Low-order face-based approaches for incompressible computational fluid dynamics

by

Luan Malikoski Vieira

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in

Civil Engineering
Simulation in Engineering and Entrepreneurship Development

Universitat Politècnica de Catalunya
ETS d'Enginyeria de Camins, Canals i Ports
Laboratori de Càlcul Numèric

in cotutelle with

Swansea University
Faculty of Science and Engineering
Zienkiewicz Centre for Computational Engineering

Advisors: Antonio Huerta, Matteo Giacomini, Rubén Sevilla

Barcelona, October 29, 2024

Declarations

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed . [redacted] ....

Date: October 29, 2024 ....

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed . [redacted] ....

Date: October 29, 2024 ....

I hereby give consent for my thesis, if accepted, to be available for electronic sharing.

Signed . [redacted] ....

Date: October 29, 2024 ....

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

Signed . [redacted] ....

Date: October 29, 2024 ....

Abstract

The fast and accurate simulation of laminar and turbulent incompressible flows is crucial in science and engineering. Low-order numerical strategies, particularly finite-volume (FV) methods based on cell-centred and vertex-centred approaches, remain essential in CFD, especially in industrial settings, due to their favourable trade-off between computational cost and accuracy. However, these methods face challenges such as stabilising convective-dominated flows, handling velocity-pressure coupling, and managing numerical flux reconstruction on distorted and stretched grids. The face-centred finite volume (FCFV) method emerges as an alternative to standard FV methods. Derived from a mixed formulation of the discontinuous Galerkin method, FCFV avoids flux reconstruction at cell faces, making its accuracy and convergence nearly insensitive to grid quality. It also satisfies the Ladyzhenskaya-Babuška-Brezzi (LBB) condition without special treatment for velocity-pressure coupling. This work showcases the FCFV method for simulating laminar and turbulent incompressible flows for the first time. The formulation is based on the Reynolds-averaged Navier-Stokes (RANS) equations coupled with the negative Spalart-Allmaras (SA) model. Three new convective stabilisations inspired by Riemann solvers are proposed, along with monolithic and staggered solution strategies for the RANS-SA system and two relaxation strategies for pseudo-time marching. A new hybrid pressure FCFV formulation is also introduced to improve the FCFV accuracy in laminar flow simulations by enriching the pressure space through relaxed compressibility conditions. Both FCFV and hybrid pressure FCFV achieve first-order convergence of velocity, velocity gradient tensor, and pressure, accurately predicting engineering quantities such as drag and lift on structured and unstructured meshes. By avoiding gradient reconstruction, these methods are less sensitive to mesh quality, even on highly distorted grids. Numerical benchmarks for laminar and turbulent, steady and transient cases assess the performance, accuracy, and robustness of the proposed methodologies. Implemented in Fortran 90, these methods lay the foundation for the integration of FCFV techniques within the CFD community. The work concludes with a detailed discussion of the `Fortran 90` FCFV code implementation.

Keywords: Finite volumes, face-centred, Incompressible flows, Hybridisable discontinuous Galerkin, Sparlat-Allmaras, CFD, Fortran 90

Acknowledgments

I would like to express my gratitude to my advisors, Prof. Antonio Huerta and Dr. Matteo Giacomini, from the LaCáN group at Universitát Politécnica de Catalunya (UPC), and Prof. Rubén Sevilla from Swansea University, for their support, guidance, and dedication throughout countless meeting hours during this journey.

I am also thankful to the members of the ProTechTion consortium, CIMNE, UPC, and Swansea University, for their administrative efforts in making this project and cotutelle a reality.

I am grateful for the friends I made during my time at LaCáN, with whom I shared wonderful moments, scientific knowledge, support, and motivation throughout these years. I am equally thankful to the fantastic friends I made in Barcelona outside of academia, who helped me find the balance I needed to stay focused on my goals. My heartfelt thanks also go to my lifelong friends for their unwavering support and companionship, despite the distance.

Finally, I am deeply thankful to my family and Noemi for being my foundation, always believing in me, and offering support and encouragement. Without them, this journey would have been far less meaningful.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The simulation of incompressible flows is central to many areas of science and engineering, including aerodynamic design in the automotive industry, the modelling of water quality, and the study of blood flow in the human circulatory system, just to name a few. The growing complexity of fluid flow models necessitates the development of accurate yet efficient methodologies to solve such problems. To meet these requirements, fast and accurate computation of fluid flow is critical for any computational fluid dynamics (CFD) methodology, particularly in an industrial framework.

The computation of incompressible fluid flows presents some numerical challenges due to the nonlinearity of the convection operator, the velocity-pressure coupling through the compressibility condition, and the stability of convection-dominated problems, as discussed by Donea and Huerta (2003); Moukalled et al. (2016). In addition to inherent numerical challenges in simulating incompressible fluid flow, turbulence modelling remains one of the most significant obstacles in the industry. Most industrial fluid flow problems are inherently turbulent, making accurate modelling essential yet complex. Selecting an appropriate way to model the turbulence is the primary challenge, which requires careful tailoring to the specific nature of the problem. From a computational standpoint, the requirement for very dense grids to well capture all turbulence scales presents a major challenge. Direct numerical simulations (DNS) of such a flow problem are still unpractical for engineering applications due to the high computational cost, and the Reynolds Averaged Navier-Stokes (RANS) model equations coupled with a suitable turbulence model are a common choice in industry design.

Several studies in the automotive industry, such as those by Jacuzzi and Granlund (2019); Ebrahim and Dominy (2020); Altinisik et al. (2015), demonstrate that Reynolds-Averaged Navier-Stokes (RANS) equations remain a valuable approach for calculating mean flow variables and dynamic quantities of interest, particularly when compared to more complex Large-Eddy Simulation (LES)-based models. By modelling the smaller scales of turbulence using an appropriate turbulence model, the RANS equations significantly reduce computational costs, albeit with some trade-off in accuracy. The most commonly used RANS turbulence models are two-equation models, such as the Shear Stress Transport (SST) model introduced by Menter (1994), the κ-ϵ model by Launder and Spalding (1974), and the κ-ω model by Wilcox (1988). Despite their simplicity, one-equation models offer a desirable balance between modelling accuracy and computational efficiency. Among these, the Spalart-Allmaras (SA) turbulence model, introduced in Spalart and Allmaras (1992, 1994); Allmaras et al. (2012), is the most widely used one-equation model for aerodynamic applications. It has several variations still in development, including the negative model Allmaras et al. (2012), corrections for rotating and curved walls Shur et al. (2000), modifications to account for wall roughness Aupoix and Spalart (2003), and improvements for low Reynolds number behaviour Spalart and Garbaruk (2020).

Regarding the order of approximation of the numerical methods, the application of higher-order techniques in the industrial context is constrained by two main engineering bottlenecks: the high computational cost due to the increased number of degrees of freedom and the difficulty of generating suitable higher-order grids. The latter requires specialised expertise and often involves significant engineering hours during the initial design phase. As a result, in the CFD industry, the most commonly used commercial solvers and grid generators are those suited for low-order unstructured grids. These solvers offer greater flexibility in terms of grid type and topology, enabling rapid engineering solutions in the fast-paced industrial environment.

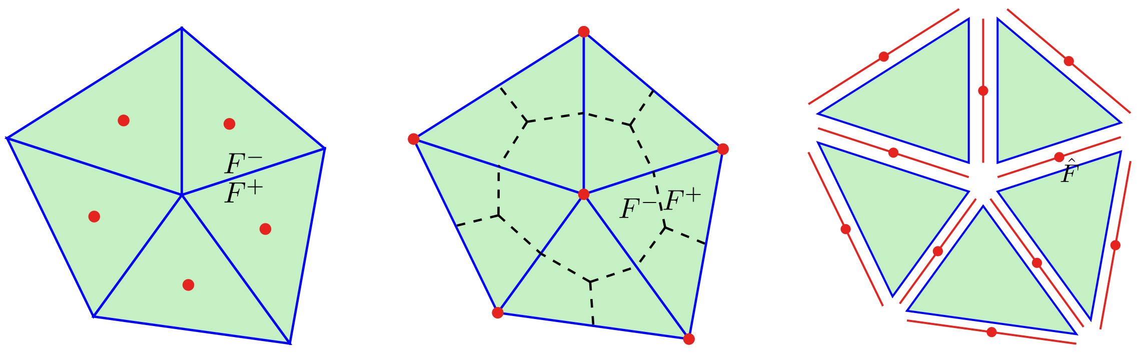
From a historical perspective, the development of most industrial and academic CFD codes since the 1980s has been based on the Finite Volume (FV) methodology. In this approach, the integral form of the conservation equations is solved within an arbitrary control volume, ensuring that conservation laws are upheld in each cell and across the entire computational domain. Because the conservative form of the equations is enforced at the cell level, flux terms are approximated at the cell faces. The balance and conservation of fluxes are fundamental aspects of the Finite Volume (FV) method, as highlighted by Barth et al. (2017); Diskin et al. (2010); Diskin and Thomas (2011); Droniou (2014); Morton and Sonar (2007). This inherent conservation property of the FVM, combined with its straightforward implementation, flexibility in grid topology, applicability to arbitrary polygonal element types, and relatively low computational cost inherent to low-order methods, has made it a preferred choice for CFD applications in industry, as well as in heat and mass transfer, for many years, as noted by Jameson (2012); Cockburn et al. (2009); Moukalled et al. (2016); Shang (2004). In fact, the majority of computational fluid dynamics (CFD) tools –open-source, commercial, or industrial– are based on either cell-centred (CCFV) or vertex-centred (VCFV) finite volume (FV) techniques, as outlined in LeVeque (2002); Barth et al. (2017). Also, given the limitations of high-order methodologies, low-order technologies may remain the major engineering tools for a while.

Prominent CFD codes based on finite volume methods include the TAU-code, developed at the DLR German Aerospace Center. This code utilises a second-order vertex-centred finite volume (VCFV) scheme to handle computations of inviscid, viscous, subsonic, and supersonic flows on hybrid unstructured meshes, making it widely used in the aerospace industry across Europe Schwamborn et al. (2008, 2006). Another notable example is the Flite system developed by researchers at Swansea University, which also employs a second-order FVM scheme, as discussed in Sørensen et al. (2003). This technology has been in use by BAE Systems since 1994 and has played a significant role in the aerodynamic design of the AIRBUS/UK A380 as well as supersonic vehicles in the Thrust SSC and Bloodhound SSC projects. The FUND3D code, developed at NASA Langley Research Centre, is another example of a lower-order CFD code with capabilities for unstructured grid simulations of incompressible and compressible transonic flows, as demonstrated in Biedron et al. (2014); Lee-Rausch et al. (2015). This code is based on a vertex-centred finite volume (VCFV) scheme with second-order accuracy. OpenFOAM, as described in Jasak (2009), is one of the leading open-source CFD libraries and employs a cell-centred finite volume (CCFV) paradigm. Due to its flexibility for testing, implementation, and sharing of innovations within the CFD community, it finds widespread use across various industries, from automotive to chemical, and is also prevalent in academia. Among major commercial CFD codes, Ansys-CFX, Ansys-Fluent, and STAR-CCM+ are notable finite volume (FV) solvers that employ the cell-centred finite volume (CCFV) method.

With recent advancements in finite element method (FEM) formulations for fluid flows, industrial software packages are increasingly utilising FEM-based solvers for a wide range of applications, including structural analysis, heat transfer, chemical engineering, electromagnetics, multi-physics, and CFD. Notable examples of FEM-based CFD packages include COMSOL Multiphysics and Autodesk Simulation CFD, which are well-regarded in the market. Over the past two decades, discontinuous Galerkin (DG) methods have gained significant traction in the scientific community, and this new class of discontinuous Galerkin formulations is now considered the future of CFD and computational mechanics in general, as explained by Jameson (2012).

1.1 Finite volume drawbacks in CFD context

As mentioned earlier, the two most common finite volume methods are the cell-centred finite volume (CCFV) and vertex-centred finite volume (VCFV) approaches, as detailed in Diskin et al. (2010); Morton and Sonar (2007); Moukalled et al. (2016). In the CCFV method, the unknowns are defined at the centroid of each cell, representing the cell-averaged solution, as illustrated in Fig. 1.1a. Fluxes are computed across the cell faces as a function of the variables at the cell centers. Conversely, the VCFV approach defines the unknowns at the cell nodes.



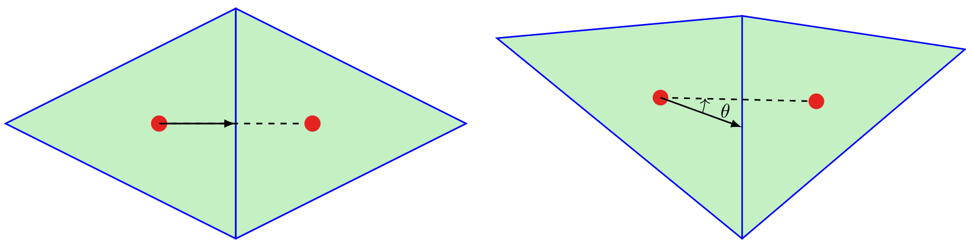
(a) CCFV (b) VCFV (c) FCFV

Figure 1.1: Finite volume methods: Location of degrees of freedom and numerical flux evaluation for (a) cell-centred finite volume (CCFV), (b) vertex-centred finite volume, and (c) face-centred finite volume (FCFV).

Non-overlapping control volumes are formed around each grid node by connecting each cell centroid to the centroids of its faces, as shown in Fig. 1.1b. Each nodal solution represents the average solution on the corresponding dual mesh cell centroid, and fluxes are computed through the faces of these dual control volumes. The main difficulty in devising an accurate finite volume approximation relies on the appropriate approximation of the numerical fluxes at the control volume boundary. Since the finite volume variables are defined at cell centres, for the CCFV, or at dual cell centres, for the VCFV, the fluxes at each control volume interface arise from an interpolation of the control volume centre variables. In the finite volume context, we often regard this interpolation as a reconstruction of gradient or flux.

Typical CFD computations using RANS equations involve grids that are often highly stretched near the domain walls to resolve the boundary layer, while also being unstructured with considerable distortion in the bulk flow region. These grid characteristics can adversely affect standard finite volume (FV) computations. Despite being formally second-order accurate, the accuracy of FV solutions is significantly degraded when highly non-orthogonal and stretched grids are used, as highlighted in several studies, such as Barth et al. (2017); Droniou (2014); Moukalled et al. (2016); Morton and Sonar (2007); Diskin et al. (2010); Diskin and Thomas (2011).

The cell non-orthogonality is directly proportional to the angle θ between the line that connects its barycenter to its neighbour barycenter and the line that connects its barycenter to the shared face barycenter, as graphically described in Fig. 1.2. A poor gradient reconstruction obtained when the grid non-orthogonality is beyond a threshold level compromises the FV accuracy and order of convergence. This is because the accuracy of the interpolation of the control volume variables to the faces is very sensitive to the alignment of the neighbouring cell centroids and shared faces centroids, which is reflected in the non-orthogonality



(a) Orthogonal (b) Non-orthogonal

Figure 1.2: Grid quality measured in terms of orthogonality metric: (a) perfectly orthogonal grid, (b) grid with a level of non-orthogonality.

metric, as explained in Moukalled et al. (2016). In fact, the FV solution can only achieve second-order convergence if the gradient computation achieves first-order convergence. If the FV method yields sub-optimal convergence for the solution and its gradient, it can result in poor estimation of common engineering quantities such as drag and lift aerodynamical forces.

In Droniou (2014), a second-order reconstruction method is suggested. The study demonstrates that if a mesh fails to meet the so-called orthogonality condition, this method fails to provide the optimal order of convergence for the flow variables. On the other hand, higher-order reconstruction techniques are available with increased computational cost and implementation difficulty. In Diskin et al. (2010); Diskin and Thomas (2011), several high-order methods are tested for stretched and distorted meshes, and results show a high loss in accuracy for the solution and its gradient.

In addition to the problems that arise when considering grids of poor quality, most gradient reconstruction techniques necessitate iterative processes, thereby increasing the computational cost of a finite volume algorithm. Further, the CFD community is well aware that special treatment for the velocity-pressure couple is required for incompressible flows. This is because the standard finite volume methods fail to fulfil the so-called Ladyzhenskaya-Babuška-Brezzi (LBB) condition, leading to spurious oscillations in the numerical solution when the same order of interpolation is adopted for velocity and pressure variables, as studied by Donea and Huerta (2003) and Moukalled et al. (2016). One remedy for such issues is the adoption of iterative algorithms such as SIMPLE, introduced by Patankar and Spalding (1972), and its variations.

1.2 From high-order to low-order face-based approaches

Since the early 1970s, finite element methods (FEM) have been applied to fluid flow problems. However, FEM faced stability challenges in convection-dominated problems and constraints on the selection of finite element spaces for flow variables to satisfy LBB condition, leading to the development of stabilised FEM formulations, as explored by Donea and Huerta (2003). Within the framework of discontinuous approaches, a class of methods known as discontinuous Galerkin (DG) emerged in 1973, introduced by Reed and Hill (1973) for solving hyperbolic equations. In a general sense, DG methods share strong similarities with finite volume methods and are often regarded as a generalisation of the FV method, as described by Arnold et al. (2000); Giacomini et al. (2020). The DG approach addresses FEM stability issues by incorporating inter-element fluxes, as discussed in Cockburn et al. (2000) and Giacomini et al. (2020). However, the primary drawback of DG methods is the relatively high number of coupled degrees of freedom.

