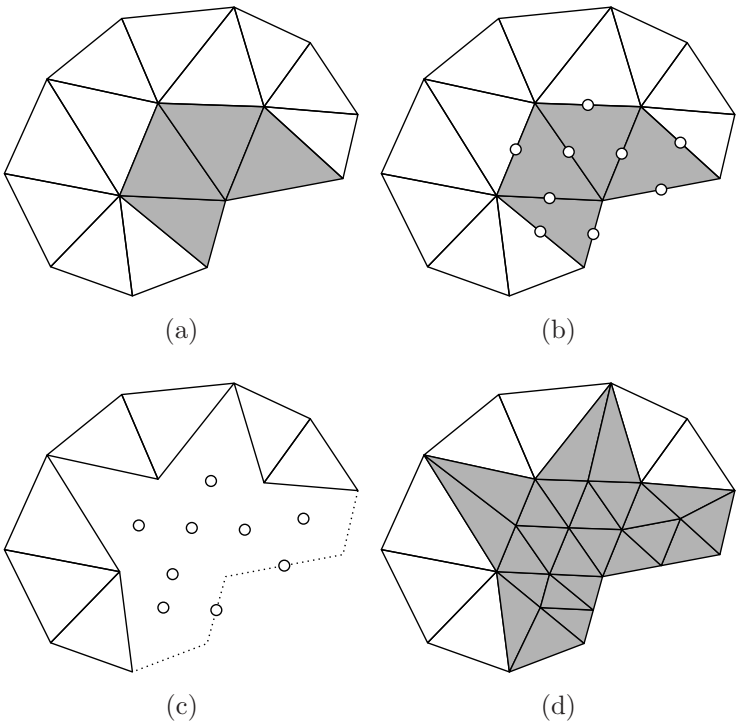


Figure 5.1: Refinement procedure: (a) identify elements to refine; (b) divide edges and insert new nodes; (c) remove all elements with split edges; (d) create new elements to recover a conforming mesh.

3. A new node is created in the mid-point of all edges that have been identified as *needing refinement*. All nodal data (and in particular initial guesses for velocity and pressure) is interpolated from the nodes that define the edge (Fig. 5.1(b)).

4. All elements with refined edges are deleted. Note that this includes elements where the error estimator was not larger than the tolerance (Fig. 5.1(c)).

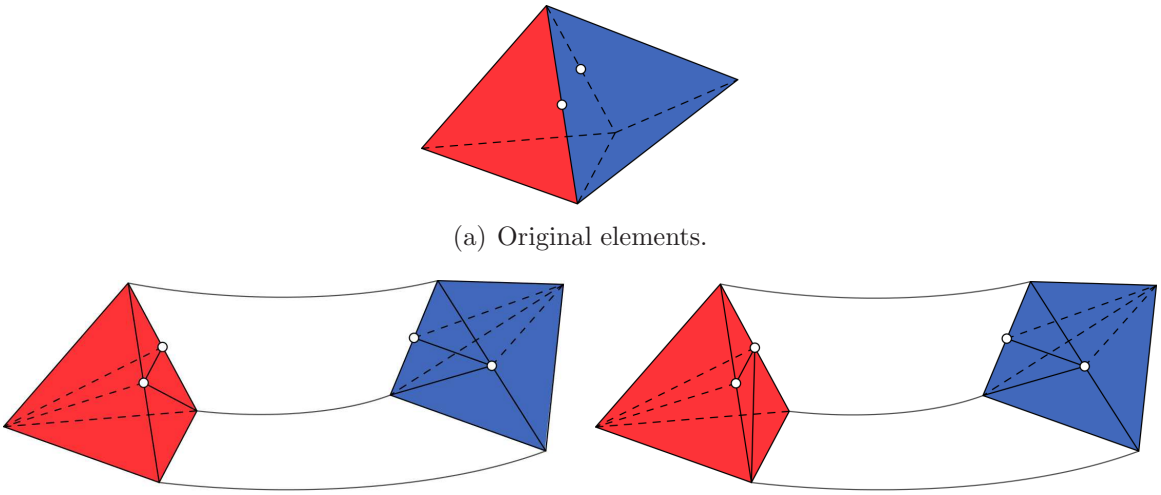
5. New elements are created using predefined patterns (Fig. 5.1(d)).

Regarding the last step, the creation of new elements is done according to predefined subdivision patterns, based on which and how many edges of each triangle or tetrahedra were subdivided in Step 2.

5.2.3 Local refinement of triangles and tetrahedra

The mesh refinement strategy described in the previous section is based on inserting nodes on edge midpoints and subdividing existing elements to include the new nodes. Unfortunately, knowing only which edges to split in each tetrahedron is not enough to ensure that the resulting mesh will be conforming. Consider for example the two elements

in Fig. 5.2, where two new nodes are inserted. There are two possible ways in which the common face can be divided, corresponding to the two tetrahedra in Fig. 5.2(c). To obtain a conforming mesh, the splitting strategy has to be designed in a way that ensures that the division of each element results in conforming faces.



(a) Original elements.

(b) Conforming division.

(c) Non-conforming division.

Figure 5.2: Division of tetrahedra based on edge splitting.

This restriction is problematic in a distributed memory environment, as the elements involved might belong to different partitions. In such situation, obtaining information from the neighbor element is not straightforward and involves communication between the partitions.

To solve this issue, we use an approach that is based exclusively in the numbering of the existing nodes, which is both available locally in the partition and known to be identical across partitions, avoiding all parallel communication. We start by assuming that all the edges of the element are split by their midpoints, resulting in four smaller elements in the case of a triangle or eight in a tetrahedra. If a given edge is not divided, at least one of the new elements touching that edge is eliminated. An example of this operation, which we call a *collapse*, is presented in Fig. 5.3, in which the four potential elements obtained by the full refinement of a triangle are reduced to three.

To choose between the two possible outcomes of the collapse, the global node indexes of nodes a and b are compared. The partitioning in Fig. 5.3(b) is adopted if the index of node a is smaller and the division of Fig. 5.3(c) is used otherwise. When subdividing two tetrahedra that share a face, using this criterion ensures that the subdivisions obtained from each element will be conforming. It must be noted that this approach has the obvious drawback of not taking the quality of the refined elements into account, but on the other hand it is purely local and therefore easily parallelizable.

Using this strategy we can determine the shape of the refined triangles or, in the

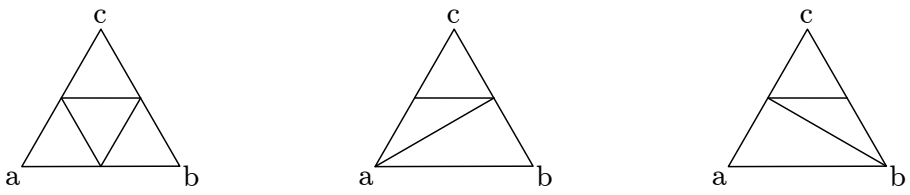


Figure 5.3: Collapse operations on a triangle: (a) no collapse; (b) edge collapsed towards node a or (c) collapsed towards node b.

case of a tetrahedron, the way in which its faces will be refined. For tetrahedra, the volume is then divided according to predefined patterns, depending on the shape of its faces.

The splitting operation defines three possible situations on each edge of the tetrahedron, corresponding to the three cases in Fig. 5.3. Given that there are six edges on each tetrahedron, up to $3^6 = 729$ potential face patterns can be defined. Each of those is matched to a division pattern using an utility[1] distributed within Kratos Multiphyisics [34] which implements the local refinement operations. This tool, given an array containing the global indices for the nodes involved in the subdivision, divides the faces according to the collapse criterion and produces the connectivities for the new elements to be created. Note that, for some division patterns, a new node is added on the element's center of mass to improve the quality of the resulting elements.

A very important point to be made here is to consider how this procedure affects mesh quality. If all edges of an element are divided, the refinement procedure ensures that its quality is preserved, as all new elements have the same angles as the original in this case. Unfortunately, this is not the case for elements where only some of the edges are refined, and mesh quality can be significantly degraded if this kind of partial refinement is performed repeatedly over the same patch of elements.

To alleviate the problem, we improve the quality of the resulting mesh using local mesh improvement techniques. For $2D$ simulations, we used the algorithm presented in [42], which is based on reconnecting the nodes of adjacent elements. For $3D$ cases, we used the procedure presented in [36], which consists in analyzing clusters of adjacent elements and changing the local topology, adding or deleting nodes and reconnecting elements to improve mesh quality within the cluster. As these techniques were incorporated in the code on a late stage in the presented work, they were not used for the turbulent flow cases. However, we took full advantage of them to simulate the Bingham flow cases presented in Section 5.5.

[1]The source code that provides this functionality can be found in the address `https://kratos.cimne.upc.es/projects/kratos/repository/changes/kratos/kratos/utilities/split_tetrahedra.h` (retrieved on July 22, 2015).

5.2.4 Parallel implementation

The proposed mesh refinement strategy has been implemented for a distributed memory environment within the Kratos Multiphysics framework. This means that the domain partitioning model introduced in Chapter 4 is also used here. Individual elements are assigned to a single parallel subdomain and are completely unknown by the others. Nodes are also assigned to a subdomain, which holds the reference values for all data associated to that node, but might be also needed in neighboring domains if they are connected to elements lying on different partitions. When this happens, other partitions store a copy of the nodal data, which has to be updated at different points during the simulation. From the point of view of a given subdomain, nodes are said to be *local nodes* when that subdomain holds the reference values for their data or *ghost nodes* otherwise.

Our implementation of the refinement strategy relies on an auxiliary sparse matrix that reproduces the edge connectivity pattern of the mesh. As in Chapter 4, we rely on the implementation of sparse distributed memory matrices provided by the Trilinos library [52] to define and store it.

We consider a simplified scenario to clarify the presentation of the algorithm. Consider an initial finite element mesh containing N nodes and M elements divided in K subdomains. The nodes are numbered consecutively in the range $0 \leq i < N$ and ordered by subdomains, such that, if a node is local to subdomain k, its index will be in the range $n_k \leq i < n_{k+1}$. We remark that this is done to simplify the notation used to describe the implementation, but it does not represent a restriction on the actual implementation. In practice we only assume that the nodes are numbered sequentially as a whole, even if consecutive nodes are not in the same partition. Additionally, we denote the edges of finite elements in the mesh as pairs of node indices (i, j), with $i < j$.

We define a sparse matrix $\boldsymbol{E}$ of size $N \times N$ representing the edge connectivity pattern. Position E_{ij} can only be different from zero if there is some elemental edge (i, j) joining nodes i and j. Note that, using our definition of the edge, $\boldsymbol{E}$ only contains terms above its diagonal. Matrix $\boldsymbol{E}$ is stored using compact storage by rows (CSR) notation and distributed, meaning that each parallel process k stores non-zero terms for rows in the range $n_k \leq i < n_{k+1}$. Note that the process will sometimes need to access rows outside this range, corresponding to ghost nodes in the partition, an operation that requires parallel communication. In the following we refer to the set including all local and ghost nodes known to partition k as S_k.

The implementation of the refinement algorithm involves multiple tasks that can be grouped into the following steps:

1. Build a sparse matrix of edge connectivities $\boldsymbol{E}$, where position E_{ij} of the matrix only exists if there is an element edge joining nodes i and j in the mesh. Positions corresponding to an edge are initialized to the arbitrary value `-1`. This is represented in pseudo-code in Algorithm 5.1. Note that, as the matrix is defined

as upper-triangular only the positions where $i < j$ are considered.

Algorithm 5.1 Initialization of the matrix of edge connectivities $\boldsymbol{E}$.

```
1: Initialize empty sparse matrix E
2: for all Elements do
3:     for all Edges (i, j), with i < j do
4:         E(i, j) = −1
5:     end for
6: end for
```

2. Iterate over the elements, checking if they should be refined. If an element is identified as needing refinement, set the positions corresponding to its edges in $\boldsymbol{E}$ to `-2` (Algorithm 5.2). Note that this step requires parallel communication to ensure that edge data is consistent on all subdomains.

Algorithm 5.2 Identification of edges to refine.

```
1: for all Elements do
2:     if Refinement needed then
3:         for all Edges i, j, with i < j do
4:             if E(i, j) is not −2 then
5:                 E(i, j) = −2
6:                 Synchronize E(i, j)
7:             end if
8:         end for
9:     end if
10: end for
```

3. The number of new nodes t to be added in each subdomain is equal to the number of `-2` entries in all known rows of $\boldsymbol{E}$. Use this information to assign a range of temporary node indices r_k, r_{k+1} to each process (Algorithm 5.3). Some new nodes will be counted twice, as they are known by multiple domains.

4. Each process claims ownership of known new nodes by assigning it node index in its range. Note that this process will sometimes produce clashes, as nodes in the interface between subdomains are known by multiple processes (Algorithm 5.4). We decided to allow overwriting node indices set by other processes, so that the last process to assign an index will own the node.

Algorithm 5.3 Definition of temporary node ranges for process k.

```
1: t = 0
2: for all i ∈ S_k do
3:     for all j in non-zero columns of row i do
4:         if E(i, j) is −2 then
5:             t = t + 1
6:         end if
7:     end for
8: end for
9: Gather P as the sum of new nodes for all processes p < k
10: Set r_k = N + P and r_{k+1} = N + P + t
```

Algorithm 5.4 Definition of node ownership.

```
1: m = r_k
2: for all i ∈ S_k do
3:     for all j in non-zero columns of row i do
4:         if E(i, j) is not −1 then
5:             E(i, j) = m
6:             Communicate change in E to other processes if needed.
7:             m = m + 1
8:         end if
9:     end for
10: end for
```

5. Each process creates new nodes locally on edge midpoints (Algorithm 5.5). Nodal data is initialized by averaging the values stored by the nodes on both ends of the edge. The new node must be created on all partitions that know it, independently of which process owns the node, as node ownership only determines which process holds the reference data during synchronization.

6. Create new elements locally using the subdivision procedure described in Section 5.2.3 (Algorithm 5.6). Note that all elements with one or more refined edges must be subdivided, not just those identified using the refinement criteria. Recall that, in $3D$, new nodes can be created at the element center in some cases, to avoid creating highly distorted elements. If this happens, the node is created as local to the partition that owns the element and its nodal data is interpolated from the nodes of the original tetrahedron.

Algorithm 5.5 Creation of new nodes.

```
1: for all i ∈ S_k do
2:     for all j in non-zero columns of row i do
3:         if E(i, j) > 0 then
4:             k = E(i, j)
5:             Create node with index k.
6:             Initialize node k using data from nodes i and j.
7:         end if
8:     end for
9: end for
```

Algorithm 5.6 Creation of new elements.

```
1: for all Elements do
2:     if Some edge refined then
3:         Divide element
4:         if New nodes created then
5:             Interpolate data for new node using existing element's nodes.
6:         end if
7:     end if
8: end for
```

7. The procedure used to assign new nodes, combined with the fact that new nodes can be unexpectedly created during elemental subdivision, means that the final node numbering might not be consecutive or, worse, contain duplicate indices. This is also an issue with elements: as original elements are erased there will be gaps on the element numbering. Renumber nodes and elements across processes so that indices are unique and consecutive again.

8. If desired, the local mesh improvement procedure mentioned in Section 5.2.2 can be used as a post-process to increase mesh quality at this point.

9. Finally, the communication strategy used to synchronize nodal data must be re-defined to account for new nodes created on interface edges. This is done using the coloring procedure described in Chapter 4.

5.2.5 Scalability test

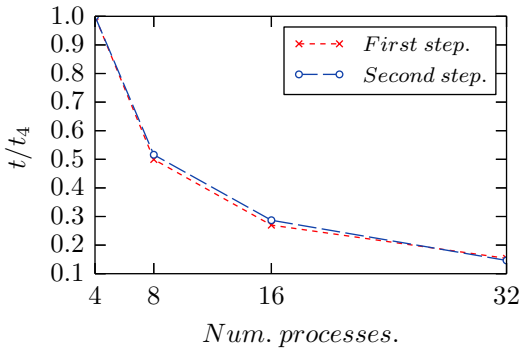
As a first test of the refinement algorithm and its parallel implementation we refined a tetrahedral mesh homogeneously. We define a cubic domain given by its corners $(-1, -1, -1)$ and $(1, 1, 1)$ and generate an initial mesh composed of slightly over one

million tetrahedra. This mesh, as all initial meshes used in later examples, has been generated using the pre-processing module of GiD [22]. The domain is fully refined in two passes, first splitting all elements to obtain about 8 million tetrahedra and again for a total of 64 million elements. The local mesh improvement strategy is not used in this case, as we are interested in testing the implementation of the refinement algorithm and its performance.

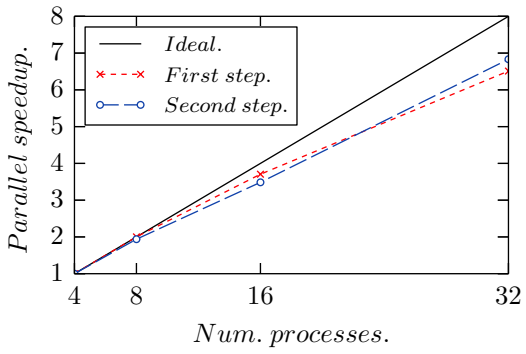
Times required to complete this operation using different amounts of parallel processes is recorded in Table 5.1. Reported times were obtained using up to three Intel Xeon E5645 processors, each consisting in two six-core CPU at 2.40 GHz connected to 48 GB RAM each. Communication between processors was done through an Infiniband connection. The results are also presented in graphical form in Fig. 5.4.

	First step		Second step	
Num. procs.	Wall time (s)	Speedup	Wall time (s)	Speedup
4	21.16	1.0	170.74	1.0
8	10.57	2.0	88.09	1.9
16	5.71	3.7	48.99	3.5
32	3.25	6.5	24.99	6.8

Table 5.1: Homogeneous refinement test – wall times and parallel speedup.



(a) Relative wall time.



(b) Parallel speedup.

Figure 5.4: Homogeneous refinement test – parallel performance.

While the test results show good parallel performance, it must be remarked that this experiment does not take into account an important aspect in practice: when refinement is not homogeneous, the number of elements in each subdomain changes and, without a load balancing strategy, parallel performance may degrade as different processors can have very different loads if refinement is concentrated on a few of the original subdomains.

5.3 Application to laminar and turbulent flows

We simulated a first set of cases combining the incompressible fluid solver presented in Chapter 2 with the refinement algorithm introduced in the previous pages. All simulations were performed with the static OSS method, coupled with the corresponding error estimator, given by Eq. (5.4). With these tests, our aim was to test the applicability of our refinement strategy on distributed memory simulations, rather than obtaining high quality solutions.

5.3.1 Flow around a cylinder at Re = 100

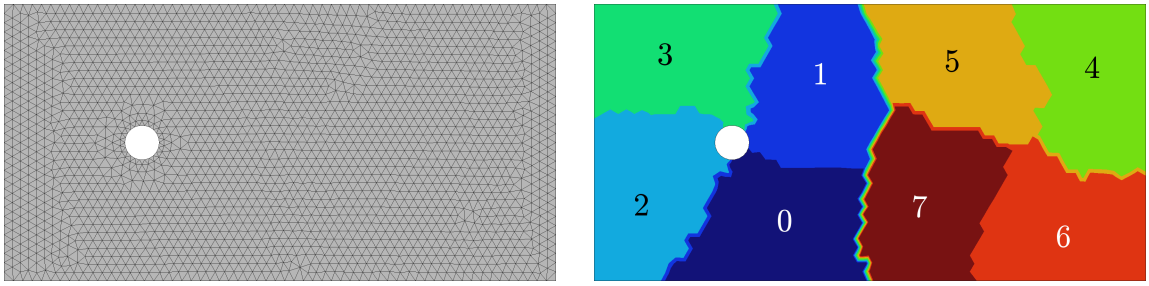
As a relatively simple, small scale application we simulate the two-dimensional flow around a cylinder using static OSS stabilization. The set up of the problem, taken from [28], consists in a cylindrical obstacle of diameter $D = 1\,\mathrm{m}$ centered on the origin of a domain defined by the range $[-4D, 12D]\times[-4D, 4D]$. A horizontal velocity $U = 1\,\mathrm{m/s}$ is imposed on the left side of the domain, while a no-penetration condition is set on the top and bottom edges. The velocity is fixed to zero on the cylinder. The fluid properties are defined as $\rho = 1\,\mathrm{Kg/m^3}$ and $\mu = 0.01\,\mathrm{Pa\cdot s}$, such that the Reynolds number in terms of the diameter is $\mathrm{Re} = 100$.

Starting from a uniform structured mesh containing 3984 linear triangles, 60 seconds of flow are simulated using a time step of 0.1 seconds. In this case, the refinement strategy is used after a starting phase to allow the simulation to transition from the initial condition to a dynamic solution. The refinement procedure is then used every 20 solution steps, resulting in a total of 40 refinement passes. The maximum number of refinements over a single original element is limited to 3 to preserve mesh quality and prevent excessive refinement on localized areas.

Although the problem is small enough to be run on a desktop computer, the simulation was performed in a distributed memory machine, using 8 processes to test the parallel implementation. The initial mesh and the partition into parallel subdomains are shown in Fig. 5.5.

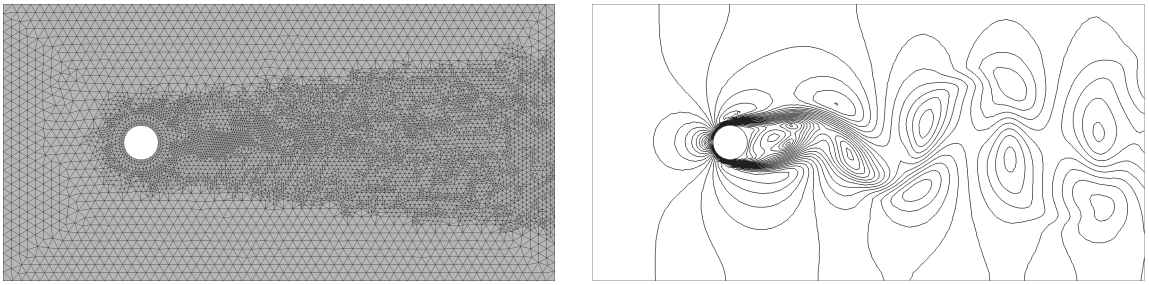
At the end of the simulation the mesh had been refined to contain a total of 11666 elements, as shown in Fig. 5.6. The total number of elements per subdomain at the begining and the end of the simulation is shown in Table 5.2. Observing the velocity isolines in Fig. 5.6(b), it can be seen that the zones that have been refined generally coincide with high velocity gradient zones in the front of the cylinder and the vortex tail that forms after it.

From the point of view of load balancing, the element counts in Table 5.2 show that refinement is not homogeneous across subdomains and, at the end of the simulation, the largest domain has close to three times more elements than the smallest one. This suggests that the practical applicability of the method in a parallel context would be much greater if it included a dynamic load balancing strategy so that elements are



(a) Initial mesh. (b) Parallel subdomains.

Figure 5.5: Cylinder test – initial mesh and parallel partition.



(a) Final mesh. (b) Isolines of velocity magnitude.

Figure 5.6: Cylinder test – final mesh and velocity isolines.

always distributed evenly across subdomains.

Process	0	1	2	3	4	5	6	7	Total
Initial	496	510	501	493	494	490	505	495	3984
Final	1056	1871	836	740	1884	1694	1542	2043	11666

Table 5.2: Cylinder test – distribution of elements per partition.

5.3.2 Flow around a 6 meter cube

Additionally, we studied a realistic three-dimensional case in order to test the refinement strategy on a tetrahedral mesh. We chose to simulate the flow around the Silsoe cube, a benchmark problem for the flow over bluff bodies. This test case reproduces an experiment performed at the Silsoe Research Institute [103] in which a cube with 6 meter sides was placed under incoming wind.

We simulate the configuration where the average wind direction is perpendicular to one of the cube faces. The cube is placed at a distance of 60 m from the inlet, while the

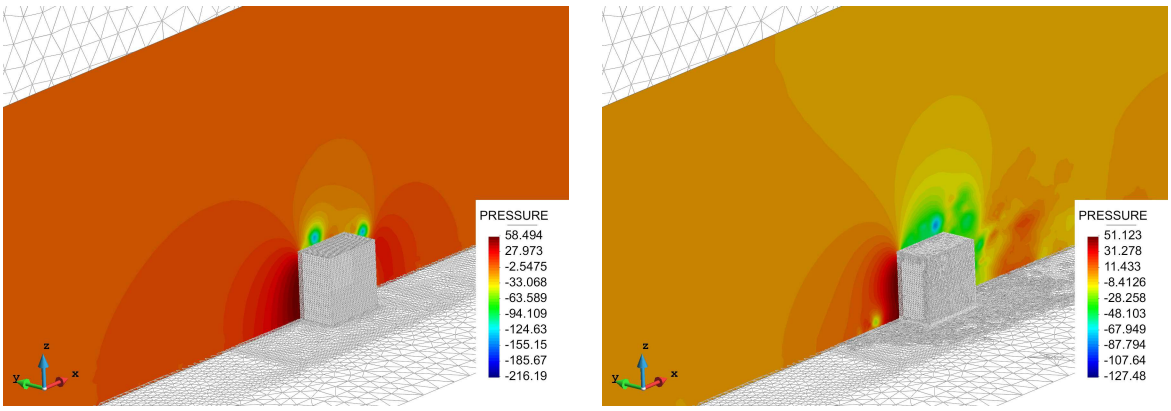
incoming wind is defined according to the logarithmic wind profile of Eq. (5.5) (see for example [101]).

$$\frac{u_x(z)}{u_\tau} = \frac{1}{\kappa}\log\left(\frac{z\,u_\tau}{\nu}\right) + B \tag{5.5}$$

For the present simulation, we define $u_\tau = 0.272\,\mathrm{m/s}$ as the friction velocity, a kinematic viscosity of $\nu = 1.51 \times 10^{-5}\,\mathrm{m^2/s}$, corresponding to air, a value of $\kappa = 0.41$ for Von Kármán's constant and $B = 5.2$. An air density $\rho = 1.225\,\mathrm{Kg/m^3}$ is assumed. We simulate a flow time of 6 seconds, using a time step of 0.1 seconds.

The simulation domain has a total length of 108 m in the direction of the mean flow (x), 48 m in the cross-wind direction (y) and a total height of 30 m in the z direction. The refinement algorithm is used for the first time after 20 solution steps of the flow problem and every 10 steps from then on, for a total of five refinement iterations. In this case, the tolerance for the error indicator is set relative to the average velocity of the flow, so elements are refined if the magnitude of the small scale on the center of the element is larger than 5% of the average large-scale velocity. To preserve mesh quality, only two refinements are allowed over a single original element.

The initial mesh is composed of 1.6 million tetrahedral elements, refined to a total of 5.3 million elements at the end of the simulation. The instantaneous pressure contours on the central x–z plane obtained for time 1 s, along with the original mesh, are shown in Fig. 5.7(a). The same contours for time 6 s and the final mesh are shown in Fig. 5.7(b).



(a) Pressure distribution at $t = 1\,\mathrm{s}$, original mesh (b) Pressure distribution at $t = 6\,\mathrm{s}$, refined mesh

Figure 5.7: Silsoe cube – pressure results on the central plane and simulation meshes.

We observe that the refined elements are concentrated near the cube and its immediate wake. In the front of the cube, they are roughly placed along the area where the flow starts to deviate due to the obstacle. Experimental results show that a horseshoe vortex should develop close to the ground in that zone, and the distribution of elements seems to follow it. On the wake region, the refined area develops progressively as vortices

detach from the cube and travel downstream. If the simulation was run for a longer period, it should be expected that the refined area would advance towards the exit to obtain a fully refined wake, as happened previously in the cylinder example.

Finally, we want to remark that this simulation was performed to test the refinement algorithm in a realistic setting and not to obtain precise results. The Silsoe cube problem was analyzed using the same solver (without adaptive mesh refinement) in [117], obtaining good agreement with the benchmark solutions.

5.4 A FEM solver with adaptive mesh refinement for viscoplastic flows

The adaptive mesh refinement procedure presented in Section 5.2 can also be applied to the simulation of viscoplastic flows. For this we use a modified version of the VMS based solver presented in Chapter 2, adapted to the solution of the steady Navier-Stokes equations with non-Newtonian constitutive relations, which will be introduced here.

5.4.1 Model equations

Defining the fluid's velocity $\boldsymbol{u}$ and density ρ, the stress tensor $\boldsymbol{\sigma}$ and the external forces $\boldsymbol{f}$, the conservation of linear momentum can be stated as

$$\rho\, \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nabla \cdot \boldsymbol{\sigma} = \boldsymbol{f} \tag{5.6}$$

and the conservation of mass implies that, for an incompressible fluid,

$$\nabla \cdot \boldsymbol{u} = 0 \tag{5.7}$$

The definition of the problem is completed by introducing a model domain Ω with boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$ and appropriate boundary conditions

$$\boldsymbol{u} = \boldsymbol{u}_0 \qquad \text{on } \Gamma_D \tag{5.8}$$

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{t} \qquad \text{on } \Gamma_N \tag{5.9}$$

where $\boldsymbol{u}_0$ is the imposed velocity on the Dirichlet boundary Γ_D, $\boldsymbol{n}$ is the outwards unit normal on the Neumann boundary Γ_N and $\boldsymbol{t}$ are the imposed tractions.

The stress tensor of Eq. (5.6) can be decomposed into a volumetric part involving the pressure p and a deviatoric stress tensor $\boldsymbol{\tau}$:

$$\boldsymbol{\sigma} = -p\, \boldsymbol{I} + \boldsymbol{\tau} \tag{5.10}$$

where $\boldsymbol{I}$ is the second order identity tensor.

A constitutive model is required to close the formulation, giving an expression for the deviatoric stresses $\boldsymbol{\tau}$. A broad class of fluid materials follow a generalized Newtonian law given by

$$\boldsymbol{\tau} = 2\eta \boldsymbol{S} \tag{5.11}$$

where the apparent viscosity η is, in general, a variable that depends on the characteristics of the flow and $\boldsymbol{S}$ is the strain rate tensor, defined as

$$\boldsymbol{S} = \nabla^s \boldsymbol{u} = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right) \tag{5.12}$$

Additionally, it is convenient to introduce the following tensor invariants to measure the magnitude of the strain rate and deviatoric stresses

$$\dot{\gamma} = \sqrt{2\,\boldsymbol{S} \,:\, \boldsymbol{S}} \qquad \|\boldsymbol{\tau}\| = \sqrt{\frac{1}{2}\,\boldsymbol{\tau} \,:\, \boldsymbol{\tau}} \tag{5.13}$$

In the present work, a regularized Bingham model will be used. A Bingham fluid [14, 85] is a material that remains rigid while the applied shear stress is lower than its yield stress τ_0. Once this limit is reached, the material starts flowing with a constant viscosity μ_p, the plastic viscosity. This can be expressed using the notation of Eq. (5.11) if the apparent viscosity is defined as:

$$\begin{aligned} \eta &= \infty && \text{if} \quad \|\boldsymbol{\tau}\| < \tau_0 \\ \eta &= \mu_p + \frac{\tau_0}{\dot{\gamma}} && \text{if} \quad \|\boldsymbol{\tau}\| \geq \tau_0 \end{aligned} \tag{5.14}$$

Unfortunately, the discontinuous nature of the Bingham model introduces numerical difficulties, as the apparent viscosity η is infinite for small strain rates that don't produce flow. As a way to prevent this issue, we have adopted the regularized equation proposed by Papanastasiou [95], replacing Eq. (5.14) with

$$\eta = \mu_p + \frac{\tau_0}{\dot{\gamma}}\left(1 - e^{-m\dot{\gamma}}\right) \tag{5.15}$$

where m is a regularization coefficient with units of time. Using this regularized expression, it can be shown that $\eta \to \mu_p + m\tau_0$ when $\dot{\gamma} \to 0$ and the apparent viscosity is never infinite.