Following advancements in DG methods, hybrid formulations gained attention, particularly after the introduction of the hybridisable discontinuous Galerkin (HDG) method by Cockburn and Gopalakrishnan (2004) and Cockburn et al. (2009). According to Cockburn and Gopalakrishnan (2005b), a hybrid formulation of a finite element method is defined as any method where one unknown is a function or one of its derivatives within the set Ω (referred to as the dual or mixed variable), and the other unknown is the trace of the function or the trace of one of its derivatives on the boundaries of the set $\partial\Omega$ (referred to as the hybrid variable). This formulation sets the HDG approach apart from DG because, in HDG, the only globally coupled degree of freedom is the trace of the solution on the element boundaries, i.e., the hybrid variable. This significantly reduces the number of coupled degrees of freedom compared to DG, resulting in more efficient implementations as described by Cockburn et al. (2009); Giacomini et al. (2020). Furthermore, as discussed in Cockburn and Gopalakrishnan (2005b); Cockburn et al. (2009, 2010), the hybrid variable at the boundaries incorporates additional information about the exact solution, enabling the HDG method to achieve optimal convergence for both primal and dual variables.

From the perspective of low-order face-based methods, the face-centred finite volume (FCFV) was initially proposed in Sevilla et al. (2018) for the solution of elliptic problems as an alternative to cell-centred or vertex-centred approaches, avoiding any reconstruction of the derivatives. In such an approach, the solution of the global system of equations lies on the element faces, where numerical fluxes are evaluated, as seen in Fig. 1.1c. This contrasts with the traditional CCFV and VCFV methods, where the solution is found at the cell centre and the cell vertex, respectively, requiring interpolation of the solution for flux evaluation. The method is derived from the HDG formulation when the interpolation space of primal, dual, and hybrid variables is constant. As a result, first-order convergence is achieved for both the solution and the gradient of the solution. In contrast with the lower-order CCFV and VCFV, the first-order convergence of the gradient is achieved without any reconstruction, given that the gradient is one of the variables in the method's solution set. Further, the method proved to be insensitive to mesh stretching and distortion. This is an important feature, as already discussed, given that lower-order methods based on CCFV or VCFV may lose their accuracy when highly distorted or stretched meshes are employed, as discussed by McBride et al. (2007); Sevilla et al. (2018).

Just as cell-centred FV methods can be seen as a low-order discontinuous Galerkin (DG) method with a constant degree of approximation in each cell and vertex-centred FV methods can be viewed as a conforming piecewise linear continuous finite element method, the FCFV can be interpreted as the hybridisable DG (HDG) method, introduced by Cockburn and Gopalakrishnan (2004, 2005a,b); Cockburn et al. (2009), using the lowest order approximation for cell and face variables. As a result, the method is nearly second-order in the sense that the velocity gradient converges with first-order accuracy without requiring reconstructions, making it insensitive to mesh distortion. Additionally, the method allows for the same degree of approximation for both velocity and pressure, thus circumventing the LBB condition Donea and Huerta (2003).

Since its introduction, the FCFV method has been applied to a variety of problems including solid mechanics Sevilla et al. (2019), heat transfer and Stokes flows Vieira et al. (2020); Giacomini and Sevilla (2020); Sevilla and Duretz (2023, 2024) and viscous laminar compressible flows Vila-Pérez et al. (2022); Vila-Pérez et al. (2023).

Despite the growing popularity of DG and HDG-based solvers for fluid flow problems, the literature only provides a few examples of their application to incompressible turbulent flow problems. In recent works, high-order discontinuous Galerkin formulation for the RANS equations and Sparlart-Allmaras (SA) turbulent model have been successfully implemented to solve incompressible aerodynamic problems in Crivellini and Bassi (2011); Crivellini et al. (2013); ZhenHua et al. (2016); Wurst et al. (2015). Difficulties in the case of SA implementation have been reported when dealing with the highly non-linear source terms, and several SA model method-tailored modifications have been proposed, as in the work of Crivellini and Bassi (2011) and ZhenHua et al. (2016). In the work of Peters and Evans (2019), a higher-order HDG implementation is proposed for the RANS SA model, and optimal convergence rates are achieved for all momentum and turbulence variables.

1.3 Thesis objectives and outline

The current work aims to provide fundamental grounds for the novel low-order face-based finite volume methods applied to the solution of incompressible laminar and turbulent flows. The methods developed target the main drawbacks of the standard FV methods in the CFD context. The key aspects making the novel strategies appealing to the CFD scientific community are listed as follows:

- No requirement for the numerical gradient reconstruction process.

- Robustness and flexibility regarding grid topology and quality.

- Capable to preserve optimal rates of convergence disregard grid quality.
- No requirement for special treatment of the velocity-pressure couple.

In previous works, the low-order face-based methods explored in this project, that is, FCFV, were applied to solve incompressible diffusion-dominated problems in Sevilla et al. (2018, 2019) and compressible flows in Vila-Pérez et al. (2022); Vila-Pérez et al. (2023). In this project, we extend these methodologies for the first time to the solution of laminar and turbulent convection-dominated incompressible flows while preserving all the key characteristics listed above, inherited from the original FCFV method.

This thesis also lays the foundation for the development of an `Fortran 90` FCFV CFD tool designed for fast, robust, and sufficiently accurate predictions of incompressible laminar and turbulent flows across a wide range of applications relevant to aerodynamics. The `Fortran 90` CFD library developed in this work includes the following features:

- Link to external libraries (`PETSc,MUMPS,HSL`) for sparse linear solvers.
- Steady solver equipped with time-relaxation strategies.
- Unsteady solver equipped with different time integrators (BDF2, and BE).
- Appliable to incompressible laminar (NS) and turbulent flows (RANS-SA).
- Monolithic and staggered RANS-SA solver.
- Appliable for 2-dimensional and 3-dimensional problems.
- Appliable for general unstructured/structured hybrid grids.

To showcase the outlined contributions, the thesis is organized as follows:

1. **A face-centred finite volume formulation for the solution of incompressible laminar (NS) and turbulent (RANS) flows.**

   The work published in Vieira et al. (2024) that develops for the first time the face-centred finite volume FCFV method to viscous incompressible flows to simulate steady-state and transient phenomena in both laminar and turbulent regimes is presented in an extended version in Chap. 2. The FCFV formulation of the incompressible Reynolds-averaged Navier-Stokes (RANS) equations coupled with the negative Spalart-Allmaras (SA) turbulence model of Allmaras et al. (2012) is first introduced. Attention is given to the *monolithic* and *staggered* approaches to solve the RANS-SA coupling. The particularisation of the RANS-SA formulation to the solution of laminar flows, namely Navier-Stokes equations, is also outlined. The computational aspects of the FCFV formulation regarding the size of the global system of equations and the number of sparse entries are also discussed in Chap. 2. The proposed FCFV numerical strategy has its performance and robustness assessed with an extensive set of laminar and turbulent examples in both steady and unsteady regimes for 2-dimensional and 3-dimensional settings. The laminar benchmark examples are found in Chap. 4, while the turbulent cases are shown in Chap. 5. The examples include benchmarks featuring highly distorted and stretched meshes, different grid types, and different Riemann solvers (see Appendix A). Additionally, the capability of the FCFV method to provide an optimal rate of convergence for the vorticity tensor magnitude is verified. For the RANS-SA case, the performance of *monolithic* and *staggered* approaches is also evaluated in the numerical examples.

2. **A hybrid pressure face-centred finite volume formulation for the solution of incompressible Navier-Stokes equations.**

   The novel hybrid pressure face-centred finite volume (hybrid pressure FCFV) formulation for the solution of the steady incompressible Navier-Stokes equations is proposed and introduced in Chap. 3. This formulation aims at providing enhanced performance of the standard FCFV, described in Chap. 2, on moderated Reynolds number in the laminar regime. As will be shown, the formulation relies on a relaxation of the divergence condition, in contrast to the standard FCFV of Vieira et al. (2024); Sevilla et al. (2018, 2019), and the enrichment of the pressure field by the introduction of a hybrid pressure variable. Another but not less important aspect of this formulation is the strong enforcement of the stress tensor symmetry, following Giacomini et al. (2018). The computational aspects of the hybrid pressure FCFV are also outlined in Chap. 3, where a comparison with the standard FCFV computational cost is provided. The proposed hybrid pressure FCFV numerical strategy has its performance and robustness assessed with an extensive set of steady-state laminar examples for 2-dimensional and 3-dimensional settings. The benchmark examples are found in Chap. 4. The examples include benchmarks featuring highly distorted and stretched meshes, different grid types, and different Riemann solvers (see Appendix A). Attention is drawn to the performance comparison between the hybrid pressure FCFV and the standard FCFV of Chap. 2.

3. **Riemann solvers for the convective stabilization.**

   Three novel stabilisation tensors for the convective part of the incompressible RANS, NS, and SA equations are proposed and derived in Appendix A. The proposed convective stabilisations are employed in all formulations discussed in this work, more precisely in Chap. 2 and Chap. 3. They are derived following the unified analysis presented in Vila-Pérez et al. (2021, 2022) for the Riemann solvers, introduced by Toro (2009), in the context of hybrid discretisation methods.

4. **A `Fortran 90` FCFV implementation.**

   The `Fortran 90` FCFV solver implementation is presented in Chap. 6. All key aspects of the `Fortran 90` FCFV are outlined. It starts from the data structure, which makes extensive use of the `Fortran 90` derived data type. Solver options, built on the derived data type structure introduced, are listed. The solver modular structure and routines classification and dependencies are discussed. The solver workflow is exposed, from the grid and input data reading to storage of solution data and quantities of interest. The time relaxation schemes are briefly explained. Special focus is given to the non-linear Newton-Raphson solver, where differences between the *monolithic* and *staggared* solver algorithms are shown. The linear solver and all important aspects related to the interface to `PETSc` library are explained. The core of the solver, that is, the build and assemble of the global vector and matrix, is detailed.

Chapter 2

The face-centred finite volume method for laminar and turbulent incompressible flows

This chapter introduces the face-centred finite volume (FCFV) method for the solution of the laminar and turbulent incompressible flows, published in Vieira et al. (2024)[1]. It starts in Section 2.1 with a brief presentation of the turbulent flow governing equations, modelled here with the RANS coupled with the Spalart-Allmaras turbulence model. In the following, Section 2.2, the FCFV formulation is presented, followed by the discretisation in Section 2.3. In Section 2.4, the two solution approaches to handle the RANS-SA coupling, namely, *monolithic* and *staggered* are delineated. The particularisation of the FCFV formulation for the solution of the incompressible Navier-Stokes equations in the realm of laminar flows is done in Section 2.5. Finally, in Section 2.6 some computational aspects of the FCFV implementation are discussed. The details on the numerical convective stabilisation, time relaxation strategies, and linearisation of the discrete FCFV forms are given in the appendices A, B, and C, respectively.

2.1 Governing equations

Let us consider an open bounded domain $\Omega \in \mathbb{R}^{\mathtt{n_{sd}}}$, where $\mathtt{n_{sd}}$ is the number of spatial dimensions, and a final time $T > 0$. The non-dimensional incompressible Reynolds-averaged Navier-Stokes (RANS) equations coupled with the negative Spalart-Allmaras (SA) turbu-

[1]This chapter presents an extended version of the FCFV formulation for the incompressible RANS-SA and incompressible Navier-Stokes equations published in: L. M. Vieira, M. Giacomini, R. Sevilla, and A. Huerta. **The Face-Centred Finite Volume Method for Laminar and Turbulent Incompressible Flows**. Computer and Fluids. 279:106339, 2024.

lence model, introduced in Allmaras et al. (2012), can be written as

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot\left(\dfrac{2(1+\nu_t)}{Re}\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}}\right) = \boldsymbol{s} & \text{in } \Omega\times(0,T], \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega\times(0,T], \\ \dfrac{\partial\tilde{\nu}}{\partial t} + \boldsymbol{\nabla}\cdot(\tilde{\nu}\boldsymbol{u}) - \boldsymbol{\nabla}\cdot\left(\dfrac{1+\tilde{\nu}f_n}{\sigma Re}\boldsymbol{\nabla}\tilde{\nu}\right) = s & \text{in } \Omega\times(0,T], \end{cases} \tag{2.1}$$

where $\boldsymbol{u}$ is the mean velocity vector field, arising from the Reynolds averaging, see Remark 2.1, p is the modified pressure field, which involves the density and the turbulent kinematic energy, the strain-rate tensor is $\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} = (\boldsymbol{\nabla}\boldsymbol{u} + \boldsymbol{\nabla}^T\boldsymbol{u})/2$, Re denotes the Reynolds number and $\boldsymbol{s}$ is the body force. The non-dimensional turbulent viscosity ν_t, introduced by the Spalart-Allmaras turbulence model, is computed in terms of the turbulent variable $\tilde{\nu}$ as $\nu_t = \tilde{\nu}f_{v1}$. The right-hand side of the SA turbulence model is given by

$$s = \begin{cases} c_{b1}(1-f_{t2})\tilde{S}\tilde{\nu} + \dfrac{c_{b2}}{\sigma Re}\boldsymbol{\nabla}\tilde{\nu}\cdot\boldsymbol{\nabla}\tilde{\nu} - \dfrac{1}{Re}\left(c_{w1}f_w - \dfrac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\dfrac{\tilde{\nu}}{d}\right)^2 & \text{if } \tilde{\nu}\geq 0, \\ c_{b1}(1-c_{t3})S\tilde{\nu} + \dfrac{c_{b2}}{\sigma Re}\boldsymbol{\nabla}\tilde{\nu}\cdot\boldsymbol{\nabla}\,\tilde{\nu} + \dfrac{c_{w1}}{Re}\left(\dfrac{\tilde{\nu}}{d}\right)^2 & \text{otherwise,} \end{cases} \tag{2.2}$$

where the three terms correspond to production, cross-diffusion, and destruction, respectively. In the production term, the modified vorticity magnitude, $\tilde{S}$, has a modification introduced by Allmaras et al. (2012) to avoid negative values, namely

$$\tilde{S} = \begin{cases} S + \bar{S} & \text{if } \bar{S} \geq -c_{v2}S, \\ S + \dfrac{S(Sc_{v2}^2 + \bar{S}c_{v3})}{S(c_{v3}-2c_{v2}) - \bar{S}} & \text{otherwise,} \end{cases} \tag{2.3}$$

where $S = \sqrt{2\boldsymbol{S}:\boldsymbol{S}}$ is the magnitude of the vorticity, $\boldsymbol{S} = (\boldsymbol{\nabla}\boldsymbol{u} - \boldsymbol{\nabla}^T\boldsymbol{u})/2$ is the vorticity tensor, $\bar{S} = \tilde{\nu}f_{v2}/(Re\,\kappa^2 d^2)$ and d is the minimum distance to the wall. The SA model is completed with the closure functions

$$\begin{aligned} &\chi = \tilde{\nu}, && f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, && f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, \\ &r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, r_{lim}\right), && g = r + c_{w2}(r^6 - r), && f_w = g\left(\frac{1+c_{w3}^6}{g^6 + c_{w3}^6}\right)^{1/6}, \\ &f_{t2} = c_{t3}\exp(-c_{t4}\chi^2), && f_n = \frac{c_{n1}+\chi^3}{c_{n1}-\chi^3}, \end{aligned} \tag{2.4}$$

and constants $\sigma = 2/3$, $c_{b1} = 0.1355$, $c_{b2} = 0.622$, $\kappa = 0.41$, $c_{w1} = c_{b1}/\kappa^2 + (1+c_{b2})/\sigma$, $c_{v1} = 7.1$, $c_{t3} = 1.2$, $c_{t4} = 0.5$, $c_{v2} = 0.7$, $c_{v3} = 0.9$, $r_{lim} = 10$, $c_{w2} = 0.3$, $c_{w3} = 2$, and $c_{n1} = 16$.

The FCFV method described in this work employs the velocity-pressure formulation of the momentum equation (2.10) to ensure that the vorticity $\boldsymbol{S}$ can be computed from the components of the velocity-gradient tensor $\boldsymbol{\nabla u}$. This is in contrast with other mixed formulations for incompressible flow problems such as Giacomini et al. (2018), and Giacomini et al. (2020); La Spina et al. (2020), that use the Cauchy formulation of the momentum equation and the symmetric gradient of the velocity as the mixed variable. This strategy is adopted to avoid performing a reconstruction of the derivatives of the velocity, as commonly done in other FV methods, which induces a loss of accuracy when employing distorted meshes.

The strong form of the incompressible RANS equations and SA turbulence model is closed with appropriate boundary conditions, written in the abstract form

$$\begin{cases} \boldsymbol{B}(\boldsymbol{u}, p, \boldsymbol{\nabla u}, \tilde{\nu}, \boldsymbol{g}) = \boldsymbol{0} & \text{on } \partial\Omega \ \times(0,T], \\ b(\boldsymbol{u}, \tilde{\nu}, \boldsymbol{\nabla}\tilde{\nu}) = 0 & \text{on } \partial\Omega \ \times(0,T], \end{cases} \tag{2.5}$$

using the boundary operators $\boldsymbol{B}$ and b. In this work, Dirichlet, Neumann, symmetry, and outflow boundary conditions are considered. The portions of the boundary $\partial\Omega$ where such conditions are imposed, denoted by Γ_D, Γ_N, Γ_O, and Γ_S, respectively, are disjoint by pairs and such that $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_O \cup \Gamma_S$. The particular expression of the boundary operators $\boldsymbol{B}$ and b is

$$\boldsymbol{B} := \begin{cases} \boldsymbol{u} - \boldsymbol{u}_D & \text{on } \Gamma_D, \\ \left(\frac{1+\nu_t}{Re}\boldsymbol{\nabla u} - \boldsymbol{u}\otimes\boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} - \boldsymbol{g} & \text{on } \Gamma_N, \\ \left(\frac{1+\nu_t}{Re}\boldsymbol{\nabla u} - p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} & \text{on } \Gamma_O, \\ \begin{Bmatrix} \boldsymbol{t}_k \cdot \left(\frac{1+\nu_t}{Re}\boldsymbol{\nabla u} - p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} \\ \boldsymbol{u}\cdot\boldsymbol{n} \end{Bmatrix} & \text{on } \Gamma_S, \end{cases} \tag{2.6a}$$

and

$$b := \begin{cases} \tilde{\nu} - \tilde{\nu}_D & \text{on } \Gamma_D, \\ \left(\frac{1+\tilde{\nu}f_n}{\sigma Re}\boldsymbol{\nabla}\tilde{\nu} - \tilde{\nu}\boldsymbol{u}\right)\boldsymbol{n} & \text{on } \Gamma_N \\ \left(\frac{1+\tilde{\nu}f_n}{\sigma Re}\boldsymbol{\nabla}\tilde{\nu}\right)\cdot\boldsymbol{n} & \text{on } \Gamma_O \cup \Gamma_S, \end{cases} \tag{2.6b}$$

where $\boldsymbol{n}$ and $\boldsymbol{t}_k$, for $k = 1, \dots, \mathtt{n_{sd}} - 1$, are the outward unit normal and unit tangential vectors to the boundary, $\boldsymbol{u}_D$ and $\tilde{\nu}_D$ are the imposed velocity and turbulent variable on the Dirichlet boundary, and $\boldsymbol{g}$ is the imposed traction on the Neumann boundary representing a material surface.

Finally, an initial condition is given for the velocity and turbulent variable, namely

$$\begin{cases} \boldsymbol{u} = \boldsymbol{u}_0 & \text{in } \Omega\times\{0\}, \\ \tilde{\nu} = \tilde{\nu}_0 & \text{in } \Omega\times\{0\}. \end{cases} \tag{2.7}$$

It is worth noting that the divergence-free condition in the RANS equations induces the compatibility condition on the velocity field given by

$$\int_{\Gamma_D} \boldsymbol{u}_D \cdot \boldsymbol{n}\, d\Gamma + \int_{\partial\Omega\setminus\Gamma_D} \boldsymbol{u} \cdot \boldsymbol{n}\, d\Gamma = 0. \tag{2.8}$$

In particular, if only Dirichlet boundary conditions are considered it is necessary to impose an extra constraint to eliminate the indeterminacy of the pressure field, as discussed in Giacomini et al. (2020).

Remark 2.1 (RANS equations and assumptions). From the Navier-Stokes momentum equation for an incompressible Newtonian fluid, we have

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = \boldsymbol{s}$$

from the incompressibility condition and Stokes hypothesis assumption, introduced by Stokes (1845), we have

$$\boldsymbol{\sigma} = -p\mathbf{I}_{\mathtt{n_{sd}}} + 2\nu\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u}, \tag{2.9}$$

with $\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} = (\boldsymbol{\nabla}\boldsymbol{u} + \boldsymbol{\nabla}^T\boldsymbol{u})/2$, p the kinematic pressure and ν the kinematic viscosity. After the averaging of the Navier-Stokes equations, by assuming that the flow velocity is composed of an average and a fluctuating part such that $\boldsymbol{u} = \bar{\boldsymbol{u}} + \boldsymbol{u}'$, it is, the Reynolds decomposition introduced in Reynolds (1895), we end up with the following RANS momentum equation

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial t} + \boldsymbol{\nabla}\cdot(\bar{\boldsymbol{u}}\otimes\bar{\boldsymbol{u}}) - \boldsymbol{\nabla}\cdot\left(2\nu\boldsymbol{\nabla}^{\mathtt{s}}\bar{\boldsymbol{u}} - p\mathbf{I}_{\mathtt{n_{sd}}}\right) + \boldsymbol{\nabla}\cdot(\overline{\boldsymbol{u}'\otimes\boldsymbol{u}'}) = \boldsymbol{s}$$

where $\bar{\boldsymbol{u}}$ and $\boldsymbol{u}'$ are the average and fluctuating part of the flow velocity $\boldsymbol{u}$. From the assumption of a linear viscosity model, the Reynolds Stress (i.e, the last term on the l.h.s.) can be written as

$$\overline{\boldsymbol{u}'\otimes\boldsymbol{u}'} = -2\nu_t\boldsymbol{\nabla}^{\mathtt{s}}\bar{\boldsymbol{u}} + \frac{2}{3}\kappa\mathbf{I}_{\mathtt{n_{sd}}}$$

with $\kappa = \frac{1}{2}\operatorname{tr}(\overline{\boldsymbol{u}'\otimes\boldsymbol{u}'})$ being the turbulent kinetic energy. It is common to write the kinematic pressure in a modified form such that $p + \frac{2}{3}\kappa$, and with the abuse of notation, we use p to denote this modified pressure. After substituting the new definitions of p and Reynolds stress into the RANS equations we obtain

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial t} + \boldsymbol{\nabla}\cdot(\bar{\boldsymbol{u}}\otimes\bar{\boldsymbol{u}}) - \boldsymbol{\nabla}\cdot\left(2(\nu+\nu_t)\boldsymbol{\nabla}^{\mathtt{s}}\bar{\boldsymbol{u}} - p\mathbf{I}_{\mathtt{n_{sd}}}\right) = \boldsymbol{s}$$

with an abuse of notation by denoting the average flow velocity as $\boldsymbol{u}$, the non-dimensional RANS momentum equation of (2.1) is obtained. In this work, $\boldsymbol{u}$ will denote the mean flow velocity whenever the RANS cases are concerned, for laminar cases $\boldsymbol{u}$ will represent the instantaneous flow velocity itself.

Remark 2.2. The conservation of momentum, see the first equation in (2.47), can be rewritten in the usual velocity-pressure formulation as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot\left(\frac{1+\nu_t}{Re}\boldsymbol{\nabla}\boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}}\right) - \frac{1}{Re}\boldsymbol{\nabla}^T\boldsymbol{u}\,\boldsymbol{\nabla}\nu_t = \boldsymbol{s}, \tag{2.10}$$

where $\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla}^T\boldsymbol{u})$ vanishes due to the point-wise divergence-free condition on the velocity.

2.2 Face-centred finite volume formulation

Let us consider a partition of the domain Ω in a set of non-overlapping cells $\{\Omega_e\}_{e=1,\dots,\mathtt{n_{el}}}$, where $\mathtt{n_{el}}$ is the number of cells. The boundary of each cell, $\partial\Omega_e$, is formed by a set of faces $\{\Gamma_{e,j}\}_{j=1,\dots,\mathtt{n}_{\mathtt{fa}}^e}$, where $\mathtt{n}_{\mathtt{fa}}^e$ is the number of faces of cell Ω_e. Finally the set of faces not on the boundary of the domain form the internal interface Γ, with $\Gamma := \left[\bigcup_{e=1}^{\mathtt{n_{el}}} \partial\Omega_e\right] \setminus \partial\Omega$.

2.2.1 Mixed formulation

In this broken domain, the incompressible RANS equations together with the SA turbulence model are written as a system of first-order equations after introducing the mixed variables $\boldsymbol{L} = -\boldsymbol{\nabla}\boldsymbol{u}$ and $\tilde{\boldsymbol{q}} = -\boldsymbol{\nabla}\tilde{\nu}$, namely

$$\left\{\begin{aligned}
\boldsymbol{L} + \boldsymbol{\nabla}\boldsymbol{u} &= \boldsymbol{0} && \text{in } \Omega_e\times(0,T],\\
\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) + \boldsymbol{\nabla}\cdot\left(\frac{1+\nu_t}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right) - \frac{1}{Re}\boldsymbol{L}^T\boldsymbol{q}_t &= \boldsymbol{s} && \text{in } \Omega_e\times(0,T],\\
\boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0 && \text{in } \Omega_e\times(0,T],\\
\tilde{\boldsymbol{q}} + \boldsymbol{\nabla}\tilde{\nu} &= \boldsymbol{0} && \text{in } \Omega_e\times(0,T],\\
\frac{\partial \tilde{\nu}}{\partial t} + \boldsymbol{\nabla}\cdot(\tilde{\nu}\boldsymbol{u}) + \boldsymbol{\nabla}\cdot\left(\frac{1+\tilde{\nu}f_n}{\sigma Re}\tilde{\boldsymbol{q}}\right) &= s && \text{in } \Omega_e\times(0,T],\\
\boldsymbol{B}(\boldsymbol{u}, p, \boldsymbol{L}, \tilde{\nu}, \boldsymbol{g}) &= \boldsymbol{0} && \text{on } \partial\Omega\times(0,T],\\
b(\boldsymbol{u}, \tilde{\nu}, \boldsymbol{q}) &= 0 && \text{on } \partial\Omega\times(0,T],
\end{aligned}\right. \tag{2.11a}$$

$$
\begin{cases}
\llbracket \boldsymbol{u} \otimes \boldsymbol{n} \rrbracket = \boldsymbol{0} & \text{on } \Gamma \times (0,T], \\
\left\llbracket (\boldsymbol{u} \otimes \boldsymbol{u})\boldsymbol{n} + \left(\dfrac{1+\nu_t}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} \right\rrbracket = \boldsymbol{0} & \text{on } \Gamma \times (0,T], \\
\llbracket \tilde{\nu} \boldsymbol{n} \rrbracket = \boldsymbol{0} & \text{on } \Gamma \times (0,T], \\
\left\llbracket (\tilde{\nu} \boldsymbol{u}) \cdot \boldsymbol{n} + \left(\dfrac{1+\tilde{\nu} f_n}{\sigma Re} \tilde{\boldsymbol{q}} \right) \cdot \boldsymbol{n} \right\rrbracket = 0 & \text{on } \Gamma \times (0,T],
\end{cases}
\tag{2.11b}
$$

where $\boldsymbol{q}_t := -\boldsymbol{\nabla} \nu_t = (4 - 3 f_{v1}) f_{v1} \tilde{\boldsymbol{q}}$, and the right-hand side of the SA model, introduced in (2.2), is also written in terms of the mixed variable $\tilde{\boldsymbol{q}}$ as

$$
s = \begin{cases}
c_{b1}(1 - f_{t2}) \tilde{S} \tilde{\nu} + \dfrac{c_{b2}}{\sigma Re} \tilde{\boldsymbol{q}} \cdot \tilde{\boldsymbol{q}} - \dfrac{1}{Re} \left(c_{w1} f_w - \dfrac{c_{b1}}{\kappa^2} f_{t2} \right) \left(\dfrac{\tilde{\nu}}{d} \right)^2 & \text{if } \tilde{\nu} \geq 0, \\
c_{b1}(1 - c_{t3}) S \tilde{\nu} + \dfrac{c_{b2}}{\sigma Re} \tilde{\boldsymbol{q}} \cdot \tilde{\boldsymbol{q}} + \dfrac{c_{w1}}{Re} \left(\dfrac{\tilde{\nu}}{d} \right)^2 & \text{otherwise,}
\end{cases}
\tag{2.12}
$$

The second equation in (2.11)(a) uses the rewriting of the momentum equation given by Remark 2.2. The equations in (2.11)(b), called transmission conditions, impose the continuity of the velocity, the turbulent variable, and the normal fluxes of the RANS and SA equations. In these four equations, the jump operator along a face in Γ, with neighbouring cells Ω_l and Ω_r, follows the notation introduced in Montlaur et al. (2008), defined as

$$
\llbracket \odot \rrbracket = \odot_l + \odot_r.
$$

2.2.2 Strong form of the local and global problems

Following the rationale of FCFV method, introduced in Sevilla et al. (2018, 2019) and extended in Giacomini and Sevilla (2020); Vieira et al. (2020); Vila-Pérez et al. (2022) and also the HDG method proposed by Cockburn and Gopalakrishnan (2009); Nguyen et al. (2010); Cockburn et al. (2010); Nguyen et al. (2011), the mixed formulation in (2.11) is solved in two steps. First, $\mathtt{n_{el}}$ *local problems* are defined, one per cell, to express the primal variables, $\boldsymbol{u}$, p and $\tilde{\nu}$, and the mixed variables, $\boldsymbol{L}$ and $\tilde{\boldsymbol{q}}$, as a function of new independent variables, $\hat{\boldsymbol{u}}$ and $\hat{\nu}$, defined on the boundary of the cell. Figure 2.1 shows a sketch of a triangular mesh with the variables defined on each cell and on each edge. As will be shown in this section, the global system of equations to be solved only involves the variables on the edges and the cell variables can then be recovered cell-by-cell. The local problems to

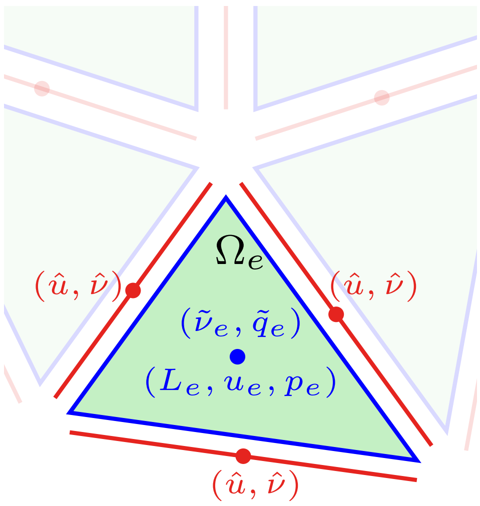


Figure 2.1: Detail of a triangular mesh highlighting a cell Ω_e where the velocity $\boldsymbol{u}_e$, velocity gradient $\boldsymbol{L}_e$, mean pressure p_e, turbulent variable $\tilde{\nu}_e$ and gradient of the turbulent variable $\tilde{\boldsymbol{q}}_e$ are defined and edges where the hybrid velocity $\hat{\boldsymbol{u}}$ and hybrid turbulent variable $\hat{\nu}$ are defined.

determine cell-by-cell $(\boldsymbol{L}, \boldsymbol{u}, p, \tilde{\boldsymbol{q}}, \tilde{\nu})$, for all $\boldsymbol{x} \in \Omega_e \subset \Omega$, are

$$\begin{cases} \boldsymbol{L} + \boldsymbol{\nabla} \boldsymbol{u} = \boldsymbol{0} & \text{in } \Omega_e \times (0, T], \\ \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) + \boldsymbol{\nabla} \cdot \left(\dfrac{1+\nu_t}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) - \dfrac{1}{Re} \boldsymbol{L}^T \boldsymbol{q}_t = \boldsymbol{s} & \text{in } \Omega_e \times (0, T], \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 & \text{in } \Omega_e \times (0, T], \\ \tilde{\boldsymbol{q}} + \boldsymbol{\nabla} \tilde{\nu} = \boldsymbol{0} & \text{in } \Omega_e \times (0, T], \\ \dfrac{\partial \tilde{\nu}}{\partial t} + \boldsymbol{\nabla} \cdot (\tilde{\nu} \boldsymbol{u}) + \boldsymbol{\nabla} \cdot \left(\dfrac{1+\tilde{\nu} f_n}{\sigma Re} \tilde{\boldsymbol{q}} \right) = s & \text{in } \Omega_e \times (0, T], \\ \boldsymbol{u} = \boldsymbol{u}_D & \text{on } \partial\Omega_e \cap \Gamma_D \times (0, T], \\ \tilde{\nu} = \tilde{\nu}_D & \text{on } \partial\Omega_e \cap \Gamma_D \times (0, T], \\ \boldsymbol{u} = \hat{\boldsymbol{u}} & \text{on } \partial\Omega_e \setminus \Gamma_D \times (0, T], \\ \tilde{\nu} = \hat{\nu} & \text{on } \partial\Omega_e \setminus \Gamma_D \times (0, T], \end{cases} \tag{2.13}$$

where $\hat{\boldsymbol{u}}$ and $\hat{\nu}$, known as *hybrid* variables, denote the trace of the velocity and turbulent variable on $\Gamma \cup \partial\Omega \setminus \Gamma_D$. The local problems only contain Dirichlet boundary conditions and, therefore, it is necessary to introduce an extra constraint to eliminate the indeterminacy of the pressure field. In this work, the constraint imposes the mean value of the pressure in each cell, ρ_e, namely

$$\frac{1}{|\Omega_e|} \int_{\Omega_e} p \, d\Omega = \rho_e. \tag{2.14}$$

In addition, as for the problem statement, see (2.8), the divergence-free condition in each cell entails the compatibility condition for the velocity along its boundary, namely,

$$\int_{\partial\Omega_e\cap\Gamma_D} \boldsymbol{u}_D\cdot\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D} \hat{\boldsymbol{u}}\cdot\boldsymbol{n}\,d\Gamma = 0. \tag{2.15}$$

The local problems provide an expression to compute $(\boldsymbol{L},\boldsymbol{u},p,\tilde{\boldsymbol{q}},\tilde{\nu})$, in each cell, in terms of the global unknowns $(\hat{\boldsymbol{u}},\boldsymbol{\rho},\hat{\boldsymbol{\nu}})$, where $\boldsymbol{\rho}=(\rho_1\dots,\rho_{\texttt{n}_{\texttt{el}}})^T$.