The regularized law of Eq. (5.15), which tends to the original Bingham model of Eq. (5.14) as m increases, is compared to an ideal Bingham fluid in Fig. 5.8. Note this is not the only choice for a regularized formulation, other alternatives have been presented in [77, 115, 121].

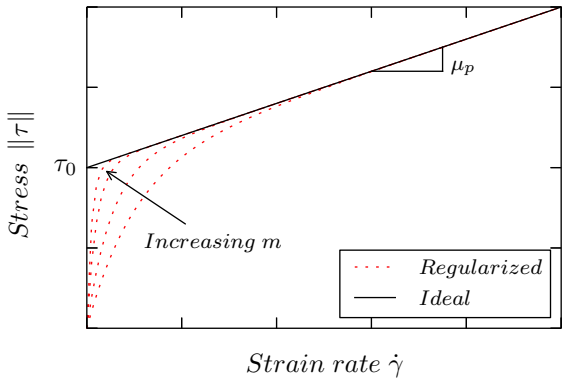


Figure 5.8: Bingham model.

5.4.2 Stabilized formulation

To introduce the variational form of the problem we define spaces V and Q containing the exact values $\boldsymbol{u}$ and p, respectively, of the solution of the problem in Ω. Additionally, we define the test functions $\boldsymbol{w} \in V_0$, $q \in Q$, where V_0 is defined as V restricted to the zero Dirichlet condition $\boldsymbol{w} = 0$ on Γ_D.

The variational form of the problem can be obtained by multiplying Eqs. (5.6) and (5.7) by test functions $\boldsymbol{w}$, q and integrating over the simulation domain Ω:

$$\begin{aligned} \int_\Omega \boldsymbol{w}\, \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u}\, \mathrm{d}\Omega + \int_\Omega 2\eta \nabla^s \boldsymbol{w} : \nabla^s \boldsymbol{u}\, \mathrm{d}\Omega \\ - \int_\Omega \nabla \cdot \boldsymbol{w}\, p\, \mathrm{d}\Omega = \int_\Omega \boldsymbol{w} \boldsymbol{f}\, \mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w} \boldsymbol{t}\, \mathrm{d}\Gamma \end{aligned} \tag{5.16}$$

$$\int_\Omega q \nabla \cdot \boldsymbol{u}\, \mathrm{d}\Omega = 0 \tag{5.17}$$

where some terms have been integrated by parts.

As seen in the previous chapters, the Galerkin weak form of the Navier-Stokes equations suffers from stability issues. In this case, we used the static versions of both the ASGS and OSS formulations to stabilize it. The choice of using a VMS based formulation has an added benefit: it allows us to use the error estimator presented in Section 5.2.1 without modification.

We introduce the discrete solution as $\boldsymbol{u}_h \in V_h$, $p_h \in Q_h$, where $V_h \subset V$ and $Q_h \subset Q$ are the discrete spaces defined by the finite element interpolation. Given these definitions, the VMS approach is based in decomposing the problem variables into a large scale part, identified with the finite element solution, and a small scale part:

$$\boldsymbol{u} = \boldsymbol{u}_h + \boldsymbol{u}_s \qquad p = p_h + p_s \tag{5.18}$$

where the small scale variables $\boldsymbol{u}_s$, p_s represent the part of the continuous solution that is not resolved with the discrete interpolation.

Next we introduce the scale separation of Eq. (5.18) in the variational problem given by Eqs. (5.16) and (5.17). Using test functions belonging to the discrete space $\boldsymbol{w}_h \in V_h$, $q_h \in Q_h$, we obtain

$$\begin{aligned} &\int_\Omega \boldsymbol{w}_h \, \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h \, \mathrm{d}\Omega + \int_\Omega 2\eta \nabla^s \boldsymbol{w}_h : \nabla^s \boldsymbol{u}_h \, \mathrm{d}\Omega \\ &- \int_\Omega \nabla \cdot \boldsymbol{w}_h \, p_h \, \mathrm{d}\Omega - \sum_e \int_{\Omega^e} \nabla \cdot (2\eta \nabla^s \boldsymbol{w}_h) \, \boldsymbol{u}_s \, \mathrm{d}\Omega \\ &- \sum_e \int_{\Omega^e} \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{w}_h \, \boldsymbol{u}_s \, \mathrm{d}\Omega - \sum_e \int_{\Omega^e} \nabla \cdot \boldsymbol{w}_h \, p_s \, \mathrm{d}\Omega = \int_\Omega \boldsymbol{w}_h \boldsymbol{f} \, \mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}_h \boldsymbol{t} \, \mathrm{d}\Gamma \end{aligned} \tag{5.19}$$

$$\int_\Omega q_h \nabla \cdot \boldsymbol{u} = \sum_e \int_{\Omega^e} \nabla q_h \boldsymbol{u}_s \, \mathrm{d}\Omega \tag{5.20}$$

where Ω^e represents a single element. Note that, to obtain this expression, we integrated by parts over individual finite element domains to ensure that the equations only involve the small scale values and not their spatial derivatives. In doing so, we neglected all integrals over element boundaries.

Using ASGS stabilization, the model for the small scales is defined as

$$\boldsymbol{u}_s = \tau_1 \boldsymbol{R}^m (\boldsymbol{u}_h, p_h) = \tau_1 (\boldsymbol{f} - \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla \cdot 2\eta \nabla^s \boldsymbol{u}_h - \nabla p_h) \tag{5.21}$$

$$p_s = \tau_2 R^c (\boldsymbol{u}_h) = \tau_2 (-\nabla \cdot \boldsymbol{u}_h) \tag{5.22}$$

where $\boldsymbol{R}^m (\boldsymbol{u}_h, p_h)$ and $R^c (\boldsymbol{u}_h)$ represent the residuals of Eqs. (5.6) and (5.7) respectively, evaluated using only the large scale part of the solution $\boldsymbol{u}_h$, p_h. The stabilization parameters τ_1, τ_2 are defined in terms of a characteristic element length h as:

$$\tau_1 = \left(\frac{2\rho \, \|\boldsymbol{u}_h\|}{h} + \frac{4\eta}{h^2} \right)^{-1} \qquad \tau_2 = \eta + \frac{\rho \, \|\boldsymbol{u}_h\| \, h}{2} \tag{5.23}$$

When using OSS stabilization, only the part of the residuals $\boldsymbol{R}^m (\boldsymbol{u}_h, p_h)$ and $R^c (\boldsymbol{u}_h)$ that is orthogonal to the finite element space is used to stabilize the solution. This results in the following modified model for the small scales:

$$\boldsymbol{u}_s = \tau_1 (\boldsymbol{f} - \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla \cdot 2\eta \nabla^s \boldsymbol{u}_h - \nabla p_h - \boldsymbol{\Pi}^m) \tag{5.24}$$

$$p_s = \tau_2 (-\nabla \cdot \boldsymbol{u}_h - \Pi^c) \tag{5.25}$$

where $\boldsymbol{\Pi}^m (\boldsymbol{u}_h, p_h)$ and $\Pi^c (\boldsymbol{u}_h)$ are the L_2 projections of the residuals onto the finite element space, that is, the solution of the auxiliary projection problem

$$\int_\Omega \boldsymbol{w}_h \boldsymbol{\Pi}^m \, \mathrm{d}\Omega = \int_\Omega \boldsymbol{w}_h \, (\boldsymbol{f} - \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla \cdot 2\eta \nabla^s \boldsymbol{u}_h - \nabla p_h) \, \mathrm{d}\Omega \tag{5.26}$$

$$\int_\Omega q_h \Pi^c \, \mathrm{d}\Omega = - \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h \, \mathrm{d}\Omega \tag{5.27}$$

It is worth mentioning that the viscous term $\nabla \cdot 2\eta\nabla^s \boldsymbol{u}_h$ in Eqs. (5.21) and (5.24), as well as in the momentum projection in Eq. (5.26), cannot be evaluated using linear finite elements, as is the case of the formulation used in the present work. This is due to the fact that second order derivatives of velocity shape functions are identically zero within the elements. Therefore, this term will be neglected in the following.

Introducing the OSS small scale model into the variational form of Eqs. (5.19) and (5.20), the following stabilized formulation is obtained:

$$\begin{aligned}\int_\Omega \rho \boldsymbol{w}_h\, \boldsymbol{u}_h \nabla \boldsymbol{u}_h \,\mathrm{d}\Omega + \int_\Omega 2\eta \nabla^s \boldsymbol{w}_h : \nabla^s \boldsymbol{u}_h \,\mathrm{d}\Omega - \int_\Omega \nabla \cdot \boldsymbol{w}_h\, p_h \,\mathrm{d}\Omega \\ + \sum_e \int_{\Omega_e} \rho \boldsymbol{u}_h \nabla \boldsymbol{w}_h \tau_1 \left(\rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h + \nabla p_h\right) \mathrm{d}\Omega - \sum_e \int_{\Omega_e} \nabla \cdot \boldsymbol{w}_h \tau_2 \left(-\nabla \cdot \boldsymbol{u}_h - \Pi^c\right) \mathrm{d}\Omega = \\ \int_\Omega \boldsymbol{w}_h \boldsymbol{f} \,\mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{w}_h \boldsymbol{t} \,\mathrm{d}\Gamma + \sum_e \int_{\Omega_e} \rho \boldsymbol{u}_h \nabla \boldsymbol{w}_h \tau_1 \left(\boldsymbol{f} - \boldsymbol{\Pi}^m\right) \mathrm{d}\Omega\end{aligned} \tag{5.28}$$

$$\int_\Omega q_h \nabla \cdot \boldsymbol{u} + \sum_e \int_{\Omega_e} \nabla q_h \tau_1 \left(\rho \boldsymbol{u}_h \nabla \boldsymbol{u}_h + \nabla p_h\right) \mathrm{d}\Omega = \sum_e \int_{\Omega_e} \nabla q_h \tau_1 \left(\boldsymbol{f} - \boldsymbol{\Pi}^m\right) \mathrm{d}\Omega \tag{5.29}$$

Analogously, the ASGS stabilized formulation can be recovered by dropping all terms involving $\boldsymbol{\Pi}^m$ and Π^c in Eqs. (5.28) and (5.29).

5.4.3 Matrix formulation

At this point we can use the standard linear finite element functions to interpolate the large scale velocity and pressure solutions using nodal shape functions N_a:

$$\boldsymbol{u}_h \approx \sum_a^{n_n} \boldsymbol{N}_a \boldsymbol{u}_a \qquad p_h \approx \sum_a^{n_n} N_a p_a \tag{5.30}$$

where a is the node index and n_n the total number of nodes, N_a represents the standard finite element functions for scalar variables and $\boldsymbol{N}_a$ its matrix equivalent for vectorial quantities.

Introducing the interpolation of Eq. (5.30) into Eqs. (5.28) and (5.29) and successively using the shape functions of each node as test functions $\boldsymbol{w}_h$, p_h, we obtain the following system of equations

$$\begin{bmatrix} \boldsymbol{C} + \boldsymbol{K} + \boldsymbol{S}_K & \boldsymbol{G} + \boldsymbol{S}_G \\ \boldsymbol{D} + \boldsymbol{S}_D & \boldsymbol{L} \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} + \boldsymbol{S}_M \\ \boldsymbol{S}_C \end{bmatrix} \tag{5.31}$$

where $\boldsymbol{U}$ and $\boldsymbol{P}$ represent the vectors of nodal values for velocity and pressure, respectively. The blocks that appear in the system matrix and the right hand side vector of

Eq. (5.31) are obtained from the finite element assembly of the different integrals that appeared in the stabilized equations. If a and b represent node indices, the Galerkin terms in Eqs. (5.28) and (5.29) give rise to the following local matrices:

$$\begin{aligned}
\boldsymbol{C}^e_{ab} &= \int_{\Omega_e} \boldsymbol{N}_a^T \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{N}_b \,\mathrm{d}\Omega \\
\boldsymbol{K}^e_{ab} &= \int_{\Omega_e} 2\eta\left(\dot{\gamma}\right) \nabla \boldsymbol{N}_a^T \nabla^s \boldsymbol{N}_b \,\mathrm{d}\Omega \\
\boldsymbol{G}^e_{ab} &= -\int_{\Omega_e} \boldsymbol{N}_a^T \nabla N_b \,\mathrm{d}\Omega \\
\boldsymbol{D}^e_{ab} &= \int_{\Omega_e} \nabla N_b^T \boldsymbol{N}_a \,\mathrm{d}\Omega = -\left(\boldsymbol{G}^e_{ba}\right)^T \\
\boldsymbol{F}^e_{a} &= \int_{\Omega_e} \boldsymbol{N}_a^T \boldsymbol{f} \,\mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{N}_a^T \boldsymbol{f} \,\mathrm{d}\Gamma
\end{aligned} \tag{5.32}$$

Analogously, the discretization of the stabilization terms allows us to write

$$\begin{aligned}
\boldsymbol{S}^e_{K\,ab} &= \int_{\Omega_e} \left(\rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{N}_a\right)^T \tau_1 \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{N}_b \,\mathrm{d}\Omega + \int_{\Omega_e} \left(\nabla \cdot \boldsymbol{N}_a\right)^T \tau_2 \nabla \cdot \boldsymbol{N}_b \,\mathrm{d}\Omega \\
\boldsymbol{S}^e_{G\,ab} &= \int_{\Omega_e} \left(\rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{N}_a\right)^T \tau_1 \nabla N_b \,\mathrm{d}\Omega \\
\boldsymbol{S}^e_{D\,ab} &= \left(\boldsymbol{S}^e_{G\,ba}\right)^T \\
\boldsymbol{L}^e_{ab} &= \int_{\Omega_e} \left(\nabla N_a\right)^T \tau_1 \nabla N_b \,\mathrm{d}\Omega \\
\boldsymbol{S}^e_{M\,ab} &= \int_{\Omega_e} \left(\rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{N}_a\right)^T \tau_1 \left(\boldsymbol{f} - \boldsymbol{\Pi}^m\right) \,\mathrm{d}\Omega - \int_{\Omega_e} \left(\nabla \cdot \boldsymbol{N}_a\right)^T \tau_2 \Pi^c \,\mathrm{d}\Omega \\
\boldsymbol{S}^e_{C\,ab} &= \int_{\Omega_e} \left(\nabla N_a\right)^T \tau_1 \left(\boldsymbol{f} - \boldsymbol{\Pi}^m\right) \,\mathrm{d}\Omega
\end{aligned} \tag{5.33}$$

The system in Eq. (5.31) contains multiple non-linear terms: the convective term is non-linear in the velocity, as are all terms involving either the apparent viscosity η, the stabilization parameters τ_1, τ_2 and, in the OSS formulation, the projections. To linearize it we use Picard iterations, evaluating all terms in the local contributions using the last known values of the variables and solving the system iteratively.

In the case of OSS stabilization, an associated problem has to be solve to calculate the projections, given by the discrete form of Eqs. (5.26) and (5.27):

$$\begin{bmatrix} \boldsymbol{M}_M & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_C \end{bmatrix} \begin{bmatrix} \boldsymbol{\Pi}_M \\ \boldsymbol{\Pi}_C \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_M \\ \boldsymbol{R}_C \end{bmatrix} \tag{5.34}$$

where $\boldsymbol{\Pi}_M$ and $\boldsymbol{\Pi}_C$ represent the vectors of nodal values for the momentum and mass projections, respectively, and the different terms in the matrix and right hand side vector

can be obtained by the finite element assembly of the following local contributions:

$$\begin{aligned}
\boldsymbol{M}^e_{M\,ab} &= \int_{\Omega_e} (\boldsymbol{N}_a)^T \boldsymbol{N}_b \,\mathrm{d}\Omega \\
\boldsymbol{M}^e_{C\,ab} &= \int_{\Omega_e} N_a \, N_b \,\mathrm{d}\Omega \\
\boldsymbol{R}^e_{M\,ab} &= \int_{\Omega_e} (\boldsymbol{N}_a)^T \left(\boldsymbol{f} - \rho \boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h - \nabla p_h\right) \mathrm{d}\Omega \\
\boldsymbol{R}^e_{C\,ab} &= \int_{\Omega_e} N_a \left(-\nabla \cdot \boldsymbol{u}_h\right) \mathrm{d}\Omega
\end{aligned} \tag{5.35}$$

Note that, as was done in previous chapters, we use the fact that the projection matrices $\boldsymbol{M}_M$ and $\boldsymbol{M}_C$ have the structure of a mass matrix to approximate them by the corresponding diagonal mass matrix. This avoids the solution of the linear system which would otherwise be required to obtain the projections.