The second step consists of a *global problem* that involves only the global unknowns, accounting for the transmission conditions and the remaining boundary conditions,

$$\begin{cases} \left[\!\left[(\boldsymbol{u}\otimes\boldsymbol{u})\boldsymbol{n}+\left(\dfrac{1+\nu_t}{Re}\boldsymbol{L}+p\mathbf{I}_{\texttt{n}_{\texttt{sd}}}\right)\boldsymbol{n}\right]\!\right]=\boldsymbol{0} & \text{on } \Gamma\times(0,T],\\ \left[\!\left[(\tilde{\nu}\boldsymbol{u})\cdot\boldsymbol{n}+\left(\dfrac{1+\tilde{\nu}f_n}{\sigma Re}\tilde{\boldsymbol{q}}\right)\cdot\boldsymbol{n}\right]\!\right]=0 & \text{on } \Gamma\times(0,T],\\ \boldsymbol{B}(\boldsymbol{u},p,\boldsymbol{L},\tilde{\nu},\boldsymbol{g})=\boldsymbol{0} & \text{on } \partial\Omega\setminus\Gamma_D\times(0,T],\\ b(\boldsymbol{u},\tilde{\nu},\tilde{\boldsymbol{q}})=0 & \text{on } \partial\Omega\setminus\Gamma_D\times(0,T], \end{cases} \tag{2.16}$$

The continuity of the velocity and the turbulent variable across the internal interface Γ is not explicitly imposed in the global problem because it is automatically satisfied due to the Dirichlet boundary conditions $\boldsymbol{u}=\hat{\boldsymbol{u}}$ and $\tilde{\nu}=\hat{\nu}$ on $\partial\Omega_e\setminus\Gamma_D$ in the local problems and the unique value of the hybrid variables $\hat{\boldsymbol{u}}$ and $\hat{\nu}$ on each face. The global problem is completed with the compatibility condition (2.8), which is written in terms of the hybrid velocity in (2.17). Note that imposing the compatibility condition (2.15) in each cell Ω_e implies that the global condition (2.17) is always verified.

$$\int_{\Gamma_D} \boldsymbol{u}_D\cdot\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega\setminus\Gamma_D} \hat{\boldsymbol{u}}\cdot\boldsymbol{n}\,d\Gamma = 0. \tag{2.17}$$

2.2.3 Integral form of the local and global problems

The integral form of the local problems, given by equations (2.13) and (2.14), for each cell is

$$\begin{cases} -\displaystyle\int_{\Omega_e}\boldsymbol{L}d\Omega = \int_{\partial\Omega_e\cap\Gamma_D}\boldsymbol{u}_D\otimes\boldsymbol{n}\,d\Gamma+\int_{\partial\Omega_e\setminus\Gamma_D}\hat{\boldsymbol{u}}\otimes\boldsymbol{n}\,d\Gamma,\\ \displaystyle\int_{\Omega_e}\frac{\partial\boldsymbol{u}}{\partial t}d\Omega+\int_{\partial\Omega_e}\left(\widehat{\boldsymbol{u}\otimes\boldsymbol{u}}\right)\boldsymbol{n}d\Gamma+\int_{\Omega_e}\boldsymbol{\nabla}\cdot\left(\frac{1+\nu_t}{Re}\boldsymbol{L}+p\mathbf{I}_{\texttt{n}_{\texttt{sd}}}\right)d\Omega\\ \qquad\displaystyle+\int_{\partial\Omega_e}\left[\left(\widehat{\frac{1+\nu_t}{Re}\boldsymbol{L}+p\mathbf{I}_{\texttt{n}_{\texttt{sd}}}}\right)-\left(\frac{1+\nu_t}{Re}\boldsymbol{L}+p\mathbf{I}_{\texttt{n}_{\texttt{sd}}}\right)\right]\boldsymbol{n}d\Gamma\\ \qquad\qquad\displaystyle-\int_{\Omega_e}\frac{1}{Re}\boldsymbol{L}^T\boldsymbol{q}_t d\Omega=\int_{\Omega_e}\boldsymbol{s}d\Omega,\\ \displaystyle\int_{\Omega_e}p\,d\Omega=|\Omega_e|\;\rho_e, \end{cases} \tag{2.18a}$$

$$
\begin{cases}
-\displaystyle\int_{\Omega_e} \tilde{\boldsymbol{q}} d\Omega = \int_{\partial\Omega_e\cap\Gamma_D} \tilde{\nu}_D \boldsymbol{n} d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D} \hat{\nu}\boldsymbol{n} d\Gamma, \\
\displaystyle\int_{\Omega_e} \frac{\partial\tilde{\nu}}{\partial t} d\Omega + \int_{\partial\Omega_e} (\widehat{\tilde{\nu}\boldsymbol{u}})\cdot\boldsymbol{n} d\Gamma + \int_{\Omega_e} \boldsymbol{\nabla}\cdot\left(\frac{1+\tilde{\nu} f_n}{\sigma Re}\tilde{\boldsymbol{q}}\right) d\Omega, \\
\qquad\qquad + \displaystyle\int_{\partial\Omega_e} \left(\widehat{\frac{1+\tilde{\nu} f_n}{\sigma Re}\tilde{\boldsymbol{q}}} - \frac{1+\tilde{\nu} f_n}{\sigma Re}\tilde{\boldsymbol{q}}\right)\cdot\boldsymbol{n} d\Gamma = \int_{\Omega_e} s d\Omega,
\end{cases}
\tag{2.18b}
$$

In the local problems, the divergence theorem is applied twice to the diffusive terms of both the momentum and SA equations. This is done to guarantee the symmetry of the global system in the case of creeping flows, as discussed in Sevilla and Huerta (2016); Giacomini et al. (2020), i.e. for Stokes problems. Furthermore, the incompressibility condition does not explicitly appear in the system (2.18) because, when written in integral form, it reduces to the compatibility condition (2.15), where no cell variable is involved. Hence, the compatibility condition in each cell yields $\mathtt{n_{el}}$ equations that are included in the global problem.

The integral form of the global problem, given by equation (2.16) and one compatibility condition (2.15) per cell, is

$$
\begin{cases}
\displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \left\{ \int_{\partial\Omega_e\setminus\partial\Omega} (\widehat{\boldsymbol{u}\otimes\boldsymbol{u}})\boldsymbol{n}\, d\Gamma + \int_{\partial\Omega_e\setminus\partial\Omega} \left(\widehat{\frac{1+\nu_t}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}}\right)\boldsymbol{n}\, d\Gamma \right. \\
\qquad\qquad \left. + \displaystyle\int_{\partial\Omega_e\cap\partial\Omega\setminus\Gamma_D} \boldsymbol{B}(\boldsymbol{u}, p, \boldsymbol{L}, \tilde{\nu}, \boldsymbol{g}) d\Gamma \right\} = \mathbf{0} \\
\displaystyle\int_{\partial\Omega_e\setminus\Gamma_D} \hat{\boldsymbol{u}}\cdot\boldsymbol{n}\, d\Gamma = -\int_{\partial\Omega_e\cap\Gamma_D} \boldsymbol{u}_D\cdot\boldsymbol{n}\, d\Gamma \qquad \text{for } e = 1,\dots,\mathtt{n_{el}} \\
\displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \left\{ \int_{\partial\Omega_e\setminus\partial\Omega} (\widehat{\tilde{\nu}\boldsymbol{u}})\cdot\boldsymbol{n}\, d\Gamma + \int_{\partial\Omega_e\setminus\partial\Omega} \left(\widehat{\frac{1+\tilde{\nu} f_n}{\sigma Re}\tilde{\boldsymbol{q}}}\right)\cdot\boldsymbol{n}\, d\Gamma \right. \\
\qquad\qquad \left. + \displaystyle\int_{\partial\Omega_e\cap\partial\Omega\setminus\Gamma_D} b(\boldsymbol{u}, \tilde{\nu}, \tilde{\boldsymbol{q}}) d\Gamma \right\} = 0
\end{cases}
\tag{2.19}
$$

The numerical trace of the convective and diffusive fluxes in the RANS equations is given by

$$
(\widehat{\boldsymbol{u}\otimes\boldsymbol{u}})\boldsymbol{n} := \begin{cases}
(\boldsymbol{u}_D\otimes\boldsymbol{u}_D)\boldsymbol{n} + \boldsymbol{\tau}^a(\boldsymbol{u}-\boldsymbol{u}_D) & \text{on } \partial\Omega_e\cap\Gamma_D, \\
(\hat{\boldsymbol{u}}\otimes\hat{\boldsymbol{u}})\boldsymbol{n} + \boldsymbol{\tau}^a(\boldsymbol{u}-\hat{\boldsymbol{u}}) & \text{elsewhere,}
\end{cases}
\tag{2.20a}
$$

and

$$
\left(\widehat{\frac{1+\nu_t}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}}\right)\boldsymbol{n} := \begin{cases}
\left(\dfrac{1+\tilde{\nu}_D f_{v1}(\tilde{\nu}_D)}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} + \boldsymbol{\tau}^d(\boldsymbol{u}-\boldsymbol{u}_D) \text{on } \partial\Omega_e\cap\Gamma_D, \\
\left(\dfrac{1+\hat{\nu} f_{v1}(\hat{\nu})}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} + \boldsymbol{\tau}^d(\boldsymbol{u}-\hat{\boldsymbol{u}}) \qquad \text{elsewhere,}
\end{cases}
\tag{2.20b}
$$

respectively. Similarly, the numerical trace of the convective and diffusive fluxes in the SA equation is given by

$$\widehat{(\tilde{\nu}\boldsymbol{u})} \cdot \boldsymbol{n} := \begin{cases} (\tilde{\nu}_D \boldsymbol{u}_D) \cdot \boldsymbol{n} + \tilde{\tau}^a(\tilde{\nu} - \tilde{\nu}_D) & \text{on } \partial\Omega_e \cap \Gamma_D, \\ (\hat{\nu}\hat{\boldsymbol{u}}) \cdot \boldsymbol{n} + \tilde{\tau}^a(\tilde{\nu} - \hat{\nu}) & \text{elsewhere.} \end{cases} \tag{2.21a}$$

and

$$\left(\widehat{\frac{1+\tilde{\nu} f_n}{\sigma Re}}\, \tilde{\boldsymbol{q}}\right) \cdot \boldsymbol{n} := \begin{cases} \left(\dfrac{1+\tilde{\nu}_D f_n(\tilde{\nu}_D)}{\sigma Re}\, \tilde{\boldsymbol{q}}\right) \cdot \boldsymbol{n} + \tilde{\tau}^d(\tilde{\nu} - \tilde{\nu}_D) \text{ on } \partial\Omega_e \cap \Gamma_D, \\ \left(\dfrac{1+\hat{\nu} f_n(\hat{\nu})}{\sigma Re}\, \tilde{\boldsymbol{q}}\right) \cdot \boldsymbol{n} + \tilde{\tau}^d(\tilde{\nu} - \hat{\nu}) \qquad \text{elsewhere,} \end{cases} \tag{2.21b}$$

respectively.

An appropriate stabilisation, given by $\boldsymbol{\tau}^a$, $\boldsymbol{\tau}^d$, $\tilde{\tau}^a$, and $\tilde{\tau}^d$, is required to ensure that the numerical scheme is well-posed. The effect of the stabilisation has been extensively studied in the literature for HDG by Cockburn et al. (2009, 2008); Nguyen et al. (2010); Sevilla et al. (2018); Giacomini et al. (2018); Vila-Pérez et al. (2021) and FCFV by Sevilla et al. (2018, 2019); Vieira et al. (2020); Giacomini and Sevilla (2020); Vila-Pérez et al. (2022). The diffusive stabilisation is taken as

$$\boldsymbol{\tau}^d := \frac{\beta\, (1+\hat{\nu} f_{v1}(\hat{\nu}))}{Re} \mathbf{I}_{\mathtt{n_{sd}}} \quad \text{and} \quad \tilde{\tau}^d := \frac{\beta\, (1+\hat{\nu} f_n(\hat{\nu}))}{\sigma Re}, \tag{2.22}$$

where β is a numerical parameter.

For the convective stabilisation, new definitions are proposed in this work. The derivation of three different convective stabilisations, based on the Lax-Friedrichs (LF), Roe, and Harten-Lax-van Leer (HLL) Riemann solvers, is detailed in Appendix A and follows the ideas presented in Vila-Pérez et al. (2021, 2022); Vila-Pérez et al. (2023) for compressible flows.

After introducing the expression of the numerical fluxes given by equations (2.20) and (2.21) into the local problem (2.18), the integral form of the coupled RANS and SA equations is

$$\begin{cases} -\displaystyle\int_{\Omega_e} \boldsymbol{L}\, d\Omega = \int_{\partial\Omega_e \cap \Gamma_D} \boldsymbol{u}_D \otimes \boldsymbol{n}\, d\Gamma + \int_{\partial\Omega_e \setminus \Gamma_D} \hat{\boldsymbol{u}} \otimes \boldsymbol{n}\, d\Gamma \\ \displaystyle\int_{\Omega_e} \frac{\partial \boldsymbol{u}}{\partial t}\, d\Omega + \int_{\partial\Omega_e} \boldsymbol{\tau} \boldsymbol{u}\, d\Gamma + \int_{\Omega_e} \boldsymbol{\nabla}\cdot \left(\frac{1+\nu_t}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}}\right) d\Omega - \int_{\Omega_e} \frac{1}{Re} \boldsymbol{L}^T \boldsymbol{q}_t d\Omega \\ \qquad = \displaystyle\int_{\Omega_e} \boldsymbol{s}\, d\Omega + \int_{\partial\Omega_e \cap \Gamma_D} \left(\boldsymbol{\tau} - (\boldsymbol{u}_D \cdot \boldsymbol{n}) \mathbf{I}_{\mathtt{n_{sd}}}\right) \boldsymbol{u}_D\, d\Gamma + \int_{\partial\Omega_e \setminus \Gamma_D} \left(\boldsymbol{\tau} - (\hat{\boldsymbol{u}} \cdot \boldsymbol{n}) \mathbf{I}_{\mathtt{n_{sd}}}\right) \hat{\boldsymbol{u}}\, d\Gamma \\ \displaystyle\int_{\Omega_e} p\, d\Omega = |\Omega_e|\ \rho_e \end{cases} \tag{2.23a}$$

$$\begin{cases} -\displaystyle\int_{\Omega_e} \tilde{\boldsymbol{q}}\, d\Omega = \int_{\partial\Omega_e\cap\Gamma_D} \tilde{\nu}_D \boldsymbol{n}\, d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D} \hat{\nu}\boldsymbol{n}\, d\Gamma \\ \displaystyle\int_{\Omega_e} \frac{\partial\tilde{\nu}}{\partial t}\, d\Omega + \int_{\partial\Omega_e} \tilde{\tau}\tilde{\nu}\, d\Gamma + \int_{\Omega_e} \frac{1}{\sigma Re} \boldsymbol{\nabla}\cdot \left((1+\tilde{\nu} f_n)\tilde{\boldsymbol{q}}\right)\, d\Omega = \int_{\Omega_e} s\, d\Omega \\ \qquad\qquad + \displaystyle\int_{\partial\Omega_e\cap\Gamma_D} (\tilde{\tau} - \boldsymbol{u}_D\cdot\boldsymbol{n})\tilde{\nu}_D\, d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D} (\tilde{\tau} - \hat{\boldsymbol{u}}\cdot\boldsymbol{n})\hat{\nu}\, d\Gamma, \end{cases} \tag{2.23b}$$

where $\boldsymbol{\tau} = \boldsymbol{\tau}^a + \boldsymbol{\tau}^d$ and $\tilde{\tau} = \tilde{\tau}^a + \tilde{\tau}^d$.