5.5 Application to Bingham fluids

The formulation introduced in the previous section can be combined with the adaptive mesh refinement procedure by introducing the small scale model of Eq. (5.21) for ASGS simulations, or Eq. (5.24) for OSS, in the error estimator of Eq. (5.1). We have used it to simulate several classical examples of Bingham flow problems. As in the previous examples, GiD is used to generate the initial mesh. Here all examples considered are small enough to be computed in a desktop computer and, as a result, the distributed memory capabilities are not relevant. In all examples but the Poiseuille flow, local mesh improvement is used to correct the refined mesh.

5.5.1 Poiseuille flow

The first test case is a simple Poiseuille flow under an imposed pressure gradient. We define a $6 \times 1\,\mathrm{m}$ plane channel and prescribe a pressure variation $\Delta p = -2 \times 10^3\,\mathrm{Pa}$ between its extremes as shown in Fig. 5.9. A no-slip condition is imposed along the edges of the channel. The fluid density is set to $\rho = 1\,\mathrm{Kg/m^3}$ while the plastic viscosity takes a value of $\mu_p = 10\,\mathrm{Pa \cdot s}$ and the regularization coefficient is set to $m = 10^3\,\mathrm{s}$. We consider two cases: a Bingham flow with yield stress $\tau_0 = 100\,\mathrm{Pa}$ and a Newtonian case with viscosity $\mu = \mu_p$.

We start the simulation using the unstructured mesh shown in Fig. 5.10, containing 66 nodes and 92 triangular elements, which corresponds to three elements along the channel width. The problem is solved iteratively: the solution of the flow is followed by the mesh refinement algorithm, repeating the procedure until the error estimator is smaller than a fixed tolerance for every element in the mesh.

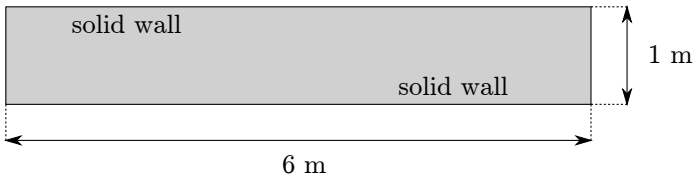


Figure 5.9: Poiseuille flow – geometry and boundary conditions.

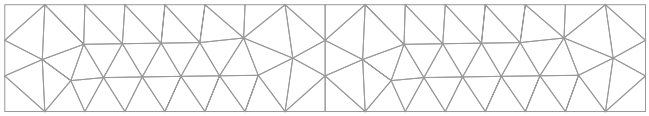


Figure 5.10: Poiseuille flow – initial mesh.

To study the sensitivity of the proposed approach with respect to the maximum admissible value for the error indicator, we simulated a series of cases with estimator tolerances in the range 10^{-3}–10^{-6}. The number of elements obtained in each case for the different stabilized formulations is shown in Fig. 5.11. The velocity profiles on the central transversal section of the domain for some values of the tolerance are shown in Fig. 5.12.

Analyzing the results, we observe a different behavior for the two cases considered. In the Newtonian case, the number of elements in the final grid increases uniformly as the tolerance is reduced. This is in agreement with the expected behavior of the estimator used: consider the that analytical solution of the Newtonian Poiseuille flow is a parabolic velocity profile given by the expression

$$u\left(y\right) = \frac{1}{2\mu}\left(-\frac{\Delta p}{\Delta x}\right) y\left(H - y\right) \qquad u_y = 0 \tag{5.36}$$

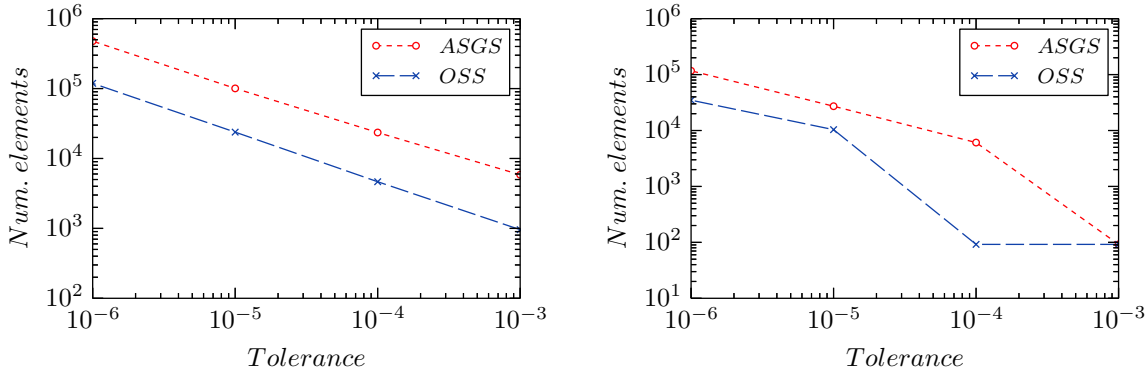
where H is the channel width and y is the vertical distance to the lower wall.

Consider the solution obtained using the ASGS stabilization. Assuming we obtained a nodally exact solution, the momentum error would be

$$R_x^m = \mu\frac{\partial^2 u}{\partial y^2} - \frac{\partial p}{\partial x} \qquad R_y^m = 0 \tag{5.37}$$

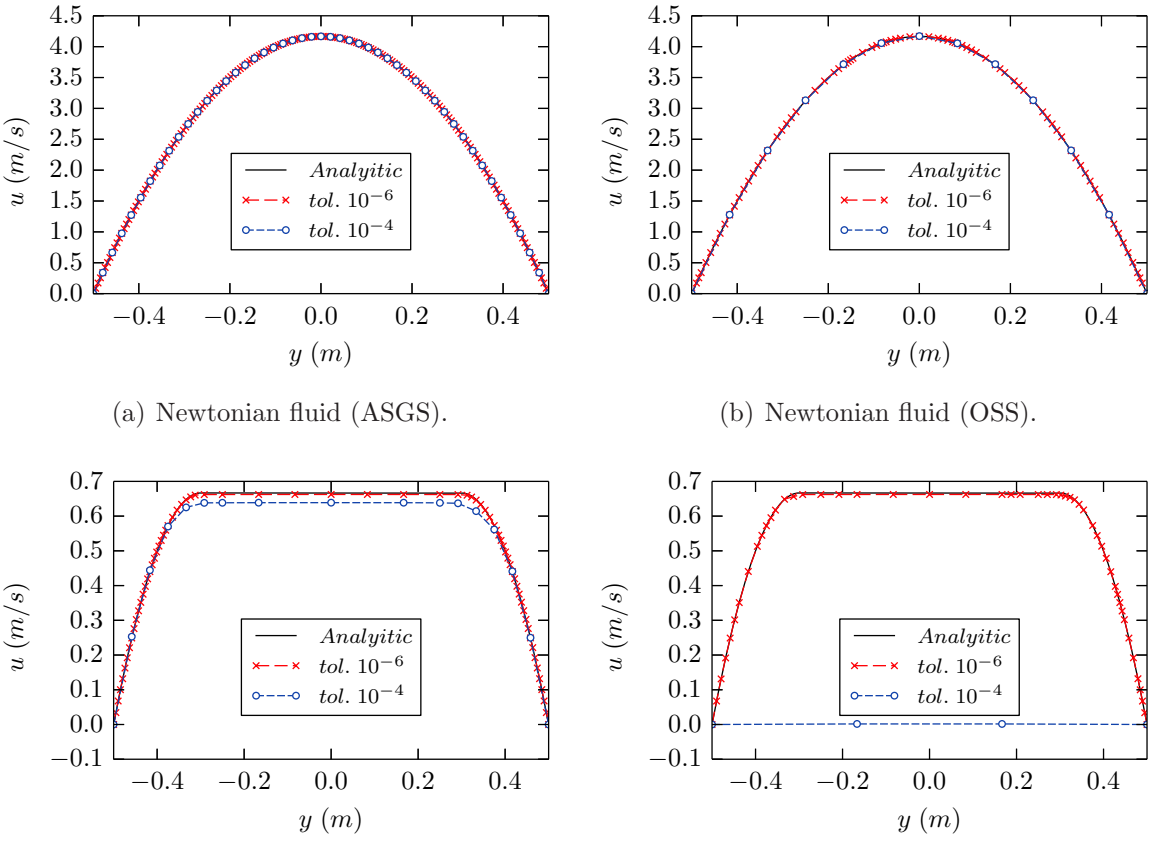
as the velocity is only different from zero in the streamwise direction and the velocity gradient is orthogonal to the velocity, cancelling the convective term. If Eq. (5.37) is evaluated using the exact solution, it can be seen to be identically zero, as $\partial^2 u/\partial y^2 = 1/\mu\Delta p/\Delta x$. However, Eq. (5.37) can never evaluate to zero numerically in our case, not even with a nodally exact solution, as second derivatives are zero when using linear finite elements.

In practice, the ASGS error estimator of Eq. 5.1 will evaluate to $\sqrt{A}\tau_1\Delta p/\Delta x$, where A is the area of the element. This explains its behavior and the large number of elements



(a) Newtonian fluid. (b) Bingham fluid.

Figure 5.11: Poiseuille flow – number of elements at the end of the simulation for different tolerances.



(a) Newtonian fluid (ASGS). (b) Newtonian fluid (OSS).

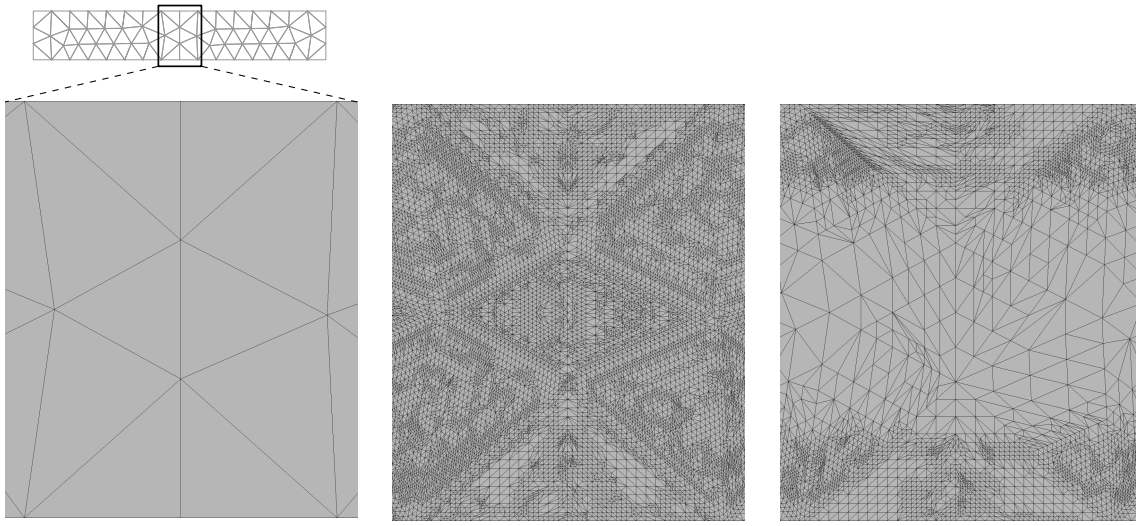
(c) Bingham fluid (ASGS). (d) Bingham fluid (OSS).

Figure 5.12: Poiseuille flow – streamwise velocity profiles.

that it introduces for all tolerances, even when the solution is already properly represented with a lower number of elements. This can be seen, for example, in Fig. 5.12(a), where the solution for the larger tolerance simulation is practically identical to the finer solution.

This effect is mitigated by the use of OSS stabilization and the corresponding error estimator which, as can be seen in Fig. 5.11(a), results in roughly five times less elements than the ASGS case for a given tolerance. When using OSS, we are missing the second derivatives both in the model for the small scales, given by Eq. (5.24), and in the calculation of the nodal projections in Eq. (5.26). As the second derivatives and the projection terms have opposite signs in Eq. (5.24), the total error in the estimation of the momentum residual is reduced.

As can be seen in Fig. 5.11, the total number of elements at the end of the simulation is larger for the Newtonian flow in all cases. This is due to the spatial distribution of the refinement, as can be observed in Fig. 5.13, and is related to the shape of the solution. For the Newtonian case, the parabolic velocity profile of the solution results in the error estimate of Eq. (5.37), which is independent of the position. As a result, the entire domain is refined homogeneously.



(a) Initial mesh. (b) Newtonian fluid. (c) Bingham fluid.

Figure 5.13: Poiseuille flow – detail of the final meshes for the OSS case with tolerance 10^{-6}.

On the other hand, in the Bingham case the profile has an unyielded central area that moves as a block and two yielded regions close to the wall with a parabolic velocity profile. The central area, with constant velocity, can be solved with practically no error with a coarse mesh size, and most of the new elements are placed in the yielded regions. As the refinement is more localized, the overall number of elements is smaller.

As a final remark, we can observe that, for small tolerances, the refinement fails to start if the mesh is too coarse to properly represent the flow. This can be seen in in Fig. 5.11(b) for the ASGS simulation with a tolerance of 10^{-3} and for the OSS cases with tolerances 10^{-3} and 10^{-4}, where no new elements are added. These cases produce a solution where the yielded regions do not develop and the velocity is practically zero everywhere.

5.5.2 Plane extrusion

We simulated the plane extrusion of a Bingham fluid through a die with a 3 to 1 reduction of the cross-section. This problem, presented in [99], was also solved in [67], using a fixed fluid mesh and an ASGS-based solver similar to that of Section 5.4, and in [80], using OSS stabilization.

As can be seen in Fig. 5.14, we use symmetry to simulate only one half of the domain. The walls are assumed to be smooth, and only the wall-normal component of velocity is restricted. The flow is driven by a ram pressure applied on the left side of the domain, which introduces a pressure gradient. The fluid parameters are reported in Table 5.3.

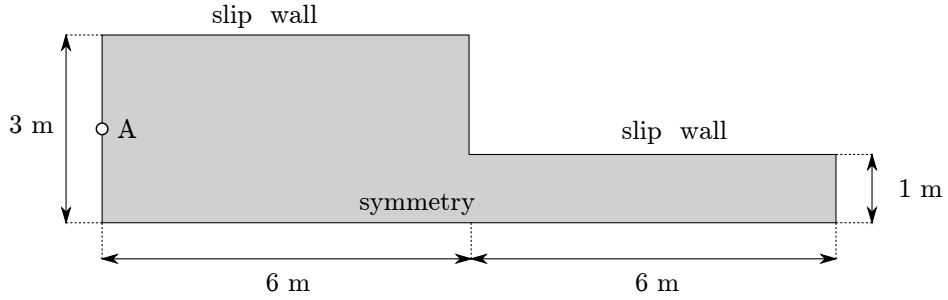


Figure 5.14: Plane extrusion – geometry and boundary conditions.

Parameter	Value
Fluid density	$\rho = 100\,\mathrm{Kg/m^3}$
Yield stress	$\tau_0 = 1000\,\mathrm{Pa}$
Fluid viscosity	$\mu_p = 10^{-6}\,\mathrm{Pa}\cdot\mathrm{s}$
Regularization coefficient	$m = 1000\,\mathrm{s}$

Table 5.3: Plane extrusion – flow parameters for the problem.