Similarly, introducing the numerical fluxes into the integral form of equation (2.19), the global problem is obtained

$$\begin{cases} \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Bigg\{ \int_{\partial\Omega_e\setminus\partial\Omega} \boldsymbol{\tau}\boldsymbol{u}\, d\Gamma + \int_{\partial\Omega_e\setminus\partial\Omega} \left(\frac{1+\hat{\nu} f_{v1}(\hat{\nu})}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} d\Gamma - \int_{\partial\Omega_e\setminus\partial\Omega} \boldsymbol{\tau}\hat{\boldsymbol{u}}\, d\Gamma \\ \qquad + \displaystyle\int_{\partial\Omega_e\cap\partial\Omega\setminus\Gamma_D} \hat{\boldsymbol{B}}(\boldsymbol{u}, p, \boldsymbol{L}, \hat{\boldsymbol{u}}, \hat{\nu}, \boldsymbol{\tau}, \boldsymbol{g})\, d\Gamma \Bigg\} = \boldsymbol{0} \\ \displaystyle\int_{\partial\Omega_e\setminus\Gamma_D} \hat{\boldsymbol{u}}\cdot\boldsymbol{n}\, d\Gamma = -\int_{\partial\Omega_e\cap\Gamma_D} \boldsymbol{u}_D\cdot\boldsymbol{n}\, d\Gamma \qquad \text{for } e = 1,\dots,\mathtt{n_{el}} \\ \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Bigg\{ \int_{\partial\Omega_e\setminus\partial\Omega} \tilde{\tau}\tilde{\nu}\, d\Gamma + \int_{\partial\Omega_e\setminus\partial\Omega} \left(\frac{1+\hat{\nu} f_n(\hat{\nu})}{\sigma Re}\tilde{\boldsymbol{q}}\right)\boldsymbol{n}\, d\Gamma - \int_{\partial\Omega_e\setminus\partial\Omega} \tilde{\tau}\hat{\nu}\, d\Gamma \\ \qquad + \displaystyle\int_{\partial\Omega_e\cap\partial\Omega\setminus\Gamma_D} \hat{b}(\tilde{\nu}, \tilde{\boldsymbol{q}}, \hat{\boldsymbol{u}}, \hat{\nu}, \tilde{\tau})\, d\Gamma \Bigg\} = 0. \end{cases} \tag{2.24}$$

It is worth noting that the first term of the convective fluxes in equations (2.20a) and (2.21a) does not appear on internal faces within the global problem (2.24) due to the unique definition of the hybrid variables $\hat{\boldsymbol{u}}$ and $\hat{\nu}$ on internal faces, nonetheless, the contribution of those terms is present on the boundary operators $\hat{\boldsymbol{B}}$ and $\hat{b}$, defined in Eq. (2.25) for each boundary type.

The FCFV version of the boundary operators appearing in the global problem is given by

$$\hat{\boldsymbol{B}} = \begin{cases} \left(\frac{1+\hat{\nu} f_{v1}(\hat{\nu})}{Re}\boldsymbol{L} + \hat{\boldsymbol{u}}\otimes\hat{\boldsymbol{u}} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} + \boldsymbol{\tau}(\boldsymbol{u} - \hat{\boldsymbol{u}}) + \boldsymbol{g} & \text{on } \Gamma_N \\ \left(\frac{1+\hat{\nu} f_{v1}(\hat{\nu})}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} + \boldsymbol{\tau}^d(\boldsymbol{u} - \hat{\boldsymbol{u}}) & \text{on } \Gamma_O \\ \left\{\begin{matrix} \boldsymbol{t}_k\cdot\left[\left(\dfrac{1+\hat{\nu} f_{v1}(\hat{\nu})}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} + \boldsymbol{\tau}^d(\boldsymbol{u} - \hat{\boldsymbol{u}})\right] \\ \hat{\boldsymbol{u}}\cdot\boldsymbol{n} \end{matrix}\right\} & \text{on } \Gamma_S \end{cases} \tag{2.25a}$$

$$\hat{b} = \begin{cases} \left(\dfrac{1+\hat{\nu} f_n(\hat{\nu})}{\sigma Re}\tilde{\boldsymbol{q}} + \hat{\nu}\hat{\boldsymbol{u}}\right)\boldsymbol{n} + \tilde{\tau}(\tilde{\nu} - \hat{\nu}) & \text{on } \Gamma_N \\ \left(\dfrac{1+\hat{\nu} f_n(\hat{\nu})}{\sigma Re}\tilde{\boldsymbol{q}}\right)\boldsymbol{n} + \tilde{\tau}^d(\tilde{\nu} - \hat{\nu}) & \text{on } \Gamma_O\cup\Gamma_S \end{cases} \tag{2.25b}$$

As the Dirichlet boundary conditions are accounted for in the local problems, the FCFV boundary operators are only described for Neumann, outflow and symmetry boundaries.

2.3 Face-centred finite volume discretisation

To simplify the presentation, the set of indices for the faces of a cell is denoted by $\mathcal{A}_e := \{1, \dots, \mathtt{n}_{\mathtt{fa}}^e\}$. In addition, the set of indices for the faces of a cell on the Dirichlet boundary, Neumann Boundary, interior to the domain and on the boundary of the domain are denoted by $\mathcal{D}_e := \{j \in \mathcal{A}_e \mid \Gamma_{e,j} \cap \Gamma_D \neq \emptyset\}$, $\mathcal{N}_e := \{j \in \mathcal{A}_e \mid \Gamma_{e,j} \cap \Gamma_N \neq \emptyset\}$, $\mathcal{I}_e := \{j \in \mathcal{A}_e \mid \Gamma_{e,j} \cap \partial\Omega = \emptyset\}$ and $\mathcal{E}_e := \{j \in \mathcal{A}_e \mid \Gamma_{e,j} \cap \partial\Omega \neq \emptyset\}$, respectively. Given the imposition of Dirichlet boundary conditions in the local problems, it is also convenient to denote by $\mathcal{B}_e := \mathcal{A}_e \setminus \mathcal{D}_e$ and $\mathcal{C}_e := \mathcal{E}_e \setminus \mathcal{D}_e$ the set of indices for the faces of a cell not on a Dirichlet boundary and for the external faces not on a Dirichlet boundary. Finally, it is useful to introduce the indicator function of a set $\mathcal{S}$, i.e.

$$\chi_{\mathcal{S}}(i) = \begin{cases} 1 & \text{if } \ i \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases} \tag{2.26}$$

2.3.1 Spatial discretisation

The FCFV method introduces a constant approximation of the primal, $\boldsymbol{u}$, p, and $\tilde{\nu}$, and mixed, $\boldsymbol{L}$ and $\tilde{\boldsymbol{q}}$, variables in each cell as well as a constant approximation of the hybrid variables, $\hat{\boldsymbol{u}}$ and $\hat{\nu}$ on each cell face. The value of the primal and mixed variables in each cell is denoted by $\mathbf{u}_e$, p_e, $\tilde{\nu}_e$, $\mathbf{L}_e$, and $\tilde{\mathbf{q}}_e$ respectively, whereas the value of the hybrid variables on the j-th face of a cell is denoted by $\hat{\mathbf{u}}_j$ and $\hat{\nu}_j$.

With this notation, the semi-discrete form of the local problem of (2.23) reads

$$\begin{cases} -|\Omega_e|\mathbf{L}_e = \displaystyle\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|\boldsymbol{u}_{D,j}{\otimes}\boldsymbol{n}_j + \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|\hat{\mathbf{u}}_j{\otimes}\boldsymbol{n}_j \\ |\Omega_e|\dfrac{d\mathbf{u}_e}{dt} + \displaystyle\sum_{j\in\mathcal{A}_e} |\Gamma_{e,j}|\boldsymbol{\tau}_j\mathbf{u}_e - \frac{|\Omega_e|}{Re}(\mathbf{L}_e{+}\mathbf{L}_e^T)\mathbf{q}_{t,e}(\tilde{\nu}_e, \tilde{\mathbf{q}}_e) = |\Omega_e|\mathbf{s}_e \\ \qquad + \displaystyle\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j{-}(\boldsymbol{u}_{D,j}{\cdot}\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{u}_{D,j} + \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j{-}(\hat{\mathbf{u}}_j{\cdot}\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\hat{\mathbf{u}}_j \\ \mathrm{p}_e = \rho_e \end{cases} \tag{2.27a}$$

$$\begin{cases} -|\Omega_e|\tilde{\mathbf{q}}_e = \sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|\tilde{\nu}_{D,j}\boldsymbol{n}_j + \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|\hat{\nu}_j\boldsymbol{n}_j \\ |\Omega_e|\dfrac{d\tilde{\nu}_e}{dt} + \sum_{j\in\mathcal{A}_e} |\Gamma_{e,j}|\tilde{\tau}_j\tilde{\nu}_e - \dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}_e\cdot\tilde{\mathbf{q}}_e f_{n,e}(\tilde{\nu}_e) - \dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}_e\cdot\tilde{\mathbf{q}}_e\tilde{\nu}_e\dfrac{\partial f_{n,e}(\tilde{\nu}_e)}{\partial\tilde{\nu}_e} = \\ \qquad |\Omega_e|\mathrm{s}_e(\tilde{\nu}_e,\tilde{\mathbf{q}}_e,\mathbf{L}_e) + \sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|(\tilde{\tau}_j - \boldsymbol{u}_{D,j}\cdot\boldsymbol{n}_j)\tilde{\nu}_{D,j} + \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|(\tilde{\tau}_j - \hat{\mathbf{u}}_j\cdot\boldsymbol{n}_j)\hat{\nu}_j. \end{cases} \tag{2.27b}$$

where $|\Omega_e|$ is the area/volume of the cell Ω_e and $|\Gamma_{e,j}|$ the length/area of the edge/face $\Gamma_{e,j}$.

Remark 2.3. The right-hand side term s_e in the last equation of (2.27) requires the vorticity magnitude S. The vorticity can be computed from the mixed variable matrix $\mathbf{L}$ as

$$\mathbf{S}_e = (\mathbf{L}_e^T - \mathbf{L}_e)/2, \tag{2.28}$$

or in terms of the hybrid velocity $\hat{\boldsymbol{u}}$ as

$$\mathbf{S}_e = \frac{1}{|\Omega_e|}\Big(\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|(\boldsymbol{u}_{D,j}\otimes\boldsymbol{n}_j - \boldsymbol{n}_j\otimes\boldsymbol{u}_{D,j}) + \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|(\hat{\mathbf{u}}_j\otimes\boldsymbol{n}_j - \boldsymbol{n}_j\otimes\hat{\mathbf{u}}_j)\Big). \tag{2.29}$$

The two expressions are equivalent and numerical tests have shown to provide an approximation that converges linearly as the mesh is refined. The simpler expression given by equation (2.28) is employed in all the numerical examples involving turbulent flows.

Similarly, the discrete form of the global problem of of (2.24) is

$$\begin{cases} \sum_{e=1}^{\mathtt{n_{el}}} |\Gamma_{e,i}| \Big\{ \Big(\boldsymbol{\tau}_i\mathbf{u}_e + \dfrac{1+\hat{\nu}_i f_{v1}(\hat{\nu}_i)}{Re}\mathbf{L}_e\boldsymbol{n}_i + \rho_e\boldsymbol{n}_i - \boldsymbol{\tau}_i\hat{\mathbf{u}}_i\Big)\chi_{\mathcal{I}_e}(i) \\ \qquad\qquad + \hat{\mathbf{B}}_i(\mathbf{u}_e,\hat{\mathbf{u}}_i,\mathbf{L}_e,\rho_e,\hat{\nu}_i,\boldsymbol{\tau}_i,\mathbf{g}_i)\chi_{\mathcal{C}_e}(i)\Big\} = \mathbf{0} \\ \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|\hat{\mathbf{u}}_j\cdot\boldsymbol{n}_j = -\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|\boldsymbol{u}_{D,j}\cdot\boldsymbol{n}_j \quad \text{for } e = 1,\dots,\mathtt{n_{el}} \\ \sum_{e=1}^{\mathtt{n_{el}}} |\Gamma_{e,i}| \Big\{ \Big(\tilde{\tau}_i\tilde{\nu}_e + \dfrac{1+\hat{\nu}_i f_n(\hat{\nu}_i)}{\sigma Re}\tilde{\mathbf{q}}_e\cdot\boldsymbol{n}_i - \tilde{\tau}_i\hat{\nu}_i\Big)\chi_{\mathcal{I}_e}(i) \\ \qquad\qquad + \hat{\mathrm{b}}_i(\tilde{\nu}_e,\hat{\nu}_i,\tilde{\mathbf{q}}_e,\tilde{\tau}_i)\chi_{\mathcal{C}_e}(i)\Big\} = 0, \end{cases} \tag{2.30}$$

for $i \in \mathcal{B}_e$, where the FCFV discrete form of the boundary operators is given by

$$\hat{\mathbf{B}}_i := \begin{cases} \Big(\dfrac{1+\hat{\nu}_i f_{v1}(\hat{\nu}_i)}{Re}\mathbf{L}_e + \hat{\mathbf{u}}_i\otimes\hat{\mathbf{u}}_i + \rho_e\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{n}_i + \boldsymbol{\tau}_i(\mathbf{u}_e - \hat{\mathbf{u}}_i) + \mathbf{g}_i & \text{on } \Gamma_N, \\ \Big(\dfrac{1+\hat{\nu}_i f_{v1}(\hat{\nu}_i)}{Re}\mathbf{L}_e + \rho_e\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{n}_i + \boldsymbol{\tau}_i^d(\mathbf{u}_e - \hat{\mathbf{u}}_i) & \text{on } \Gamma_O, \\ \begin{Bmatrix} \mathbf{t}_{k,i}\cdot\Big[\Big(\dfrac{1+\hat{\nu}_i f_{v1}(\hat{\nu}_i)}{Re}\mathbf{L} + \rho_e\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{n}_i + \boldsymbol{\tau}_i^d(\mathbf{u}_e - \hat{\mathbf{u}}_i)\Big] \\ \hat{\mathbf{u}}_i\cdot\boldsymbol{n}_i \end{Bmatrix} & \text{on } \Gamma_S, \end{cases} \tag{2.31a}$$

and

$$\hat{\mathrm{b}}_i := \begin{cases} \left(\dfrac{1+\hat{\nu}_i f_n(\hat{\nu}_i)}{\sigma Re}\,\tilde{\mathbf{q}}_e + \tilde{\nu}_e \hat{\mathbf{u}}_i\right) \cdot \boldsymbol{n}_i + \tilde{\tau}_i(\tilde{\nu}_e - \hat{\nu}_i) & \text{on } \Gamma_N, \\ \left(\dfrac{1+\hat{\nu}_i f_n(\hat{\nu}_i)}{\sigma Re}\,\tilde{\mathbf{q}}_e\right) \cdot \boldsymbol{n}_i + \tilde{\tau}_i^d(\tilde{\nu}_e - \hat{\nu}_i) & \text{on } \Gamma_O \cup \Gamma_S. \end{cases} \tag{2.31b}$$

2.3.2 Temporal discretisation

The time integration is performed in this work using implicit multi-step backward differentiation formulae (BDF), Ascher and Petzold (1998). The first-order time derivative in the local problems is approximated as

$$\left.\frac{\partial y}{\partial t}\right|^n \approx \sum_{s=0}^{\mathtt{n_s}} a_s y^{n-s}, \tag{2.32}$$

where $y^r(\boldsymbol{x}) := y(\boldsymbol{x}, t^r)$, $\mathtt{n_s}$ is the number of stages and, to simplify the notation, the coefficients a_s include the dependence on the time step Δt. For steady-state computations pseudo-time-stepping relaxation strategies are employed, as described in Appendix B. For such cases, the first-oder BDF method (BDF1) is employed, which corresponds to $\mathtt{n_s} = 1$ with $a_0 = 1/\Delta t$ and $a_1 = -1/\Delta t$. For transient simulations the second-oder BDF method (BDF2) is employed, which corresponds to $\mathtt{n_s} = 2$ with $a_0 = 3/(2\Delta t)$, $a_1 = -2/\Delta t$ and $a_2 = 1/(2\Delta t)$.