As discussed in [99], using the present settings, with smooth walls and a very small plastic viscosity, the problem is analogue to a perfect plasticity problem. An exact solution for the plasticity problem, obtained using slip line theory, is reported by Lubliner in [72]. This solution predicts the formation of slip lines once the applied pressure reaches

$$p = \frac{4}{3}\left(1 + \frac{\pi}{2}\right)\tau_y \approx 3427\,\text{Pa} \tag{5.38}$$

An increasing normal pressure is applied on the left end in steps of 2 Pa, starting from 0 to a maximum value of $p_{max} = 5000\,\text{Pa}$. After each step, the mesh refinement algorithm is used to improve mesh resolution, with a tolerance of 10^{-6} for the error indicator. In this case we applied additional controls to prevent an excessive refinement in specific zones. The minimum allowed area for refined elements is set to $10^{-4}\,\text{m}^2$. The domain is initially discretized with an unstructured mesh composed of 102 nodes and 152 linear triangles.

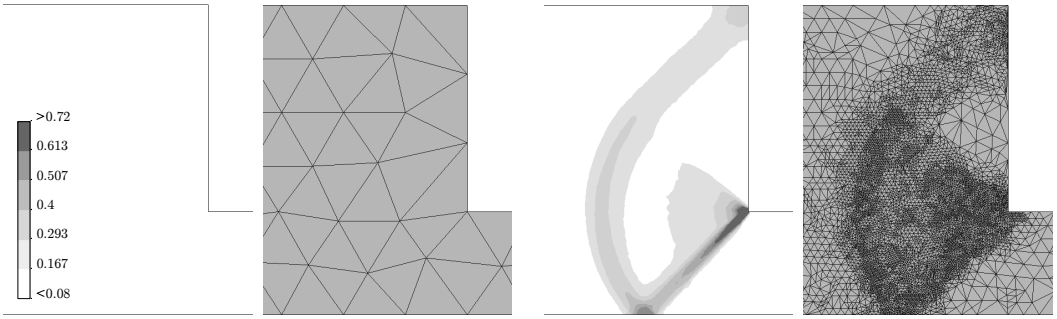
The evolution of the strain rate and the refined mesh for different values of ram pressure is shown in Fig. 5.15 for the ASGS formulation and in Fig. 5.16 for the OSS method. Both simulations show the same qualitative behavior: as the ram pressure increases, a yielded region characterized by high strain rates develops, matching the slip line mechanism. The finite element mesh is refined accordingly, following the distribution of high strain rates. The ASGS solution seems to be slightly advanced in this case, producing larger strain rates for a given ram pressure.

The evolution of mesh during the simulation is shown in Fig. 5.17. The number of elements required to solve the problem remains relatively constant for low values of the ram pressure until the yielded zone starts to develop. At this point, the number of elements increases rapidly as the material starts to flow. The new elements are concentrated at the fluidified regions, as can be observed in Figs. 5.15 and 5.16, and the refinement process continues as the yielded region expands and moves downstream. The mesh at the end of the simulation, corresponding to an external pressure of 5000 Pa, contains 15173 nodes and 30166 elements in the ASGS case and 14466 nodes and 28756 elements in the OSS case.

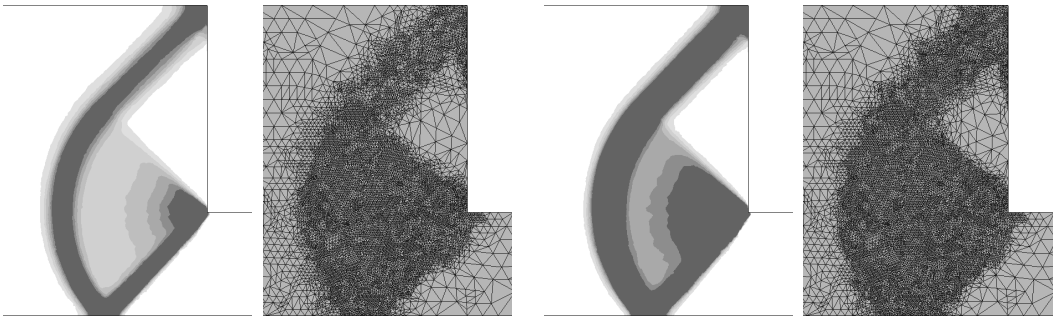
The velocity of the fluid on the left boundary (measured on point A in Fig. 5.14) is related to the ram pressure in Fig. 5.18. We can observe that velocity is very low until the pressure reaches about 3460 Pa, when the material starts to flow, accelerating rapidly. Again, the change in behavior corresponds to the formation of a yielded zone just before the extrusion section. This is found to be in agreement with the expected behavior, although the material starts flowing at slightly higher pressures than expected from perfect plasticity theory, which is indicated in Fig. 5.18 using a dotted line.

We can also observe in Fig. 5.18 that the OSS simulation predicts a slightly later plastification, corresponding to higher ram pressures, when compared to the ASGS method, while the latter is closer to the plasticity theory result. A possible explanation for this could be that the ASGS estimator produces a larger number of elements overall, as can be seen in Fig. 5.17, which provides a slightly better accuracy in the solution.

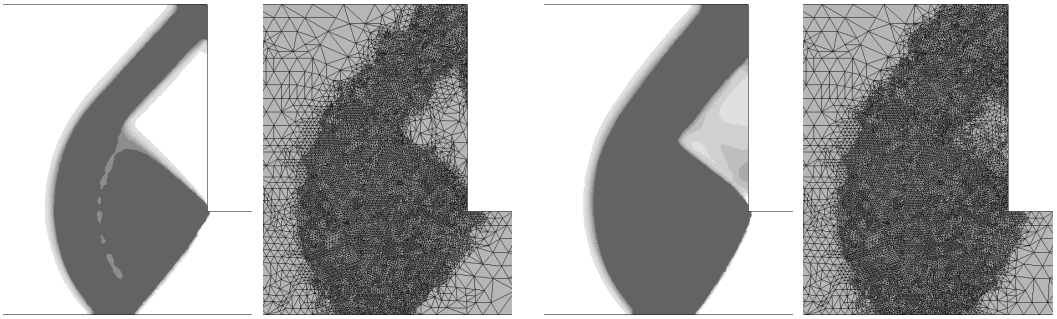
As an attempt to quantify the effectiveness of the refinement procedure, we also simulated the problem using OSS and a uniform mesh containing 28072 elements, a similar amount as in the finest mesh obtained during the refinement, to use as a reference



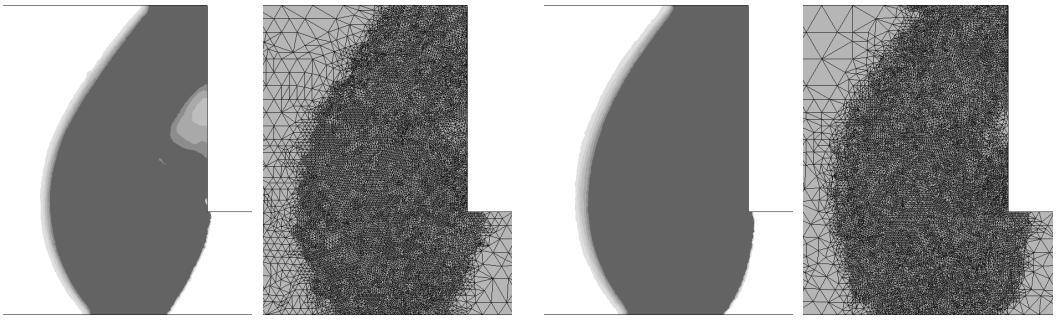
(a) 0 Pa. (b) 3448 Pa.



(c) 3454 Pa. (d) 3462 Pa.

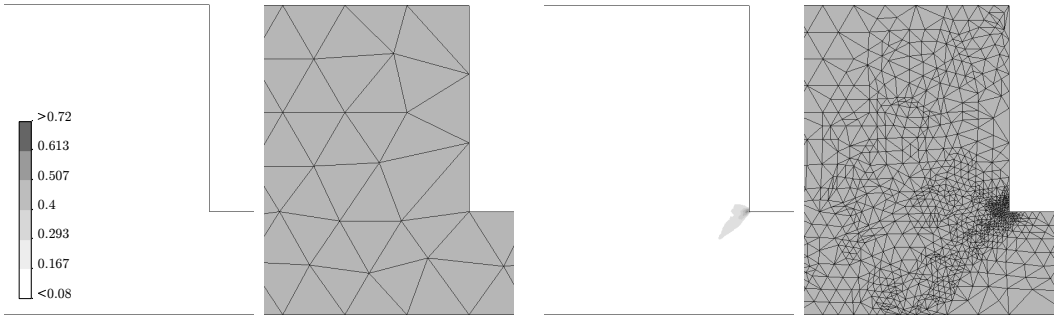


(e) 3500 Pa. (f) 3750 Pa.



(g) 4000 Pa. (h) 4500 Pa.

Figure 5.15: Plane extrusion (ASGS) – evolution of strain rate (left) and mesh (right).



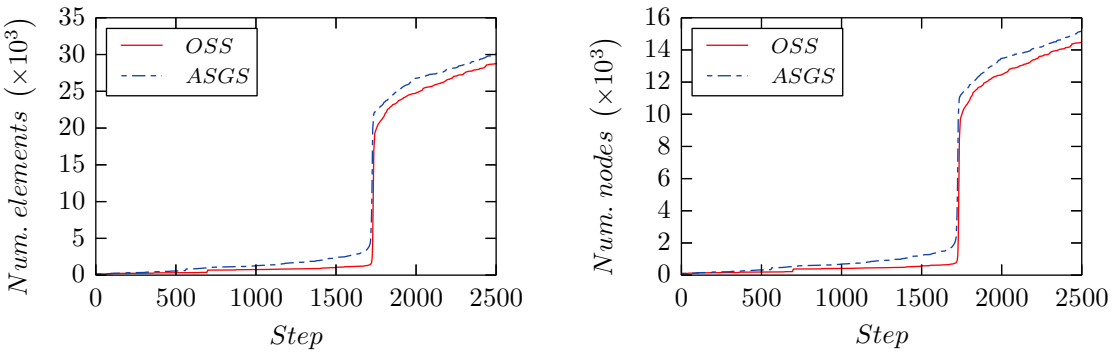
(a) 0 Pa. (b) 3448 Pa.

(c) 3454 Pa. (d) 3462 Pa.

(e) 3500 Pa. (f) 3750 Pa.

(g) 4000 Pa. (h) 4500 Pa.

Figure 5.16: Plane extrusion (OSS) – evolution of strain rate (left) and mesh (right).



(a) Number of elements. (b) Number of nodes.

Figure 5.17: Plane extrusion – evolution of the simulation mesh.

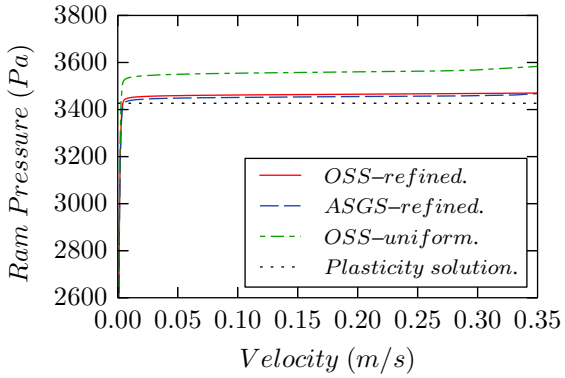


Figure 5.18: Plane extrusion – applied pressure vs. inlet velocity.

solution. The velocity–pressure relation for this simulation is also plotted in Fig. 5.18 and shows a delayed formation of the yielded region, which appears at higher ram pressures than expected. While using a uniform mesh would be a very naive approach in this case, the results show that the use of a refinement procedure results in an improved solution for a given number of elements in the mesh.

5.5.3 Cavity flow

The next example we considered is the 2D cavity flow of a Bingham fluid. The problem uses the same settings as Mitsoulis and Zisis in [79]. Defining a square domain $\Omega = (0, H) \times (0, H)$, we impose a horizontal velocity $U = 1\,\mathrm{m/s}$ on the $y = H$ side and zero velocity on the remaining sides. We simulate a leaky cavity, the top left and top right corner nodes have a fixed horizontal velocity. This condition is re-imposed after each refinement step, so that the wall node immediately next to the corner always has zero

velocity, as shown in Fig. 5.19, even if this node didn't exist in previous iterations.

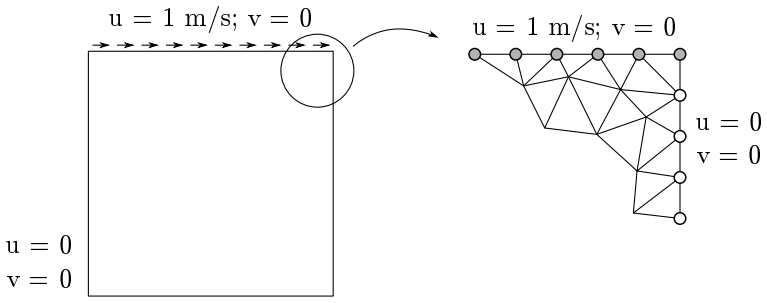


Figure 5.19: $2D$ cavity flow – boundary conditions.

The fluid density is set to $\rho = 1\,\mathrm{Kg/m^3}$ and the dynamic viscosity for the yielded region is set to $\mu_p = 1\,\mathrm{Pa}\cdot\mathrm{s}$. We will simulate multiple cases with different yield stresses to test a range of values of the Bingham number, defined as

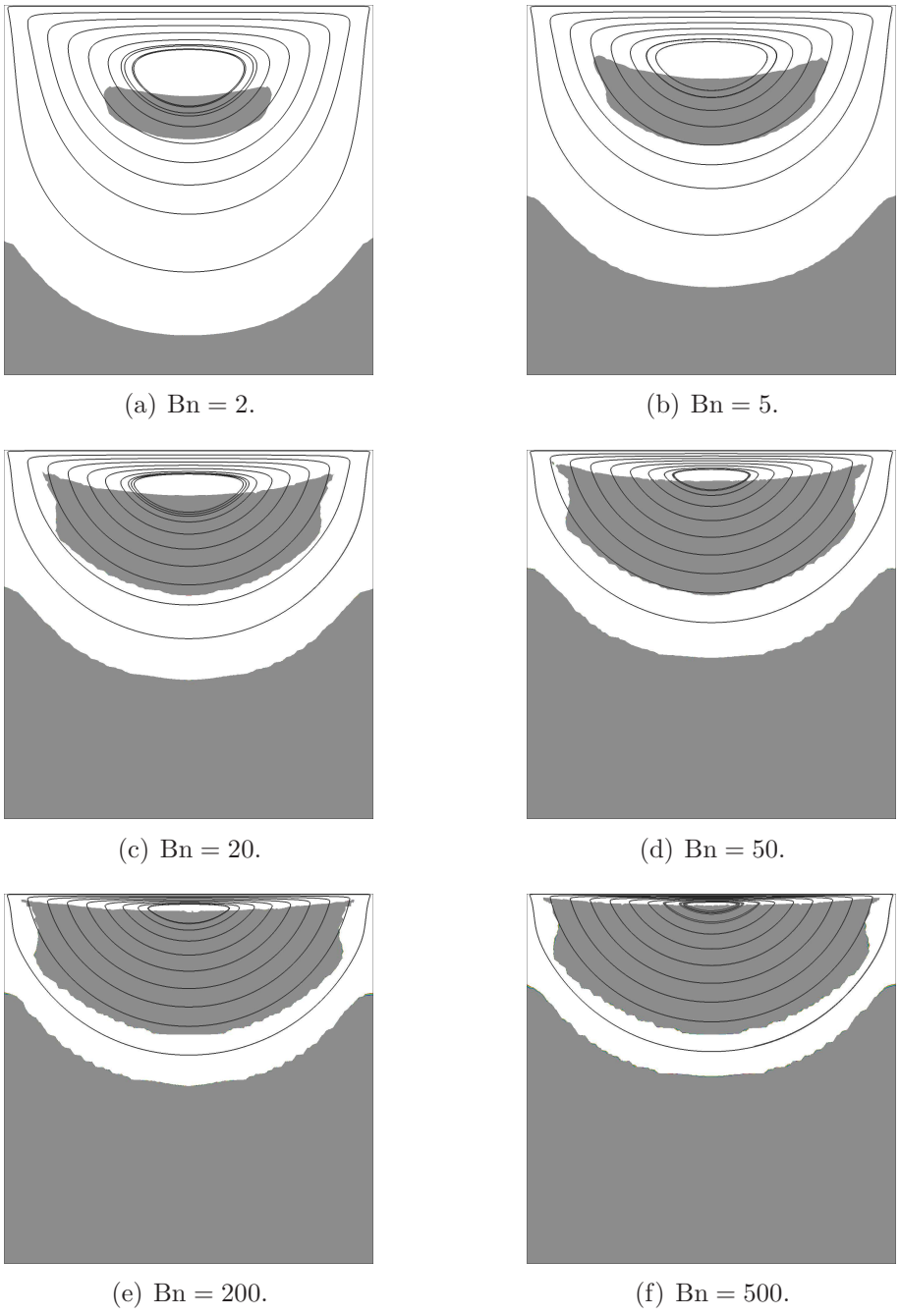
$$\mathrm{Bn} = \frac{\tau_0 H}{\mu_p U} \tag{5.39}$$

The regularization coefficient is set to $m = 300\,\mathrm{s}$. In this case, all simulations were performed using OSS stabilization. As in the previous test, we start from a relatively coarse uniform mesh composed of 2900 nodes and 5600 triangular elements and we solve the problem iteratively, with a mesh refinement phase after each solution. The refinement algorithm is set to a tolerance of 10^{-6} and to a maximum of 10 refinement steps over the same original element. This is important in this case, as a concentration of pressure can be expected to appear in the corners of the cavity and the mesh refinement could potentially continue indefinitely on these points. The final distribution of yielded and unyielded regions and the corresponding velocity streamlines are shown in Fig. 5.20 for the different simulations.

The vertical position of the vortex center for each case is compared to the results reported in [79] in Fig. 5.21. The results are in agreement with the reference, although we obtain a slightly higher position for the center in the higher Bingham numbers.

Fig. 5.22 displays the evolution of the number of elements during the simulation, which can be seen to increase quickly during the first steps, until the error estimator complies with the imposed tolerance in most of the domain.

Although the cavity flow is essentially a $2D$ problem for the range of values we are testing, we also simulated a $3D$ case in order to validate our approach for tetrahedra. The geometry of the problem is shown in Fig. 5.23 and follows the definition of Elias *et al.* in [40]. The domain is a cube of side $H = 1\,\mathrm{m}$, where velocity is fixed to $(U, 0, 0)$ on the top side. Taking Y to be the vertical axis, the velocity is set to zero on the bottom and on the sides of the cube normal to the flow. On the remaining sides, parallel to the flow on the top, only the normal (Z) component of velocity is restricted.



(a) Bn = 2. (b) Bn = 5.

(c) Bn = 20. (d) Bn = 50.

(e) Bn = 200. (f) Bn = 500.

Figure 5.20: $2D$ cavity flow – velocity streamlines and distribution of yielded (light) and unyielded (dark) regions.

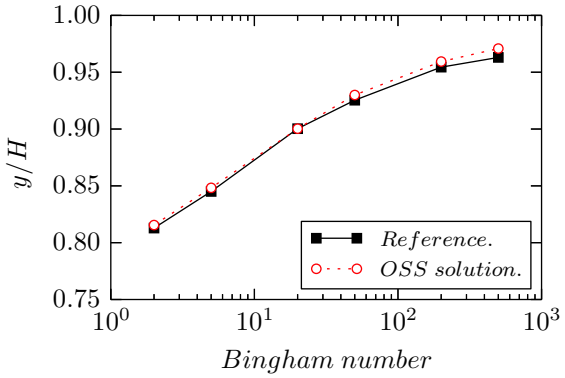
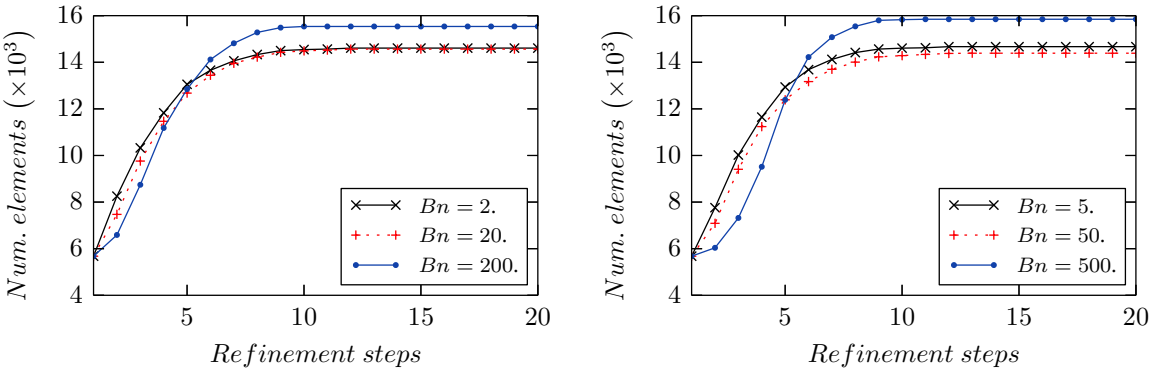


Figure 5.21: $2D$ cavity flow – vertical position of the vortex center, compared to the results of [79].



(a) Bn $= 2, 20, 200$

(b) Bn $= 5, 50, 500$

Figure 5.22: $2D$ cavity flow – evolution of the number of elements.

We solve the flow for a Reynolds number Re $= 1$ and a Bingham number Bn $= 5$. All fluid parameters are defined as in the $2D$ case, setting the top velocity to $U = 1\,m/s$ and the yield stress is $\tau_0 = 5\,\text{Pa}$. The regularization coefficient is set to $m = 1000$ s.

The flow is simulated in 10 solution steps, refining after each solution. Starting from a uniform tetrahedral mesh with 30 divisions along each edge, containing approximately 51 thousand nodes and 277 thousand elements, a final mesh with 113 thousand nodes and 612 thousand elements is obtained. The final distribution of yielded and unyielded regions and velocity streamlines is shown in Fig. 5.24. The vortex center in this case is placed at a vertical position $y/H = 0.848$, in agreement with the $2D$ results shown in Fig. 5.21.

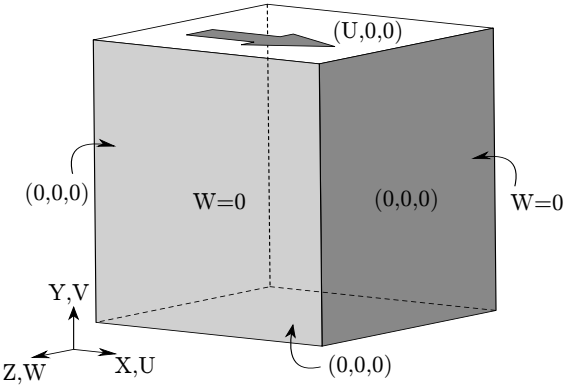


Figure 5.23: $3D$ cavity flow – geometry and velocity boundary conditions.

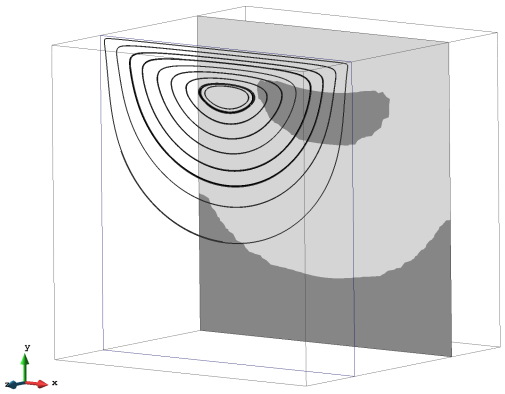


Figure 5.24: $3D$ cavity flow – velocity streamlines and distribution of yielded (light) and unyielded (dark) regions.

5.5.4 Flow through a sudden expansion

As a final test case we have simulated the $3D$ flow through a square sudden expansion. This problem was studied in [4, 18] for Herschel-Bulkley fluids and represents a three-dimensional version of the more common planar or axisymmetric expansions (see for example [5, 78, 111]). The cross section of the problem is shown in Fig. 5.25. We modeled the flow through expansions with 1 to 2 and 1 to 4 width ratios, which correspond to $W/H = 2$ and $W/H = 4$ using the notation of Fig. 5.25.

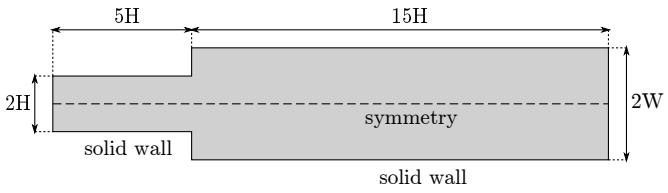


Figure 5.25: Sudden expansion – geometry.

Using symmetry, only one fourth of the domain is simulated, resulting in the calculation geometries shown in Fig. 5.26. No-slip boundary conditions are used to model the solid walls, while a no-penetration conditions is set for the symmetry planes. The flow is driven by a pressure applied on the narrow side of the domain.

The problem is solved for the case where both the Reynolds and Bingham numbers equal to one, calculated using H as the reference length and a reference velocity U_0 defined in [18] as

$$U_0 = \frac{1}{\mu_p}\left(H\left|\frac{\Delta p}{\Delta x}\right| - \tau_0\right)H \tag{5.40}$$

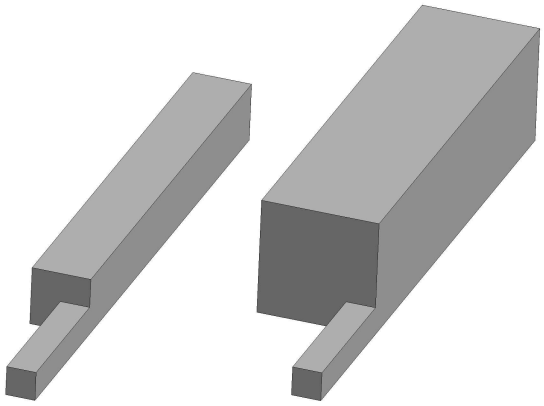


Figure 5.26: Sudden expansion – simulation domains.

where $\Delta p/\Delta x$ is the pressure gradient that drives the flow.

We apply the external pressure in 10 incremental load steps, refining after each iteration. Once the loading process is finished, we simulate 5 extra steps under full load to ensure that the final solution does not require additional refinement. The distribution of yielded and unyielded regions on different sections can be observed in Fig. 5.27 for the $W/H = 2$ expansion and in Fig. 5.28 for the $W/H = 4$ case.

(a) Side view.

(b) A–A'

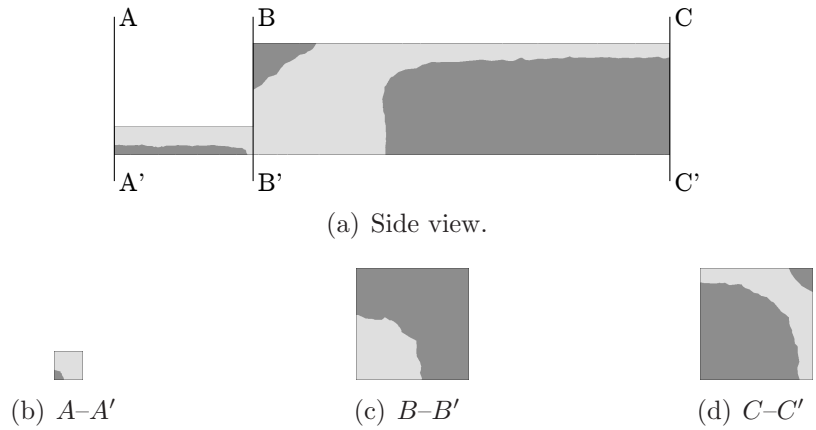
(c) B–B'

(d) C–C'

Figure 5.27: $W/H = 2$ expansion – yielded (dark) and unyielded (light) regions.

Both simulations exhibit a qualitatively similar behavior, in agreement with the results obtained in the references. Far from the expansion, we can differentiate a yielded region close to the wall, due to the shear produced by wall friction, and a central core of unyielded material. Close to the expansion, high velocity gradients develop as the flow adapts to the change in cross section and the central region is completely fluidified. Just after the expansion, on the wide side, a region of stationary unyielded material appears, unaffected by the main flow, which can be understood as equivalent to the recirculation zones for a Newtonian fluid.

As in the previous cases, the simulation begins with a uniform tetrahedral mesh, composed of approximately 5000 nodes and 22000 elements for the $W/H = 2$ case or



(a) Side view.

(b) A–A' (c) B–B' (d) C–C'

Figure 5.28: $W/H = 4$ expansion – yielded (dark) and unyielded (light) regions.

10000 nodes and 53000 elements for the $W/H = 4$ case. The evolution of the number elements during the solution is shown in Fig. 5.29. The number of elements grows as the applied pressure gradient increases and stabilizes once the loading process finishes, resulting in a final grid of 106000 nodes and 596000 elements for the $W/H = 2$ case and 123000 nodes and 694000 elements for the $W/H = 4$ expansion.

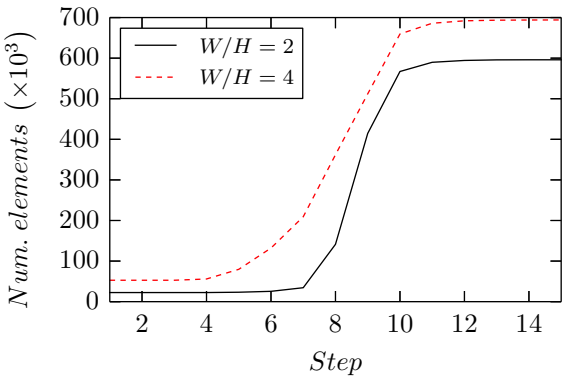
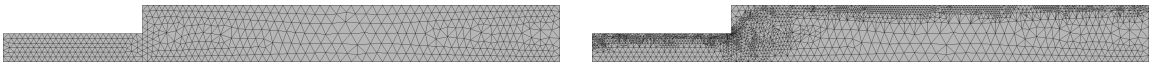


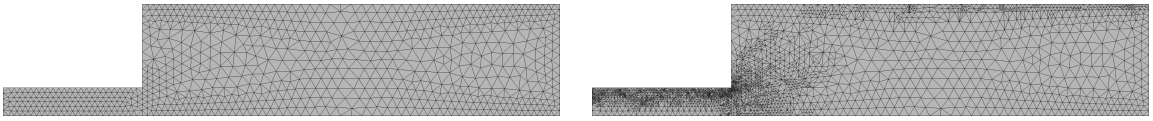
Figure 5.29: Sudden expansion – evolution of the number of elements during the simulation.

The initial and final meshes are shown in Fig. 5.30 for the $W/H = 2$ test and Fig. 5.31 for the $W/H = 4$ case. It can be observed that refined areas coincide with yielded regions, where higher velocity gradients are generally found: close to the solid walls and just after the expansion section.



(a) Initial mesh. (b) Final mesh.

Figure 5.30: $W/H = 2$ expansion – side view of the calculation meshes.



(a) Initial mesh. (b) Final mesh.

Figure 5.31: $W/H = 4$ expansion – side view of the calculation meshes.

5.6 Summary and conclusions

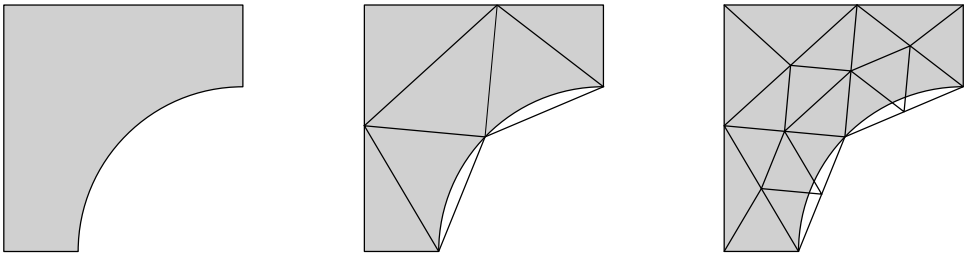
In the previous pages we have presented an adaptive mesh refinement technique that combines an *a posteriori* error estimator based on the VMS scale separation and a mesh refinement strategy based on edge division, designed and implemented to work in a distributed memory parallel environment. This approach has been applied to the simulation of turbulent and viscoplastic flows, improving the resolution of triangular meshes in $2D$ and tetrahedral meshes in $3D$. To conclude the presentation, we will give some final thoughts on the method and propose future lines of improvement.