Introducing the approximation of the first-order time derivative in (2.27), the fully discrete residuals of the local problems are obtained, namely

$$\begin{cases} \mathbf{R}_{e,L}^n := |\Omega_e|\mathbf{L}_e^n + \displaystyle\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}| \boldsymbol{u}_{D,j}^n \otimes \boldsymbol{n}_j + \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}| \hat{\mathbf{u}}_j^n \otimes \boldsymbol{n}_j, \\ \mathbf{R}_{e,u}^n := |\Omega_e| \displaystyle\sum_{s=0}^{\mathtt{n_s}} a_s \mathbf{u}_e^{n-s} + \sum_{j\in\mathcal{A}_e} |\Gamma_{e,j}| \boldsymbol{\tau}_j^{n-1} \mathbf{u}_e^n \\ \qquad -\dfrac{|\Omega_e|}{Re}(\mathbf{L}_e^n + [\mathbf{L}_e^n]^T)\mathrm{q}_{t,e}(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n) - |\Omega_e|\mathbf{s}_e^n \\ \qquad - \displaystyle\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}| \left(\boldsymbol{\tau}_j^{n-1} - (\boldsymbol{u}_{D,j}^n \cdot \boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\right) \boldsymbol{u}_{D,j}^n \\ \qquad - \displaystyle\sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}| \left(\boldsymbol{\tau}_j^{n-1} - (\hat{\mathbf{u}}_j^n \cdot \boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\right) \hat{\mathbf{u}}_j^n, \end{cases} \tag{2.33a}$$

$$\begin{cases} \mathbf{R}^n_{e,\tilde{q}} := |\Omega_e|\tilde{\mathbf{q}}^n_e + \displaystyle\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|\tilde{\nu}^n_{D,j}\boldsymbol{n}_j + \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|\hat{\nu}^n_j\boldsymbol{n}_j, \\ \mathrm{R}^n_{e,\tilde{\nu}} := |\Omega_e|\displaystyle\sum_{s=0}^{\mathtt{n_s}} a_s\tilde{\nu}^{n-s}_e + \sum_{j\in\mathcal{A}_e} |\Gamma_{e,j}|\tilde{\tau}^{n-1}_j\tilde{\nu}^n_e \\ \qquad -\dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}^n_e\cdot\tilde{\mathbf{q}}^n_e f_{n,e}(\tilde{\nu}^n_e) - \dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}^n_e\cdot\tilde{\mathbf{q}}^n_e\tilde{\nu}^n_e\dfrac{\partial f_{n,e}(\tilde{\nu}^n_e)}{\partial\tilde{\nu}^n_e} - |\Omega_e|\mathrm{s}_e(\tilde{\nu}^n_e, \tilde{\mathbf{q}}^n_e, \mathbf{L}^n_e) \\ \qquad - \displaystyle\sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|(\tilde{\tau}^{n-1}_j - \boldsymbol{u}^n_{D,j}\cdot\boldsymbol{n}_j)\tilde{\nu}^n_{D,j} - \sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|(\tilde{\tau}^{n-1}_j - \hat{\mathbf{u}}^n_j\cdot\boldsymbol{n}_j)\hat{\nu}^n_j, \end{cases} \tag{2.33b}$$

It is worth noting that, following Remark 2.3, the dependence of the SA right-hand side on $\boldsymbol{L}$, $\tilde{\nu}$ and $\tilde{\boldsymbol{q}}$ is explicitly detailed. For clarity, the dependence on the turbulent variable and its mixed counterpart is also explicitly stated in the momentum RANS equation.

Remark 2.4. The residuals in the local and global problems assume the stabilisation to be evaluated at time t^{n-1} to avoid the need to linearise the stabilisation tensors and scalars for the RANS and SA equations, respectively. This choice significantly facilitates the linearisation and, as it will be shown in the numerical examples, it does not have a noticeable impact in the stability of the time marching scheme.

The residuals of the global problem, denoted by $\mathbf{R}_{\hat{u}}$, $\mathbf{R}_\rho$ and $\mathbf{R}_{\hat{\nu}}$, are obtained from (2.30) by assembling the cell contributions of

$$\begin{cases} \mathbf{R}^n_{e,i,\hat{u}} := |\Gamma_{e,i}|\Big\{\Big(\boldsymbol{\tau}^{n-1}_i\mathbf{u}^n_e + \dfrac{1+\hat{\nu}^n_i f_{v1}(\hat{\nu}^n_i)}{Re}\mathbf{L}^n_e\boldsymbol{n}_i + \rho^n_e\boldsymbol{n}_i - \boldsymbol{\tau}^{n-1}_i\hat{\mathbf{u}}^n_i\Big)\chi_{\mathcal{I}_e}(i) \\ \qquad\qquad + \hat{\mathbf{B}}_i(\mathbf{u}^n_e, \hat{\mathbf{u}}^n_i, \mathbf{L}^n_e, \rho^n_e, \hat{\nu}^n_i, \boldsymbol{\tau}^{n-1}_i, \mathbf{g}^n_i)\chi_{\mathcal{C}_e}(i)\Big\}, \\ \mathrm{R}^n_{e,\rho} := \displaystyle\sum_{j\in\mathcal{B}_e} |\Gamma_{e,j}|\hat{\mathbf{u}}^n_j\cdot\boldsymbol{n}_j + \sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|\boldsymbol{u}^n_{D,j}\cdot\boldsymbol{n}_j, \\ \mathrm{R}^n_{e,i,\hat{\nu}} := |\Gamma_{e,i}|\Big\{\Big(\tilde{\tau}^{n-1}_i\tilde{\nu}^n_e + \dfrac{1+\hat{\nu}^n_i f_n(\hat{\nu}^n_i)}{\sigma Re}\tilde{\mathbf{q}}^n_e\cdot\boldsymbol{n}_i - \tilde{\tau}^{n-1}_i\hat{\nu}^n_i\Big)\chi_{\mathcal{I}_e}(i) \\ \qquad\qquad + \hat{\mathrm{b}}_i(\tilde{\nu}^n_e, \hat{\nu}^n_i, \tilde{\mathbf{q}}^n_e, \tilde{\tau}^{n-1}_i)\chi_{\mathcal{C}_e}(i)\Big\}. \end{cases} \tag{2.34}$$

for all $i \in \mathcal{B}_e$. It is important to note that the third equation in (2.27) is omitted in (2.33) and the value of p_e in the global residual $\mathbf{R}_{e,i,\hat{u}}$ of equation (2.34) is directly taken as ρ_e.

2.4 Face-centred finite volume solution

This work considers two numerical strategies to solve the discrete FV problem described in the previous section. The first strategy, referred to as a *monolithic* approach, consists of solving the RANS and SA equations in a fully coupled manner, whereas the second strategy, referred to as a *staggered* approach, consists of solving the RANS and the SA equations separately.

2.4.1 Monolithic solution technique

The resulting FCFV discrete problem consists of solving the nonlinear system of equations

$$\mathbf{R}(\mathbf{L}_e^n, \mathbf{u}_e^n, \dots, \mathbf{u}_e^{n-\mathtt{n_s}}, \tilde{\mathbf{q}}_e^n, \tilde{\nu}_e^n, \dots, \tilde{\nu}_e^{n-\mathtt{n_s}}, \hat{\mathbf{u}}^n, \hat{\mathbf{u}}^{n-1}, \boldsymbol{\rho}^n, \hat{\boldsymbol{\nu}}^n, \hat{\boldsymbol{\nu}}^{n-1}) = \mathbf{0}, \tag{2.35}$$

at each time step n, where the residuals are obtained by assembling the cell contributions of

$$\mathbf{R}_{e,i} := \left\{ \begin{matrix} \mathbf{R}_{e,L}(\mathbf{L}_e^n, \hat{\mathbf{u}}^n) \\ \mathbf{R}_{e,u}(\mathbf{L}_e^n, \mathbf{u}_e^n, \dots, \mathbf{u}_e^{n-\mathtt{n_s}}, \tilde{\mathbf{q}}_e^n, \tilde{\nu}_e^n, \hat{\mathbf{u}}^n, \hat{\mathbf{u}}^{n-1}, \hat{\boldsymbol{\nu}}^{n-1}) \\ \mathbf{R}_{e,\tilde{q}}(\tilde{\mathbf{q}}_e^n, \hat{\boldsymbol{\nu}}^n) \\ \mathrm{R}_{e,\tilde{\nu}}(\tilde{\mathbf{q}}_e^n, \tilde{\nu}_e^n, \dots, \tilde{\nu}_e^{n-\mathtt{n_s}}, \mathbf{L}_e^n, \hat{\mathbf{u}}^n, \hat{\mathbf{u}}^{n-1}, \hat{\boldsymbol{\nu}}^n, \hat{\boldsymbol{\nu}}^{n-1}) \\ \mathbf{R}_{e,i,\hat{u}}(\mathbf{L}_e^n, \mathbf{u}_e^n, \rho_e^n, \hat{\mathbf{u}}^n, \hat{\mathbf{u}}^{n-1}, \hat{\boldsymbol{\nu}}^n, \hat{\boldsymbol{\nu}}^{n-1}) \\ \mathrm{R}_{e,\rho}(\hat{\mathbf{u}}^n) \\ \mathbf{R}_{e,i,\hat{\nu}}(\tilde{\mathbf{q}}_e^n, \tilde{\nu}_e^n, \mathbf{u}^{n-1}, \hat{\boldsymbol{\nu}}^n, \hat{\boldsymbol{\nu}}^{n-1}) \end{matrix} \right\}, \tag{2.36}$$

for $e = 1, \dots, \mathtt{n_{el}}$ and $i \in \mathcal{B}_e$. Where the notation in Remark 2.5 is employed.

Following Remark 2.4, the dependence of the residuals on $\hat{\mathbf{u}}^{n-1}$ and $\hat{\boldsymbol{\nu}}^{n-1}$, used to compute the stabilisation of the RANS and SA equations, is explicitly stated.

The Newton-Raphson method is applied to linearise the residual of equation (2.35). After truncating the Taylor expansion at first order, the linear system to be solved at each Newton-Raphson iteration, m, can be written as

$$\begin{bmatrix} \mathbf{T}_{UU} & \mathbf{T}_{U\Lambda} \\ \mathbf{T}_{\Lambda U} & \mathbf{T}_{\Lambda\Lambda} \end{bmatrix}^{n,m} \begin{Bmatrix} \Delta\mathbf{U} \\ \Delta\boldsymbol{\Lambda} \end{Bmatrix}^{n,m} = - \begin{Bmatrix} \mathbf{R}_U \\ \mathbf{R}_\Lambda \end{Bmatrix}^{n,m}, \tag{2.37}$$

where

$$\mathbf{T}_{UU} = \begin{bmatrix} \mathbf{T}_{LL} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{uL} & \mathbf{T}_{uu} & \mathbf{T}_{u\tilde{q}} & \mathbf{T}_{u\tilde{\nu}} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_{\tilde{q}\tilde{q}} & \mathbf{0} \\ \mathbf{T}_{\tilde{\nu}L} & \mathbf{0} & \mathbf{T}_{\tilde{\nu}\tilde{q}} & \mathbf{T}_{\tilde{\nu}\tilde{\nu}} \end{bmatrix}, \quad \mathbf{T}_{U\Lambda} = \begin{bmatrix} \mathbf{T}_{L\hat{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{u\hat{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_{\tilde{q}\hat{\nu}} \\ \mathbf{T}_{\tilde{\nu}\hat{u}} & \mathbf{0} & \mathbf{T}_{\tilde{\nu}\hat{\nu}} \end{bmatrix},$$

$$\mathbf{T}_{\Lambda U} = \begin{bmatrix} \mathbf{T}_{\hat{u}L} & \mathbf{T}_{\hat{u}u} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_{\hat{\nu}\tilde{q}} & \mathbf{T}_{\hat{\nu}\tilde{\nu}} \end{bmatrix}, \quad \mathbf{T}_{\Lambda\Lambda} = \begin{bmatrix} \mathbf{T}_{\hat{u}\hat{u}} & \mathbf{T}_{\hat{u}\rho} & \mathbf{T}_{\hat{u}\hat{\nu}} \\ \mathbf{T}_{\rho\hat{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{\hat{\nu}\hat{u}} & \mathbf{0} & \mathbf{T}_{\hat{\nu}\hat{\nu}} \end{bmatrix}, \tag{2.38}$$

$$\mathbf{U} = \begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \\ \tilde{\mathbf{q}} \\ \tilde{\boldsymbol{\nu}} \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\rho} \\ \hat{\boldsymbol{\nu}} \end{Bmatrix}, \quad \mathbf{R}_U = \begin{Bmatrix} \mathbf{R}_L \\ \mathbf{R}_u \\ \mathbf{R}_{\tilde{q}} \\ \mathbf{R}_{\tilde{\nu}} \end{Bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_\rho \\ \mathbf{R}_{\hat{\nu}} \end{Bmatrix},$$

with $\mathbf{T}_{ab}$ denoting the tangent matrix obtained by assembling the cell contributions of $(\mathbf{T}_{ab})_e := \partial \mathbf{R}_{e,a}/\partial b$ and $\Delta \circledcirc^{n,m} = \circledcirc^{n,m+1} - \circledcirc^{n,m}$ the solution increment from the Newton-Raphson iteration m to $m+1$.

Given the block-diagonal structure of the matrix $\mathbf{T}_{UU}$, the linear sytem (2.37), to be solved at each Newton-Raphson iteration, can be reduced to involve only the unknowns $\boldsymbol{\Lambda}$, namely

$$\mathbb{K}^{n,m}\Delta\boldsymbol{\Lambda}^{n,m} = \mathbb{F}^{n,m}, \tag{2.39}$$

where $\mathbb{K} = \mathbf{T}_{\Lambda\Lambda} - \mathbf{T}_{\Lambda U}\mathbf{T}_{UU}^{-1}\mathbf{T}_{U\Lambda}$ and $\mathbb{F} = -\mathbf{R}_{\Lambda} + \mathbf{T}_{\Lambda U}\mathbf{T}_{UU}^{-1}\mathbf{R}_{U}$. With

$$\mathbb{K} = \begin{bmatrix} \mathbf{K}_{\hat{u}\hat{u}} & \mathbf{K}_{\hat{u}\rho} & \mathbf{K}_{\hat{u}\hat{\nu}} \\ \mathbf{K}_{\rho\hat{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{\hat{\nu}\hat{u}} & \mathbf{0} & \mathbf{K}_{\hat{\nu}\hat{\nu}} \end{bmatrix}, \mathbb{F} = \begin{Bmatrix} \mathbf{F}_{\hat{u}} \\ \mathbf{F}_{\rho} \\ \mathbf{F}_{\hat{\nu}} \end{Bmatrix}, \tag{2.40}$$

After solving the reduced system (2.39), the local problems can be used to compute the solution in each cell as

$$\mathbf{T}_{UU}^{n,m}\Delta\mathbf{U}^{n,m} = -\mathbf{R}_{U}^{n,m} - \mathbf{T}_{U\Lambda}^{n,m}\Delta\boldsymbol{\Lambda}^{n,m}. \tag{2.41}$$

It is worth emphasizing that the block-diagonal structure of the matrix $\mathbf{T}_{UU}$ enables the solution of a small system of equations, cell-by-cell. The blocks composing the matrices $\mathbf{T}_{UU}$, $\mathbf{T}_{U\Lambda}$, $\mathbf{T}_{\Lambda U}$, and $\mathbf{T}_{\Lambda\Lambda}$ arising from the exact Newton-Raphson linearization, and also the blocks of $\mathbb{K}$ and $\mathbb{F}$, after the local degrees of freedom elimination, are detailed in Appendix C, Section C.1.

Remark 2.5. To simplify the presentation, an abuse of notation is introduced in equation (2.36), where $\mathbf{R}_{e,L}$ and $\mathbf{L}_e^n$ denote the vectorised version of the matrices appearing in equations (2.33) and (2.34).

2.4.2 Staggered solution technique

The non-linear problem of Eq. (2.35) can be solved in a staggered fashion by splitting the monolithic approach of Section 2.4.1 into two non-linear problems, to handle the RANS and SA equations separately. This induces the following non-linear problem for the RANS equations, where at each time step n, the residuals are obtained by assembling the cell contributions of

$$\mathbf{R}_{e,i}^{\mathtt{RANS}} := \begin{Bmatrix} \mathbf{R}_{e,L}(\mathbf{L}_e^n, \hat{\mathbf{u}}^n) \\ \mathbf{R}_{e,u}(\mathbf{L}_e^n, \mathbf{u}_e^n, \dots, \mathbf{u}_e^{n-\mathtt{n_s}}, \tilde{\mathbf{q}}_e^n, \tilde{\nu}_e^n, \hat{\mathbf{u}}^n, \hat{\mathbf{u}}^{n-1}, \hat{\boldsymbol{\nu}}^{n-1}) \\ \mathbf{R}_{e,i,\hat{u}}(\mathbf{L}_e^n, \mathbf{u}_e^n, \rho_e^n, \hat{\mathbf{u}}^n, \hat{\mathbf{u}}^{n-1}, \hat{\boldsymbol{\nu}}^n, \hat{\boldsymbol{\nu}}^{n-1}) \\ \mathrm{R}_{e,\rho}(\hat{\mathbf{u}}^n) \end{Bmatrix}, \tag{2.42}$$

while for the SA equations, the non-linear residuals at each time step are obtained by assembling the cell contributions of

$$\mathbf{R}_{e,i}^{\mathtt{SA}} := \begin{Bmatrix} \mathbf{R}_{e,\tilde{q}}(\tilde{\mathbf{q}}_e^n, \hat{\boldsymbol{\nu}}^n) \\ \mathrm{R}_{e,\tilde{\nu}}(\tilde{\mathbf{q}}_e^n, \tilde{\nu}_e^n, \ldots, \tilde{\nu}_e^{n-\mathtt{n_s}}, \mathbf{L}_e^n, \hat{\mathbf{u}}^n, \hat{\mathbf{u}}^{n-1}, \hat{\boldsymbol{\nu}}^n, \hat{\boldsymbol{\nu}}^{n-1}) \\ \mathbf{R}_{e,i,\hat{\nu}}(\tilde{\mathbf{q}}_e^n, \tilde{\nu}_e^n, \hat{\mathbf{u}}^{n-1}, \hat{\boldsymbol{\nu}}^n, \hat{\boldsymbol{\nu}}^{n-1}) \end{Bmatrix}, \tag{2.43}$$