We have shown that our approach to mesh refinement is parallel by design and its distributed memory implementation shows good parallel performance. This is achieved through the use of a very simple refinement procedure, based only on data that is local to the element. In fact, the division procedure presented in Section 5.2.3 is based only on the nodal numbering.

The drawback of this approach is that the quality of the resulting refined mesh is not taken into account when dividing elements, which can result in a sub-optimal mesh quality. A local division strategy that took into account the shape of the element and its neighbors when refining could allow us to chose a more convenient division pattern in many cases, resulting in better mesh quality. This suggests a first venue for future improvement, for example devising an improved approach that incorporates the ideas of the local mesh improvement techniques we now use as a post-process to choose an optimal refined configuration, but always keeping in mind that the parallelization of such approach would have to be carefully considered.

A second possibility for improvement of the mesh refinement strategy is the treatment of curved surfaces. While this has not been significant in the examples presented here (only the cylinder example included curved edges), a drawback of our refinement algorithm is that, when working with curved surfaces, the quality of their representation is given by the *coarsest* mesh used during the process. This issue is shown graphically

in Fig. 5.32, and would likely be problematic in cases where a precise description of the geometry is required, such as flows around wing profiles or wind turbine blades, regardless of the final mesh size. A starting point to solve this issue could be storing the original geometry using, for example, a NURBS-based description and using this information to correct the position of new nodes, placing them closer to the original surface instead of on the edge midpoint.



(a) Original domain. (b) Coarse mesh. (c) Refined mesh.

Figure 5.32: Refinement of curved edges.

The incompressible flow examples presented in Section 5.3 allow us to say that our approach correctly identifies the zones where refinement is required. However, regarding the parallel efficiency of the method, we found that dynamic load balancing is necessary to solve large scale examples efficiently, as local refinement can quickly unbalance the original partitioning of the model. Additionally, mesh coarsening would be a welcome addition to solve dynamic problems as, in our experience, the regions with larger error tend to follow moving vortices and this currently results in a complete refinement of turbulent wakes.

Although our adaptive refinement strategy was originally designed to simulate turbulent flows, we found that it is also suited to the simulation of viscoplastic fluids. The fact that many problems of interest in this field are generally static and have sharp interfaces between yielded and unyielded regions plays to the strengths of our approach, as coarsening is rarely required. We found that our method was able to identify the regions where fluidification occurs even when starting from a uniform initial mesh. While the capabilities of the method have been demonstrated by the test cases presented in Section 5.5, there are some questions that could be addressed in the future. For example, how should the tolerance for the error estimator be set? Is it problem dependent? In our experience, the solution is sensitive to the tolerance set for the error estimator: a small tolerance produces very fine meshes, while a large tolerance can even prevent the refinement from starting if the original mesh is too coarse. This was shown to be the case in some of the Poiseuille flow examples presented in Section 5.5.1.

Going back to turbulent flow, there is one interesting question regarding the chosen estimator. According to the arguments presented in Chapter 2, the small scale part of the solution can be understood as the basic ingredient of a VMS-based turbulence modeling.

In this sense, imposing a restriction on its magnitude seems to be contradictory with our goal in previous chapters, which was to perform a LES-like simulation thanks to the contribution of the small scale terms, and seems to push the mesh size towards DNS simulations, which we discarded as prohibitively expensive for problems of engineering interest.

Chapter 6

Conclusions

6.1 Summary and main results

In the present work we have explored the capabilities of stabilized finite element formulations for the solution of turbulent flow problems. To achieve this goal, we have studied two families of methods: VMS and FIC formulations, applying them to the solution of different benchmark problems. In addition, we have studied numerical techniques relevant to the solution of large complex problems, in particular the parallel implementation of the code and the use of adaptive mesh refinement to improve the quality of the simulation mesh.

As a general observation for the different stabilized formulations considered, we noticed a notable difference between using linear tetrahedral or hexahedral finite elements in terms of the quality of the solution. While this was not unexpected, we were able to quantify the difference for the turbulent channel flow, where we see that, in general, we need an order of magnitude more tetrahedral elements to achieve the same quality in the solution as for hexahedra.

For VMS methods we have studied the behavior of classical formulations on the channel flow problem, in terms of velocity averages and variances and of the turbulence kinetic energy balance. Additionally, we have presented a new model for the pressure subscale based on the use of an approximate small scale space, which has been shown to provide a promising improvement in the quality of the solution for the channel flow test when compared to the usual approach. However, we want to remark that we consider this only a starting point, since the effect of the pressure small scale on the solution has been shown to be problem-dependent in the literature.

Regarding FIC formulations, introduced in Chapter 3, we have presented a new method that includes a diffusive term based on imposing the FIC balance in the direction of the gradients of each component of velocity, in addition to the usual streamline diffusion. Although this term is derived from the FIC balance (as opposed to a tur-

bulence modeling argument), we have shown its presence allows us to obtain a more accurate solution in the turbulent channel flow benchmark. We have also applied the resulting formulation to more complex geometries, and in particular to the wind flow around a parabolic solar collector. While the simulation performed represents only a first approximation to the problem, as more tests and longer simulation times would be required to obtain a more reliable solution, they are encouraging in terms of the performance of the method.

In Chapter 4, the parallel performance of the solver was studied. We presented the measured calculation times in large simulations, measuring the parallel scalability when using up to 3072 processors. We observed that the solution time is dominated by the linear system solver, which is significantly more expensive than the finite element assembly procedure. While the parallel solution of linear systems is beyond the scope of the present work, we have concentrated our efforts in obtaining a good parallel performance in the parts of the solution where we have direct control, relying on external libraries for system solution.

Finally, in Chapter 5 we investigated the possibility of using adaptive mesh refinement to optimize the mesh during the simulation, which we applied both to turbulent and non-Newtonian flow examples. While the results show the potential of the method, we found that the applicability of our approach to turbulent flow problems is limited by the lack of a mesh coarsening procedure. Since turbulence is a highly dynamic problem, regions that required a finer resolution at a given step might be solvable with a much coarser grid at a later time but, without mesh coarsening, we are stuck with the finer resolution for all subsequent steps, greatly increasing the total number of elements. A second issue that was detected, in the case of distributed memory simulations, is that dynamic load balancing is crucial to maintain a good parallel performance during the entire solution. In spite of these limitations, the approach was found to be well-suited to laminar non-Newtonian flows, where finer resolutions are typically required along localized high-shear regions.

6.2 Research outcomes

Parts of the work presented in this monograph have been published in scientific journals. In particular, some of the work related to the parallel implementation of the solver was included in

- P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela-Dalmau, E. Juanpere, S. R. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to HPC environments. *Computers & Fluids*, 80:301–309, 2013.

While the work in adaptive mesh refinement has been used to prepare the following publications

- R. Rossi, J. Cotela-Dalmau, N. M. Lafontaine, P. Dadvand, and S. R. Idelsohn. Parallel adaptive mesh refinement for incompressible flow problems. *Computers & Fluids*, 80:324–355, 2013.
- J. Cotela-Dalmau, R. Rossi, and A. Larese. Simulation of two and three-dimensional viscoplastic flows using adaptive mesh refinement. Submitted to *International Journal of Non-Newtonian Fluid Mechanics*, 2015.

Articles related to the work contained in Chapters 2 and 3 are planned.

Finally, we want to note that an implementation of all formulations and techniques presented in this document is available within Kratos Multiphysics. In particular, the VMS methods introduced in Chapter 2 currently constitute the basis of the CFD module of Kratos Multiphyisics, both in the monolithic form which has been presented and used in the present work and as a fractional step implementation of the Q-OSS formulation. This solver has been used by other researchers, both in collaboration with the authors and in other groups, and in the elaboration of multiple Master theses, where students have used it for example to perform studies of the wind flow around bridge sections or to build an ALE-Chimera solver for CFD problems with moving parts.

6.3 Future lines of research

The results obtained in the course of the present work suggest several possibilities for future investigation and improvement.

In terms of VMS formulations we are of the opinion that, while using scale separation and mesh projection is a valid option introduce a LES-like separation between the resolved and unresolved part of the solution, there is still work to do in understanding the practical behavior of the resulting method as a turbulence model. In particular, the behavior of the pressure small scale and its impact on the solution is still poorly understood. While we were able to compare the results obtained using different formulations, it is not always clear why a particular method provides a more accurate solution than the other. Additionally, for the proposed pressure subscale model, although the results in our test are encouraging, there is still work to do in understanding precisely why we obtained a better approximation and to verify if this result would hold for other problems.

A similar argument could be made in terms of the new FIC formulation: while the resulting formulation results in an improved accuracy, we did not provide a justification of why this happens. In this particular case, it is interesting to note that the gradient diffusion terms that constitute the main difference to the classical FIC approach have a similar structure to the family of LES methods known as gradient models such as the Clark model [23] or Modulated Gradient Diffusion [71]. Exploring its relation to such models could allow us to obtain a better understanding of the method in terms of its behavior in turbulent problems and motivate future improvements.

Regarding the development of parallel capabilities, the immediate work will be centered on continuing the integration of the AMGCL library as it is developed. Another possibility that will be explored is the addition of a hybrid implementation, combining shared and distributed memory capabilities. It could also be worthwhile to work on improving the partitioning procedure, using a more efficient coloring procedure and, most importantly, adding dynamic load balancing procedures to adjust the work load in each processor if the simulation mesh is changed. This would be a welcome addition in particular for the adaptive mesh refinement procedure presented in Chapter 5, as it would extend the range of applicability of the method.

Bibliography

[1] M. Ainsworth and J. T. Oden. A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142(12):1 – 88, 1997.

[2] M. Ainsworth and J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, Ltd, 2000.

[3] I. Akkerman, Y. Bazilevs, V. Calo, T. Hughes, and S. Hulshoff. The role of continuity in residual-based variational multiscale modeling of turbulence. *Computational Mechanics*, 41:371–378, 2008.

[4] A. N. Alexandrou, T. M. McGilvreay, and G. Burgos. Steady Herschel-Bulkley fluid flow in three-dimensional expansions. *Journal of Non-Newtonian Fluid Mechanics*, 100(1-3):77 – 96, 2001.

[5] N. Alleborn, K. Nandakumar, H. Raszillier, and F. Durst. Further contributions on the two-dimensional flow in a sudden expansion. *Journal of Fluid Mechanics*, 330:169–188, 1 1997.

[6] M. Andre, M. Mier-Torrecilla, and R. Wüchner. Numerical simulation of wind loads on a parabolic trough solar collector using lattice boltzmann and finite element methods. *Journal of Wind Engineering and Industrial Aerodynamics*, 146:185 – 194, 2015.

[7] M. Ávila. *Nonlinear subgrid finite element models for low Mach number flows coupled with radiative heat transfer*. PhD thesis, Universitat Politècnica de Catalunya, 2012.

[8] M. Ávila, J. Príncipe, and R. Codina. A finite element dynamical nonlinear subscale approximation for the low Mach number flow equations. *Journal of Computational Physics*, 230(22):7988 – 8009, 2011.

[9] Y. Bazilevs, V. Calo, J. Cottrell, T. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 197(1-4):173 – 201, 2007.

[10] Y. Bazilevs, C. Michler, V. Calo, and T. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(13-16):780–790, 2010. Turbulence Modeling for Large Eddy Simulations.

[11] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196(49-52):4853 – 4862, 2007.

[12] P. Beaudan and P. Moin. Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number. Technical Report TF-62, Department of Mechanical Engineering, Stanford University, December 1994.

[13] N. Bernabeu, P. Saramito, and C. Smutek. Numerical modeling of non-Newtonian viscoplastic flows: Part II. Viscoplastic fluids and general tridimensional topographies. *International Journal of Numerical Analysis and Modeling*, 11(1):213–228, Jan. 2014. Dedicated to Professor Francisco J. Lisbona on the occasion of his 65th Birthday.

[14] E. C. Bingham. *Fluidity and Plasticity*. McGraw-Hill, 1922.

[15] P. B. Bochev, M. D. Gunzburger, and R. B. Lehoucq. On stabilized finite element methods for the Stokes problem in the small time step limit. *International Journal for Numerical Methods in Fluids*, 53(4):573–597, 2007.

[16] J. P. Boris, F. F. Grinstein, E. S. Oran, and R. L. Kolbe. New insights into large eddy simulation. *Fluid Dynamics Research*, 10(4-6):199, 1992.

[17] E. Bou-Zeid, C. Meneveau, and M. Parlange. A scale-dependent Lagrangian dynamic model for large eddy simulation of complex turbulent flows. *Physics of Fluids (1994-present)*, 17(2):025105, 2005.

[18] G. R. Burgos and A. N. Alexandrou. Flow development of Herschel-Bulkley fluids in a sudden three-dimensional square expansion. *Journal of Rheology*, 43(3):485–498, 1999.

[19] T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. Technical Report STAN-CS-79-773, Department of Computer Science, Stanford University, 1979.

[20] G. Chiandussi, G. Bugeda, and E. Oñate. A simple method for automatic update of finite element meshes. *Communications in Numerical Methods in Engineering*, 16(1):1–19, 2000.

[21] K.-Y. Chien. Predictions of channel and boundary-layer flows with a low-Reynolds-number turbulence model. *AIAA journal*, 20(1):33–38, 1982.

[22] CIMNE. GiD, the personal pre and post processor. `www.gidhome.com`. Retrieved on August 2015.

[23] R. A. Clark, J. H. Ferziger, and W. C. Reynolds. Evaluation of subgrid-scale models using an accurately simulated turbulent flow. *Journal of Fluid Mechanics*, 91:1–16, 3 1979.

[24] R. Codina. A discontinuity-capturing crosswind-dissipation for the finite element solution of the convection-diffusion equation. *Computer Methods in Applied Mechanics and Engineering*, 110(3-4):325 – 342, 1993.

[25] R. Codina. Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 190(13-14):1579–1599, 2000.

[26] R. Codina. Pressure stability in fractional step finite element methods for incompressible flows. *Journal of Computational Physics*, 170(1):112–140, 2001.

[27] R. Codina. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 190(20-21):268–2706, 2001.

[28] R. Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191(39-40):4295–4321, 2002.

[29] R. Codina, J. Principe, and M. Ávila. Finite element approximation of turbulent thermally coupled incompressible flows with numerical sub-grid scale modeling. *International Journal for Numerical Methods for Heat & Fluid Flow*, 20:492–516, 2010.

[30] R. Codina, J. Príncipe, O. Guasch, and S. Badia. Time dependent subscales in the stabilized finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 196(21-24):2413–2430, 2007.

[31] A. Coll Sans. *Robust volume mesh generation for non-watertight geometries*. PhD thesis, Universitat Politècnica de Catalunya, 2014.

[32] O. Colomés, S. Badia, R. Codina, and J. Príncipe. Assessment of variational multiscale models for the large eddy simulation of turbulent incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 285(0):32 – 63, 2015.

[33] P. Dadvand. *A framework for developing finite element codes for multi-disciplinary applications.* PhD thesis, Universitat Politècnica de Catalunya, 2007.

[34] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela-Dalmau, E. Juanpere, S. R. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to HPC environments. *Computers & Fluids*, 80:301–309, 2013.

[35] P. Dadvand, R. Rossi, and E. Oñate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of Computational Methods in Engineering*, 17:253–297, 2010.

[36] J. P. D'Amato and M. Vénere. A CPU–GPU framework for optimizing the quality of large meshes. *Journal of Parallel and Distributed Computing*, 73(8):1127 – 1134, 2013.