The staggered solution of both problems requires solving a global and a local problem to obtain $\hat{\mathbf{u}}$, $\boldsymbol{\rho}$, $\mathbf{L}$, and $\mathbf{u}$ followed by a global and a local problem to obtain $\hat{\boldsymbol{\nu}}$,$\tilde{\mathbf{q}}$, and $\tilde{\boldsymbol{\nu}}$. This is then repeated until convergence. The two linear systems, for RANS and SA, to be solved at each Newton-Raphson iteration m and staggered iteration k reads

$$\begin{bmatrix} \mathbf{T}_{UU} & \mathbf{T}_{U\Lambda} \\ \mathbf{T}_{\Lambda U} & \mathbf{T}_{\Lambda\Lambda} \end{bmatrix}^{n,m,k} \begin{Bmatrix} \Delta\mathbf{U} \\ \Delta\boldsymbol{\Lambda} \end{Bmatrix}^{n,m,k} = - \begin{Bmatrix} \mathbf{R}_U \\ \mathbf{R}_\Lambda \end{Bmatrix}^{n,m,k}, \tag{2.44}$$

where, for the RANS equations

$$\begin{gathered} \mathbf{T}_{UU} = \begin{bmatrix} \mathbf{T}_{LL} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{uu} \end{bmatrix}, \qquad \mathbf{T}_{U\Lambda} = \begin{bmatrix} \mathbf{T}_{L\hat{u}} & \mathbf{0} \\ \mathbf{T}_{u\hat{u}} & \mathbf{0} \end{bmatrix}, \\ \mathbf{T}_{\Lambda U} = \begin{bmatrix} \mathbf{T}_{\hat{u}L} & \mathbf{T}_{\hat{u}u} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad \mathbf{T}_{\Lambda\Lambda} = \begin{bmatrix} \mathbf{T}_{\hat{u}\hat{u}} & \mathbf{T}_{\hat{u}\rho} \\ \mathbf{T}_{\rho\hat{u}} & \mathbf{0} \end{bmatrix}, \\ \mathbf{U} = \begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\rho} \end{Bmatrix}, \quad \mathbf{R}_U = \begin{Bmatrix} \mathbf{R}_L \\ \mathbf{R}_u \end{Bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_\rho \end{Bmatrix}, \end{gathered} \tag{2.45}$$

while for the SA equations

$$\begin{gathered} \mathbf{T}_{UU} = \begin{bmatrix} \mathbf{T}_{\tilde{q}\tilde{q}} & \mathbf{0} \\ \mathbf{T}_{\tilde{\nu}\tilde{q}} & \mathbf{T}_{\tilde{\nu}\tilde{\nu}} \end{bmatrix}, \qquad \mathbf{T}_{U\Lambda} = \begin{bmatrix} \mathbf{T}_{\tilde{q}\hat{\nu}} \\ \mathbf{T}_{\tilde{\nu}\hat{\nu}} \end{bmatrix}, \\ \mathbf{T}_{\Lambda U} = \begin{bmatrix} \mathbf{T}_{\hat{\nu}\tilde{q}} & \mathbf{T}_{\hat{\nu}\tilde{\nu}} \end{bmatrix}, \qquad \mathbf{T}_{\Lambda\Lambda} = \begin{bmatrix} \mathbf{T}_{\hat{\nu}\hat{\nu}} \end{bmatrix}, \\ \mathbf{U} = \begin{Bmatrix} \tilde{\mathbf{q}} \\ \tilde{\boldsymbol{\nu}} \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\boldsymbol{\nu}} \end{Bmatrix}, \quad \mathbf{R}_U = \begin{Bmatrix} \mathbf{R}_{\tilde{q}} \\ \mathbf{R}_{\tilde{\nu}} \end{Bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{\nu}} \end{Bmatrix}, \end{gathered} \tag{2.46}$$

As done for the Monolithic solver, the system of equations of (2.44) can be rewritten only in terms of the global variables by eliminating the local degrees of freedom, leading to a linear system of equations of the form (2.39) for both RANS and SA equations. Later, the RANS and SA local problems can be used to compute the solution in each cell as in Eq. (2.41).

It is worth emphasizing that in the staggered case the local residuals of the RANS equations, namely $\mathbf{R}_L$, and $\mathbf{R}_u$ are linear in terms of the local variables $\boldsymbol{L}$, and $\boldsymbol{u}$, allowing the elimination of $\mathbf{R}_L$, and $\mathbf{R}_u$ from the non-linear problem in (2.42). This leads to a

Newton-Rapshon solver only in terms of global variables and the local solution becomes just a postprocess step. This aspect is further explored in Section 2.5.1, where the FCFV solution is particularized to the solution of the incompressible Navier-Stokes equations in a laminar setting.

2.5 A particular case: Laminar flows

The FCFV derivation outlined in Section 2.2 takes into account the RANS equations coupled with the Sparlart-Allmaras turbulence model, thus the velocity flow field $\boldsymbol{u}$ is the average one arising from the Reynolds averaging, as explained in Remark 2.1.

The formulation can be particularized for laminar flows by setting the turbulence viscosity contribution to zero through $\tilde{\nu} = 0$, and under the assumption that the vector field $\boldsymbol{u}$ now denotes the instantaneous flow velocity and not the average one as in RANS. This reduces the system of equations in (2.1) to the Navier-Stokes equations. Again, considering an open bounded domain $\Omega \in \mathbb{R}^{\mathtt{n_{sd}}}$, where $\mathtt{n_{sd}}$ is the number of spatial dimensions, and a time interval $(0, T)$, with $T > 0$, non-dimensional Navier-Stokes equations reads

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot\left(\dfrac{1}{Re}\boldsymbol{\nabla}\boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}}\right) = \boldsymbol{s} & \text{in } \Omega\times(0,T], \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega\times(0,T], \end{cases} \tag{2.47}$$

where the pointwise divergence-free condition on velocity is explored, as in Remark 2.2, to remove the term $1/Re\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla}^T\boldsymbol{u})$. Under the assumptions above the FCFV formulation for laminar flows follows the same steps described in Section 2.2. In fact, with the abuse of notation regarding $\boldsymbol{u}$, the laminar formulation is only a particular case of the RANS formulation when $\nu_t = 0$ and no turbulence modelling is involved. After following the FCFV formulation steps of Section 2.2, the integral form of the Navier-Stokes equations reads

$$\begin{cases} -\displaystyle\int_{\Omega_e} \boldsymbol{L} d\Omega = \int_{\partial\Omega_e\cap\Gamma_D} \boldsymbol{u}_D\otimes\boldsymbol{n}\, d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D} \hat{\boldsymbol{u}}\otimes\boldsymbol{n}\, d\Gamma \\ \displaystyle\int_{\Omega_e} \frac{\partial \boldsymbol{u}}{\partial t} d\Omega + \int_{\partial\Omega_e} \boldsymbol{\tau}\boldsymbol{u} d\Gamma = \int_{\Omega_e} \boldsymbol{s} d\Omega \\ \qquad\qquad + \displaystyle\int_{\partial\Omega_e\cap\Gamma_D} \left(\boldsymbol{\tau} - (\boldsymbol{u}_D\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{u}_D d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D} \left(\boldsymbol{\tau} - (\hat{\boldsymbol{u}}\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}}\right)\hat{\boldsymbol{u}} d\Gamma \\ \displaystyle\int_{\Omega_e} p\, d\Omega = |\Omega_e|\ \rho_e, \end{cases} \tag{2.48}$$

Similarly, the integral form of the global problem is obtained

$$
\left\{
\begin{aligned}
&\sum_{e=1}^{\mathtt{n_{el}}} \left\{ \int_{\partial\Omega_e \setminus \partial\Omega} \boldsymbol{\tau} \boldsymbol{u} d\Gamma + \int_{\partial\Omega_e \setminus \partial\Omega} \left(\frac{1}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} d\Gamma - \int_{\partial\Omega_e \setminus \partial\Omega} \boldsymbol{\tau} \hat{\boldsymbol{u}} d\Gamma \right. \\
&\qquad \left. + \int_{\partial\Omega_e \cap \partial\Omega \setminus \Gamma_D} \hat{\boldsymbol{B}}(\boldsymbol{u}, p, \boldsymbol{L}, \hat{\boldsymbol{u}}, \boldsymbol{\tau}, \boldsymbol{g}) d\Gamma \right\} = \mathbf{0} \\
&\int_{\partial\Omega_e \setminus \Gamma_D} \hat{\boldsymbol{u}} \cdot \boldsymbol{n} \, d\Gamma = - \int_{\partial\Omega_e \cap \Gamma_D} \boldsymbol{u}_D \cdot \boldsymbol{n} \, d\Gamma \qquad \text{for } e = 1, \dots, \mathtt{n_{el}}
\end{aligned}
\right.
\tag{2.49}
$$

with the numerical trace of the diffusive flux being reduced to

$$
\widehat{\left(\frac{1}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right)} \boldsymbol{n} := \begin{cases} \left(\frac{1}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau}^d (\boldsymbol{u} - \boldsymbol{u}_D) \text{on } \partial\Omega_e \cap \Gamma_D, \\ \left(\frac{1}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau}^d (\boldsymbol{u} - \hat{\boldsymbol{u}}) \text{ elsewhere,} \end{cases}
\tag{2.50}
$$

where $\boldsymbol{\tau} = \boldsymbol{\tau}^a + \boldsymbol{\tau}^d$. The diffusive $\boldsymbol{\tau}^d$ stabilisation reduces to

$$
\boldsymbol{\tau}^d := \frac{\beta}{Re} \mathbf{I}_{\mathtt{n_{sd}}}
\tag{2.51}
$$

The numerical trace of the convective flux as well as the convective stabilization $\boldsymbol{\tau}^a$ remains unchanged with respect to the one used for the RANS equations, see (2.20a), with the latter being described in Appendix A. The FCFV boundary operator $\hat{\boldsymbol{B}}$ reduces to

$$
\hat{\boldsymbol{B}} := \begin{cases} \left(\frac{1}{Re} \boldsymbol{L} + \hat{\boldsymbol{u}} \otimes \hat{\boldsymbol{u}} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau} (\boldsymbol{u} - \hat{\boldsymbol{u}}) + \boldsymbol{g} & \text{on } \Gamma_N \\ \left(\frac{1}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau}^d (\boldsymbol{u} - \hat{\boldsymbol{u}}) & \text{on } \Gamma_O \\ \left\{ \begin{matrix} \boldsymbol{t}_k \cdot \left[\left(\frac{1}{Re} \boldsymbol{L} + p \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau}^d (\boldsymbol{u} - \hat{\boldsymbol{u}}) \right] \\ \hat{\boldsymbol{u}} \cdot \boldsymbol{n} \end{matrix} \right\} & \text{on } \Gamma_S \end{cases}
\tag{2.52}
$$

Eq. 2.48, and Eq. 2.49 sets up the FCFV formulation for laminar flows where, $\mathtt{n_{el}}$ *local problems* are defined, one per cell, to express the primal variables, $\boldsymbol{u}$, and p, and the mixed variable, $\boldsymbol{L}$, as a function of the new independent variable, $\hat{\boldsymbol{u}}$ on the boundary of the cell.

2.5.1 Face-centred finite volume discretisation and solution

After employing the same notation and discretisation strategy described in Section 2.3, we arrive at the following set of discrete equations for the local problem, namely

$$\begin{cases} -|\Omega_e|\mathbf{L}_e^n = \displaystyle\sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\boldsymbol{u}_{D,j}^n\otimes\boldsymbol{n}_j + \sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\hat{\mathbf{u}}_j^n\otimes\boldsymbol{n}_j, \\ |\Omega_e|\displaystyle\sum_{s=0}^{\mathtt{n_s}}a_s\mathbf{u}_e^{n-s}+\sum_{j\in\mathcal{A}_e}|\Gamma_{e,j}|\boldsymbol{\tau}_j^{n-1}\mathbf{u}_e^n = |\Omega_e|\mathbf{s}_e^n \\ -\displaystyle\sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j^{n-1}+(\boldsymbol{u}_{D,j}^n\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{u}_{D,j}^n-\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j^{n-1}+(\hat{\mathbf{u}}_j^n\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\hat{\mathbf{u}}_j^n, \end{cases} \tag{2.53}$$

On the other hand, the residuals of the global problem, $\mathbf{R}_{\hat{u}}$, and $\mathbf{R}_\rho$ are obtained by assembling the following cell contributions

$$\begin{cases} \mathbf{R}_{e,i,\hat{u}}^n := |\Gamma_{e,i}|\Big\{\Big(\boldsymbol{\tau}_i^{n-1}\mathbf{u}_e^n + \dfrac{1}{Re}\mathbf{L}_e^n\boldsymbol{n}_i + \rho_e^n\boldsymbol{n}_i - \boldsymbol{\tau}_i^{n-1}\hat{\mathbf{u}}_i^n\Big)\chi_{\mathcal{I}_e}(i) \\ \qquad\qquad + \hat{\mathbf{B}}_i(\mathbf{u}_e^n,\hat{\mathbf{u}}_i^n,\mathbf{L}_e^n,\rho_e^n,\boldsymbol{\tau}_i^{n-1},\mathbf{g}_i)\chi_{\mathcal{C}_e}(i)\Big\}, \\ \mathrm{R}_{e,\rho}^n := \displaystyle\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\hat{\mathbf{u}}_j^n\cdot\boldsymbol{n}_j + \sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\boldsymbol{u}_{D,j}^n\cdot\boldsymbol{n}_j, \end{cases} \tag{2.54}$$

for all $i\in\mathcal{B}_e$. Here we note that the local discrete form in (2.53) is linear concerning the local variables $\boldsymbol{L}$, and $\boldsymbol{u}$, removing the need for a non-linear Newton-Raphson solver at the local problem. This allows the reduction of the global residuals in (2.54) to involve only the global unknowns $\hat{\mathbf{u}}$, and $\boldsymbol{\rho}$, which is done by plugging the explicit expressions for $\boldsymbol{u}$, and $\boldsymbol{L}$ of (2.53), into the global residuals expressions (2.54). This leads to a set of residuals to be minimized only in terms of global variables as

$$\begin{cases} \mathbf{R}_{e,i,\hat{u}}^n := |\Gamma_{e,i}|\Big\{\Big(\boldsymbol{\tau}_i^{n-1}[\boldsymbol{\alpha}_e(\boldsymbol{\tau}_j^{n-1})]^{-1}\displaystyle\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j^{n-1}-(\hat{\mathbf{u}}_j^n\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\hat{\mathbf{u}}_j^n \\ \qquad + \boldsymbol{\tau}_i^{n-1}[\boldsymbol{\alpha}_e(\boldsymbol{\tau}_j^{n-1})]^{-1}\displaystyle\sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j^{n-1}-(\boldsymbol{u}_{D,j}^n\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{u}_{D,j}^n \\ \qquad + \boldsymbol{\tau}_i^{n-1}[\boldsymbol{\alpha}_e(\boldsymbol{\tau}_j^{n-1})]^{-1}|\Omega_e|\displaystyle\sum_{s=1}^{\mathtt{n_s}}a_s\mathbf{u}_e^{n-s} + \boldsymbol{\tau}_i^{n-1}[\boldsymbol{\alpha}_e(\boldsymbol{\tau}_j^{n-1})]^{-1}|\Omega_e|\mathbf{s}_e^n \\ \qquad - \dfrac{|\Omega_e|^{-1}}{Re}\displaystyle\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|(\boldsymbol{n}_i\cdot\boldsymbol{n}_j)\hat{\mathbf{u}}_j^n - \dfrac{|\Omega_e|^{-1}}{Re}\sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|(\boldsymbol{n}_i\cdot\boldsymbol{n}_j)\boldsymbol{u}_{D,j}^n \\ \qquad + \rho_e^n\boldsymbol{n}_i - \boldsymbol{\tau}_i^{n-1}\hat{\mathbf{u}}_i^n\Big)\chi_{\mathcal{I}_e}(i) + \hat{\mathbf{B}}_i(\hat{\mathbf{u}}_i^n,\rho_e^n,\boldsymbol{\tau}_i^{n-1},\boldsymbol{\tau}_j^{n-1},\mathbf{g}_i^n)\chi_{\mathcal{C}_e}(i)\Big\}, \\ \mathrm{R}_{e,\rho}^n := \displaystyle\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\hat{\mathbf{u}}_j^n\cdot\boldsymbol{n}_j + \sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\boldsymbol{u}_{D,j}^n\cdot\boldsymbol{n}_j, \end{cases} \tag{2.55}$$

for all $i \in \mathcal{B}_e$. With $\boldsymbol{\alpha}_e$ being defined below. The constant a_0 is the first coefficient of the corresponding time marching scheme, as described in Section 2.3.2.

$$\boldsymbol{\alpha}_e = |\Omega_e| a_0 + \sum_{j \in \mathcal{A}_e} |\Gamma_{e,j}| \boldsymbol{\tau}_j^{n-1}$$

The same elimination of local degrees of freedom is applied to the boundary operator $\hat{\mathbf{B}}_i$, as explicitly stated in (2.55). With the residuals written only in terms of the global degrees of freedom, the linearization is performed, leading the linear system of equations to be solved at each Newton-Raphson iteration m

$$\mathbb{K}^{n,m} \Delta \boldsymbol{\Lambda} = -\mathbf{R}_\Lambda^{n,m}, \tag{2.56}$$

where

$$\mathbb{K}^{n,m} = \begin{bmatrix} \mathbf{K}_{\hat{u}\hat{u}} & \mathbf{K}_{\hat{u}\rho} \\ \mathbf{K}_{\rho\hat{u}} & \mathbf{0} \end{bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_\rho \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\rho} \end{Bmatrix}, \quad \mathbf{U} \ = \begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \end{Bmatrix}, \tag{2.57}$$

with $\mathbf{K}_{ab}$ denoting the tangent matrix obtained by assembling the cell contributions of $(\mathbf{K}_{ab})_e := \partial \mathbf{R}_{e,a}/\partial b$ and $\Delta \circledcirc^{n,m} = \circledcirc^{n,m+1} - \circledcirc^{n,m}$ the solution increment from the Newton-Raphson iteration m to $m+1$. The local solution $\mathbf{U}^n$ is retrieved as a post-process after obtaining the global solution $\boldsymbol{\Lambda}^n$ at the end of the Newton-Raphson solver. The blocks composing the matrix $\mathbb{K}$, arising from Newton-Raphson linearisation, are detailed in Appendix C,Section C.2.