[37] D. Demidov. AMGCL: a C++ library for solution of large sparse linear systems with algebraic multigrid method. `https://github.com/ddemidov/amgcl`. Retrieved on October 2015.

[38] D. Demidov and R. Rossi. Subdomain deflation and algebraic multigrid: combining multiscale with multilevel. Unpublished manuscript.

[39] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems.* John Wiley & Sons, Ltd, 2003.

[40] R. Elias, M. Martins, and A. Coutinho. Parallel edge-based solution of viscoplastic flows with the SUPG/PSPG formulation. *Computational Mechanics*, 38(4-5):365–381, 2006.

[41] A. Ern and J.-L. Guermond. *Theory and practice of finite elements*, volume 159. Springer Science & Business Media, 2013.

[42] P. J. Frey and P. L. George. *Mesh Generation: Application to finite elements.* John Wiley & Sons, Ltd, 2nd edition, 2008.

[43] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. ML 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[44] V. Gravemeier. The variational multiscale method for laminar and turbulent flow. *Archives of Computational Methods in Engineering*, 13(2):249–324, 2006.

[45] V. Gravemeier, M. W. Gee, M. Kronbichler, and W. A. Wall. An algebraic variational multiscale–multigrid method for large eddy simulation of turbulent flow. *Computer Methods in Applied Mechanics and Engineering*, 199(13-16):853 – 864, 2010. Turbulence Modeling for Large Eddy Simulations.

[46] V. Gravemeier, W. A. Wall, and E. Ramm. A three-level finite element method for the instationary incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 193(15-16):1323 – 1366, 2004. Recent Advances in Stabilized and Multiscale Finite Element Methods.

[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. *Computer Methods in Applied Mechanics and Engineering*, 261–262(0):154 – 166, 2013.

[48] E. Hachem, S. Feghali, R. Codina, and T. Coupez. Anisotropic adaptive meshing and monolithic Variational Multiscale method for fluid-structure interaction. *Computers & Structures*, 122:88 – 100, 2013.

[49] F. E. Ham, F. Lien, and A. B. Strong. A fully conservative second-order finite difference scheme for incompressible flow on nonuniform grids. *Journal of Computational Physics*, 177(1):117 – 133, 2002.

[50] G. Hauke, M. H. Doweidar, and M. Miana. The multiscale approach to error estimation and adaptivity. *Computer Methods in Applied Mechanics and Engineering*, 195(13-16):1573–1593, 2006. A Tribute to Thomas J.R. Hughes on the Occasion of his 60th Birthday.

[51] G. Hauke, D. Fuster, and F. Lizarraga. Variational multiscale a posteriori error estimation for systems: The Euler and Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 283:1493 – 1524, 2015.

[52] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. An overview of the Trilinos project. *ACM Transactions on Mathematical Software*, 31(3):397–423, 2005.

[53] T. J. R. Hughes. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, 127(1-4):387–401, 1995.

[54] T. J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Inc, 2000.

[55] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy. The variational multiscale method–a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166(1-2):3–24, 1998. Advances in Stabilized Methods in Computational Mechanics.

[56] T. J. R. Hughes, L. P. Franca, and G. M. Hulbert. A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2):173–189, 1989.

[57] 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. *Computer Methods in Applied Mechanics and Engineering*, 58(3):305–328, 1986.

[58] T. J. R. Hughes, L. Mazzei, and K. E. Jansen. Large eddy simulation and the variational multiscale method. *Computing and Visualization in Science*, 3:47–59, 2000.

[59] T. J. R. Hughes, L. Mazzei, A. A. Oberai, and A. A. Wray. The multiscale formulation of large eddy simulation: Decay of homogeneous isotropic turbulence. *Physics of Fluids*, 13(2):505–512, 2001.

[60] T. J. R. Hughes and J. R. Stewart. A space-time formulation for multiscale phenomena. *Journal of Computational and Applied Mathematics*, 74(1-2):217–229, 1996.

[61] V. John. A numerical study of a posteriori error estimators for convection-diffusion equations. *Computer Methods in Applied Mechanics and Engineering*, 190(5-7):757 – 781, 2000.

[62] J. C. Kaimal, J. C. Wyngaard, Y. Izumi, and O. R. Coté. Spectral characteristics of surface-layer turbulence. *Quarterly Journal of the Royal Meteorological Society*, 98(417):563–589, 1972.

[63] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[64] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Doklady Akademiia Nauk SSSR*, 30:301–305, 1941.

[65] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 434(1890):9–13, 1991.

[66] A. G. Kravchenko and P. Moin. Numerical studies of flow over a circular cylinder at $Re_D = 3900$. *Physics of Fluids*, 12(2):403–417, February 2000.

[67] A. Larese. *A coupled Eulerian-PFEM model for the simulation of overtopping in rockfill dams*. PhD thesis, Universitat Politècnica de Catalunya. UPC BarcelonaTech, 2012.

[68] B. E. Launder and B. I. Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in Heat and Mass Transfer*, 1(2):131 – 137, 1974.

[69] J. H. Lienhard. Synopsis of lift, drag, and vortex frequency data for rigid circular cylinders. Technical report, Washington State University, 1966.

[70] R. Löhner. Adaptive remeshing for transient problems. *Computer Methods in Applied Mechanics and Engineering*, 75(13):195 – 214, 1989.

[71] H. Lu and F. Porté-Agel. A modulated gradient model for large-eddy simulation: Application to a neutral atmospheric boundary layer. *Physics of Fluids*, 22(1), 2010.

[72] J. Lubliner. *Plasticity Theory*. Macmillan Publishing Company, New York, 1990.

[73] J. Mann. Wind field simulation. *Probabilistic Engineering Mechanics*, 13(4):269 – 282, 1998.

[74] Message Passing Interface Forum. MPI: A message passing interface standard. Version 2.1, 2008. Available online from `www.mpi-forum.org/docs/`.

[75] M. Mier-Torrecilla, E. Herrera, and M. Doblaré. Numerical calculation of wind loads over solar collectors. *Energy Procedia*, 49(0):163 – 173, 2014. Proceedings of the SolarPACES 2013 International Conference.

[76] J. Misra and D. Gries. A constructive proof of Vizing's theorem. *Information Processing Letters*, 41(3):131 – 133, 1992.

[77] E. Mitsoulis. Flows of viscoplastic materials: models and computations. In *Rheology Reviews 2007*. British Society of Rheology, 2007.

[78] E. Mitsoulis and R. R. Huilgol. Entry flows of Bingham plastics in expansions. *Journal of Non-Newtonian Fluid Mechanics*, 122(1–3):45 – 54, 2004. *XIIIth* International Workshop on Numerical Methods for Non-Newtonian Flows.

[79] E. Mitsoulis and T. Zisis. Flow of Bingham plastics in a lid-driven square cavity. *Journal of Non-Newtonian Fluid Mechanics*, 101(13):173 – 180, 2001.

[80] E. Moreno and M. Cervera. Elementos finitos mixtos estabilizados para flujos confinados de Bingham y de Herschel-Bulkley Parte II: soluciones numéricas. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 2015. In press.

[81] R. D. Moser, J. Kim, and N. N. Mansour. Direct numerical simulation of turbulent channel flow up to $\mathrm{Re}_\tau = 590$. *Physics of Fluids*, 11(4):943–945, 1999.

[82] C. Norberg. Fluctuating lift on a circular cylinder: review and new measurements. *Journal of Fluids and Structures*, 17(1):57–96, 2003.

[83] A. A. Oberai and J. Wanderer. A dynamic approach for evaluating parameters in a numerical method. *International Journal for Numerical Methods in Engineering*, 62(1):50–71, 2005.

[84] A. A. Oberai and J. Wanderer. Variational formulation of the Germano identity for the Navier–Stokes equations. *Journal of Turbulence*, 6:1–17, 2005.

[85] J. G. Oldroyd. A rational formulation of the equations of plastic flow for a Bingham solid. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43:100–105, 1 1947.

[86] E. Oñate. Derivation of stabilized equations for numerical solution of advective-diffusive transport and fluid flow problems. *Computer Methods in Applied Mechanics and Engineering*, 151(1 - 2):233 – 265, 1998. Containing papers presented at the Symposium on Advances in Computational Mechanics.

[87] E. Oñate. A stabilized finite element method for incompressible viscous flows using a finite increment calculus formulation. *Computer Methods in Applied Mechanics and Engineering*, 182(3-4):355 – 370, 2000.

[88] E. Oñate. Possibilities of finite calculus in computational mechanics. *International Journal for Numerical Methods in Engineering*, 60(1):255–281, 2004.

[89] E. Oñate, A. Franci, and J. M. Carbonell. Lagrangian formulation for finite element analysis of quasi-incompressible fluids with reduced mass losses. *International Journal for Numerical Methods in Fluids*, 74(10):699–731, 2014.

[90] E. Oñate, S. R. Idelsohn, and C. A. Felippa. Consistent pressure Laplacian stabilization for incompressible continua via higher-order finite calculus. *International Journal for Numerical Methods in Engineering*, 87(1-5):171–195, 2011.

[91] E. Oñate, P. Nadukandi, S. R. Idelsohn, J. García, and C. Felippa. A family of residual-based stabilized finite element methods for Stokes flows. *International Journal for Numerical Methods in Fluids*, 65(1-3):106–134, 2011.

[92] E. Oñate, A. Valls, and J. García. Computation of turbulent flows using a finite calculus-finite element formulation. *International Journal for Numerical Methods in Fluids*, 54(6-8):609–637, 2007.

[93] E. Oñate, A. Valls, and J. García. Modeling incompressible flows at low and high Reynolds numbers via a finite calculus-finite element approach. *Journal of Computational Physics*, 224(1):332 – 351, 2007. Special Issue Dedicated to Professor Piet Wesseling on the occasion of his retirement from Delft University of Technology.

[94] L. Ong and J. Wallace. The velocity field of the turbulent very near wake of a circular cylinder. *Experiments in Fluids*, 20:441–453, 1996. 10.1007/BF00189383.

[95] T. C. Papanastasiou. Flows of materials with yield. *Journal of Rheology (1978-present)*, 31(5):385–404, 1987.

[96] A. Papastavrou and R. Verfürth. A posteriori error estimators for stationary convection-diffusion problems: a computational comparison. *Computer Methods in Applied Mechanics and Engineering*, 189(2):449 – 462, 2000.

[97] P. P. Pébay. Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments. Technical Report SAND2008-6212, Sandia National Laboratories, 2008.

[98] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72(2):449 – 466, 1987.

[99] D. Peric and S. Slijepcevic. Computational modelling of viscoplastic fluids based on a stabilised finite element method. *Engineering Computations*, 18(3/4):577–591, 2001.

[100] U. Piomelli and E. Balaras. Wall-layer models for large-eddy simulations. *Annual Review of Fluid Mechanics*, 34:349–374, 2002.

[101] S. B. Pope. *Turbulent Flows*. Cambridge University Press, Sept. 2000.

[102] J. Príncipe, R. Codina, and F. Henke. The dissipative structure of variational multiscale methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 199(13-16):791–801, 2010. Turbulence Modeling for Large Eddy Simulations.

[103] P. J. Richards, R. P. Hoxey, and L. J. Short. Wind pressures on a 6m cube. *Journal of Wind Engineering and Industrial Aerodynamics*, 89(14–15):1553–1564, 2001.

[104] N. Roquet and P. Saramito. An adaptive finite element method for Bingham fluid flows around a cylinder. *Computer Methods in Applied Mechanics and Engineering*, 192(31-32):3317 – 3341, 2003.

[105] N. Roquet and P. Saramito. An adaptive finite element method for viscoplastic flows in a square pipe with stick-slip at the wall. *Journal of Non-Newtonian Fluid Mechanics*, 155(3):101 – 115, 2008.

[106] R. Rossi, J. Cotela-Dalmau, N. M. Lafontaine, P. Dadvand, and S. R. Idelsohn. Parallel adaptive mesh refinement for incompressible flow problems. *Computers & Fluids*, 80:324–355, 2013.

[107] R. Rossi, M. Lazzari, and R. Vitaliani. Wind field simulation for structural engineering purposes. *International Journal for Numerical Methods in Engineering*, 61(5):738–763, 2004.

[108] Y. Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[109] P. Saramito and N. Roquet. An adaptive finite element method for viscoplastic fluid flows in pipes. *Computer Methods in Applied Mechanics and Engineering*, 190(40–41):5391 – 5412, 2001.

[110] V. Selmin and L. Formaggia. Simulation of hypersonic flows on unstructured grids. *International Journal for Numerical Methods in Engineering*, 34(2):569–606, 1992.

[111] H. J. Sheen, W. J. Chen, and J. S. Wu. Flow patterns for an annular flow over an axisymmetric sudden expansion. *Journal of Fluid Mechanics*, 350:177–188, 11 1997.

[112] D. Sieger, S. Menzel, and M. Botsch. RBF morphing techniques for simulation-based design optimization. *Engineering with Computers*, 30(2):161–174, 2014.

[113] E. Simiu and R. H. Scanlan. *Wind effects on structures: fundementals and applications to design*. John Wiley & Sons, Ltd, 1996.

[114] J. Smagorinsky. General circulation experiments with the primitive equations: I The basic equations. *Monthly Weather Review*, 91:99–164, 1963.

[115] P. R. Souza Mendes and E. S. S. Dutra. Viscosity function for yield-stress liquids. *Applied Rheology*, 14(6):296–302, 2004.

[116] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. *Recherche Aerospatiale*, 1:5–21, 1994.

[117] G. Steber. Evaluation of the finite element method for turbulent flows with the open source software Kratos. Master's thesis, Technische Universität München, June 2012.

[118] E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer. TOP500 list. `http://www.top500.org`. June 2015 edition.

[119] K. A. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1–2):281 – 309, 2001. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[120] R. B. Stull. *An introduction to boundary layer meteorology*. Springer, 1988.

[121] R. I. Tanner and J. F. Milthorpe. Numerical simulation of the flow of fluids with yield stress. In C. Taylor, J. A. Johnson, and W. R. Smith, editors, *Numerical Methods for Laminar and Turbulent Flow*, pages 680–690. Pineridge Press, 1983. Proceedings of the Third International Conference, Seattle.

[122] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73 – 100, 1973.

[123] A. E. Tejada-Martínez and K. E. Jansen. On the interaction between dynamic model dissipation and numerical dissipation due to streamline upwind/Petrov-Galerkin stabilization. *Computer Methods in Applied Mechanics and Engineering*, 194(9-11):1225 – 1248, 2005.

[124] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis*. American Mathematical Soc., 2001.

[125] H. Tennekes and J. L. Lumley. *A first course in turbulence*. MIT Press, 1972.

[126] T. P. Terriberry. Computing higher-order moments online. `http://people.xiph.org/~tterribe/notes/homs.html`, 2008. Retrieved on October 2015.

[127] A. V. Trofimova, A. E. Tejada-Martínez, K. E. Jansen, and R. T. L. Jr. Direct numerical simulation of turbulent channel flows using a stabilized finite element method. *Computers & Fluids*, 38(4):924 – 938, 2009.

[128] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic press, 2000.

[129] R. S. Tuminaro, M. A. Heroux, S. A. Hutchinson, and J. N. Shadid. Official Aztec users's guide version 2.1. Technical Report SAND99-8801J, Sandia National Laboratories, 1999.

[130] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[131] H. Werner and H. Wengle. Large-eddy simulation of turbulent flow over and around a cube in a plate channel. In F. Durst, R. Friedrich, B. E. Launder, F. W. Schmidt, U. Schumann, and J. H. Whitelaw, editors, *Turbulent Shear Flows 8. Selected Papers from the Eighth International Symposium on Turbulent Shear Flows, Munich, Germany, September 9-11 1991*, pages 155–168. Springer Berlin Heidelberg, 1993.

[132] D. C. Wilcox. *Turbulence Modeling for CFD*. DCW industries, 1998.

[133] O. C. Zienkiewicz and J. Wu. Automatic directional refinement in adaptive analysis of compressible flows. *International Journal for Numerical Methods in Engineering*, 37(13):2189–2210, 1994.