2.6 Computational aspects

This section briefly discusses some computational aspects to be considered when implementing the proposed FCFV for solving the incompressible RANS equations with the SA turbulence model.

2.6.1 Size of local and global problems

The cost of each Newton-Raphson iteration is dominated by the cost of solving the global problem. The size of the sparse non-symmetric linear system of equations (2.39) is $\mathtt{n_{eq}} = \mathtt{n_{fa}}(\mathtt{n_{sd}} + 1) + \mathtt{n_{el}}$ for the Monolithic case, where $\mathtt{n_{fa}}$ is the total number of edges/faces in the mesh and $\mathtt{n_{el}}$ the total number of elements in the mesh. The staggered approach requires solving two global systems, associated with the RANS and SA equations, of size $\mathtt{n_{eq}^{RANS}} = \mathtt{n_{fa}}\mathtt{n_{sd}} + \mathtt{n_{el}}$ and $\mathtt{n_{eq}^{SA}} = \mathtt{n_{fa}}$, respectively. In Table 2.1, the number of equations $\mathtt{n_{eq}}$, for the Monolithic and staggered approaches, is shown for 2D and 3D element types considering a mesh with a given N nodes. The size of each local problem is $(\mathtt{n_{sd}} + 1)^2$

Type	$\mathtt{n_{eq}}$	$\mathtt{n_{eq}^{RANS}}$	$\mathtt{n_{eq}^{SA}}$
TRI	$11N$	$8N$	$3N$
QUA	$7N$	$5N$	$2N$
TET	$45N$	$35N$	$10N$
HEX	$13N$	$10N$	$3N$
PRI	$22N$	$17N$	$5N$
PYR	$88/5N$	$68/5N$	$4N$

Table 2.1: Size of the global linear system of equations $\mathtt{n_{eq}}$ per element type in terms of number of nodes N. 2D and 3D elements. A comparison between the monolithic and staggered approaches is shown.

for the Monolithic case. The staggered case entails a local problem size of $\mathtt{n_{sd}}(\mathtt{n_{sd}}+1)$, and $\mathtt{n_{sd}}+1$ for RANS and SA respectively. The local problems can be trivially solved in parallel as there is no communication required between cells. Furthermore, the solution of a local problem does not actually require solving a linear system of equations of size $(\mathtt{n_{sd}}+1)^2$. After solving the global system (2.39), the increments of $\mathbf{L}_e$ and $\tilde{\mathbf{q}}_e$ are explicitly obtained from the increments of $\hat{\mathbf{u}}$ and $\hat{\boldsymbol{\nu}}$, using the first and third equations in (2.33). With the updated values of $\hat{\mathbf{u}}$, $\hat{\boldsymbol{\nu}}$, and $\tilde{\mathbf{q}}_e$, the last equation of (2.33) is used to obtain the increment of the turbulent variable $\tilde{\nu}_e$. Finally, the velocity can be updated using the second equation of (2.33). In summary, each local problem involves the solution of four independent equations, where three provide explicit expressions for the unknowns, whereas one requires the solution of a nonlinear problem with a scalar unknown. In this work, the solution of the linear system of equations of the global problem is performed using the parallel multifrontal sparse direct solver `MUMPS`, from Amestoy et al. (2001, 2019). The interface to the `MUMPS` solver is implemented through the `PETSc` library, described in Balay et al. (2023).

2.6.2 Number of non-zeros in the sparse global matrix

A priori, it is not feasible to compare the cost of the two approaches because it is not possible to know the number of iterations required by the staggered scheme to converge, but some comments can be made about the memory requirements of both strategies.

Assuming that the number of exterior edges/faces in the mesh is negligible when compared to the number of interior edges/faces, the number of non-zero entries in the global system of the monolithic approach is

$$\mathtt{n_{nz}} \simeq \sum_{e=1}^{\mathtt{n_{el}}} \mathtt{n}_{\mathtt{fa}}^{e} \left[\mathtt{n}_{\mathtt{fa}}^{e}(\mathtt{n}_{\mathtt{sd}}^{2}+1) + 2\mathtt{n_{sd}}(\mathtt{n}_{\mathtt{fa}}^{e}+1)\right] - \mathtt{n_{fa}}(\mathtt{n_{sd}}+1)^2. \tag{2.58}$$

In contrast, the number of non-zero entries of the global problems associated with the RANS

Type	$\mathtt{n_{nz}}$	$\mathtt{n_{nz}^{RANS}}$	$\mathtt{n_{nz}^{SA}}$	$(\mathtt{n_{nz}^{RANS}}+\mathtt{n_{nz}^{SA}})/\mathtt{n_{nz}}$
TRI	$159N$	$84N$	$15N$	0.62
QUA	$142N$	$72N$	$14N$	0.61
TET	$1240N$	$750N$	$70N$	0.66
HEX	$564N$	$333N$	$33N$	0.65
PRI	$780N$	$465N$	$45N$	0.65
PYR	$624N$	$372N$	$36N$	0.65

Table 2.2: Number of non-zeros $\mathtt{n_{nz}}$ in the global sparse matrix per element type in terms of number of nodes N. 2D and 3D elements. A comparison between the monolithic and staggered approaches is shown.

and SA equations in the staggered approach is

$$\mathtt{n_{nz}^{RANS}} \simeq \sum_{e=1}^{\mathtt{n_{el}}} \mathtt{n_{fa}^{\mathit{e}}}\mathtt{n_{sd}}(\mathtt{n_{fa}^{\mathit{e}}}\mathtt{n_{sd}}+2) - \mathtt{n_{fa}}\mathtt{n_{sd}^2}, \qquad \mathtt{n_{nz}^{SA}} \simeq \sum_{e=1}^{\mathtt{n_{el}}} (\mathtt{n_{fa}^{\mathit{e}}})^2 - \mathtt{n_{fa}}. \tag{2.59}$$

with $\mathtt{n_{fa}^{\mathit{e}}}$ being the number of faces in the element e and $\mathtt{n_{fa}}$ the total number of faces in the mesh. In Table 2.2, the number of non-zeros $\mathtt{n_{nz}}$ in the sparse global matrix, for the Monolithic and staggered approaches, is shown for 2D and 3D element types considering a mesh with a given N nodes. The results show that the staggered approach requires approximately 60% of the memory required by the monolithic approach to store the global matrices for 2D and 3D cases. We note that the solution of the laminar case, described in Section 2.5, has the same cost as the one to solve the RANS problem in a staggered approach.

Chapter 3

The hybrid pressure face-centred finite volume method for the incompressible Navier-Stokes equations

This chapter introduces the hybrid pressure face-centred finite volume (hybrid pressure FCFV)[1] method for solving the laminar incompressible flows. It starts in Section 3.1 with a brief presentation of the governing equations and assumptions on flow compressibility. In the following, Section 3.2, the hybrid pressure FCFV formulation is presented. In Section 3.3 we introduce the Voigt notation, used to impose the symmetry of the mixed variable defined in this formulation. In the following, the discrete form of the method is shown in Section 3.4. The solution strategy is delineated in Section 3.5. Finally, in Section 3.6 some computational aspects of the hybrid pressure FCFV implementation are discussed, and attention is given to the comparison of this formulation against the standard FCFV formulation presented in Chap. 2 for incompressible Navier-Stokes equations. The details on the numerical convective stabilisation and linearisation of the discrete hybrid pressure FCFV forms are given in the appendices A and C, respectively.

[1]The formulation presented in this chapter is part of the paper: M. Giacomini, D. Cortellessa, L. M. Vieira, R. Sevilla, and A. Huerta. **A Hybrid Pressure Formulation of the Face-Centred Finite Volume Method for Viscous Laminar Incompressible Flows**. Submitted to the International Journal for Numerical Methods in Engineering.

3.1 Governing equations

In compact form, the steady-state incompressible Navier-Stokes problem in the open bounded computational domain $\Omega \in \mathbb{R}^{\mathtt{n_{sd}}}$ with $\mathtt{n_{sd}}$ the number of spatial dimensions reads

$$\begin{cases} \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = \boldsymbol{s} & \text{in } \Omega, \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega, \end{cases} \tag{3.1}$$

where $\boldsymbol{u}$ is the velocity field, $\boldsymbol{s}$ the is the body force, and $\boldsymbol{\sigma}$ is the Cauchy stress tensor. To close this problem a constitutive relation is required, and typically for Newtonian fluids with Stoke's hypothesis, Stokes (1845), the constitutive equation reads:

$$\boldsymbol{\sigma} = -p\mathbf{I}_{\mathtt{n_{sd}}} + 2\nu\left(\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - \frac{1}{3}\left(\boldsymbol{\nabla}\cdot\boldsymbol{u}\right)\mathbf{I}_{\mathtt{n_{sd}}}\right), \tag{3.2}$$

where $\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u}$ is called the rate of deformation (or strain rate) tensor, i.e. $\boldsymbol{\nabla}^{\mathtt{s}} = \frac{1}{2}(\boldsymbol{\nabla} + \boldsymbol{\nabla}^{T})$, p is the kinematic pressure, and ν is the kinematic viscosity.

Remark 3.1 (Pointwise divergence-free condition). Eq. 3.2 is usually simplified by imposing that the velocity field is pointwise divergence-free. Thus, neglecting the last term of (3.2). Nonetheless, the formulation presented here enforces the incompressibility condition only in a weak sense, and consequently, the term proportional to $\boldsymbol{\nabla}\cdot\boldsymbol{u}$ is preserved in (3.2) and in the following derivation

Finally, after plugging the constitutive equation (3.2) into the momentum equation in (3.1) the explicit form of the non-dimensional incompressible Navier-Stokes problem reads

$$\begin{cases} \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot\left(\frac{2}{Re}\left(\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - \frac{1}{3}\boldsymbol{\nabla}\cdot\boldsymbol{u}\mathbf{I}_{\mathtt{n_{sd}}}\right) - p\mathbf{I}_{\mathtt{n_{sd}}}\right) = \boldsymbol{s} & \text{in } \Omega, \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega, \end{cases} \tag{3.3}$$

where Re denotes the Reynolds number. The strong form of the incompressible Navier-Stokes equations is closed with appropriate boundary conditions, written in the abstract form

$$\boldsymbol{B}(\boldsymbol{u}, p, \boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u}, \boldsymbol{\nabla}\cdot\boldsymbol{u}, \boldsymbol{g}) = \mathbf{0} \quad \text{on } \partial\Omega, \tag{3.4}$$

using the boundary operator $\boldsymbol{B}$. This work considers Dirichlet, Neumann, outflow, and symmetry boundary conditions. See Giacomini et al. (2020) for a discussion on the boundary conditions. The portions of the boundary $\partial\Omega$ where such conditions are imposed, denoted by Γ_D, Γ_N, Γ_O, and Γ_S, respectively, are disjoint by pairs and such that $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_O \cup \Gamma_S$.

The particular expression of the boundary operator $\boldsymbol{B}$ is

$$\boldsymbol{B} := \begin{cases} \boldsymbol{u} - \boldsymbol{u}_D & \text{on } \Gamma_D, \\ \left(\dfrac{2}{Re}\left(\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - \dfrac{1}{3}\boldsymbol{\nabla}\cdot\boldsymbol{u}\mathbf{I}_{\mathtt{n_{sd}}}\right) - \boldsymbol{u}\otimes\boldsymbol{u} - p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} - \boldsymbol{g} & \text{on } \Gamma_N, \\ \left(\dfrac{2}{Re}\left(\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - \dfrac{1}{3}\boldsymbol{\nabla}\cdot\boldsymbol{u}\mathbf{I}_{\mathtt{n_{sd}}}\right) - p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} & \text{on } \Gamma_O, \\ \left\{\begin{matrix} \boldsymbol{t}_k \cdot \left(\dfrac{2}{Re}\left(\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - \dfrac{1}{3}\boldsymbol{\nabla}\cdot\boldsymbol{u}\mathbf{I}_{\mathtt{n_{sd}}}\right) - p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} \\ \boldsymbol{u}\cdot\boldsymbol{n} \end{matrix}\right\} & \text{on } \Gamma_S, \end{cases} \tag{3.5}$$

where $\boldsymbol{n}$ and $\boldsymbol{t}_k$, for $k = 1, \ldots, \mathtt{n_{sd}} - 1$, are the outward unit normal and unit tangential vectors to the boundary, $\boldsymbol{u}_D$ is the imposed velocity on the Dirichlet boundary, and $\boldsymbol{g}$ is the imposed traction on the Neumann boundary representing a material surface.

In particular, if only Dirichlet boundary conditions are considered an additional constraint on the pressure needs to be imposed to avoid its indeterminacy. It is common to impose the mean pressure, namely

$$\int_\Omega p \, d\Omega = 0. \tag{3.6}$$

3.2 Hybrid pressure face-centred finite volume formulation

Consider a partition of the domain Ω in a set of non-overlapping cells $\{\Omega_e\}_{e=1,\ldots,\mathtt{n_{el}}}$, where $\mathtt{n_{el}}$ is the number of cells. The boundary of each cell, $\partial\Omega_e$, is formed by a set of faces $\{\Gamma_{e,j}\}_{j=1,\ldots,\mathtt{n}^e_{\mathtt{fa}}}$, where $\mathtt{n}^e_{\mathtt{fa}}$ is the number of faces of cell Ω_e. Finally the set of faces not on the boundary of the domain form the internal interface Γ, with $\Gamma := \left[\bigcup_{e=1}^{\mathtt{n_{el}}} \partial\Omega_e\right] \setminus \partial\Omega$.

3.2.1 Mixed formulation

In this broke domain, the Navier-Stokes problem reported in (3.3) is written as a system of first-order equations after introducing the mixed variables $\boldsymbol{L}$, namely

$$\begin{cases} \boldsymbol{L} + \left(\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - \frac{1}{3}(\boldsymbol{\nabla}\cdot\boldsymbol{u})\mathbf{I}_{\mathtt{n_{sd}}}\right) = \mathbf{0} & \text{in } \Omega_e \\ \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) + \boldsymbol{\nabla}\cdot\left(\dfrac{2}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right) = \boldsymbol{s} & \text{in } \Omega_e \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega_e \\ \boldsymbol{B}(\boldsymbol{u}, p, \boldsymbol{L}, \boldsymbol{g}) = \mathbf{0} & \text{on } \partial\Omega, \\ [\![\boldsymbol{u}\otimes\boldsymbol{n}]\!] = \mathbf{0} & \text{on } \Gamma, \\ \left[\!\!\left[(\boldsymbol{u}\otimes\boldsymbol{u})\boldsymbol{n} + \left(\dfrac{2}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n}\right]\!\!\right] = \mathbf{0} & \text{on } \Gamma, \end{cases} \tag{3.7}$$

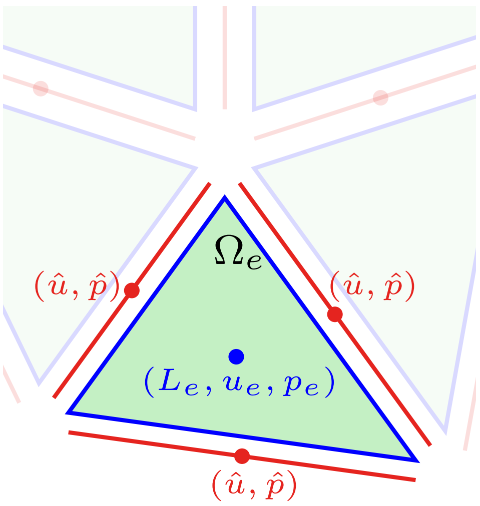


Figure 3.1: Detail of a triangular mesh highlighting a cell Ω_e where the velocity $\boldsymbol{u}_e$, velocity gradient $\boldsymbol{L}_e$, mean pressure p_e are defined and edges where the hybrid velocity $\hat{\boldsymbol{u}}$ and hybrid pressure variable $\hat{p}$ are defined.

The last two equations in (3.7), called transmission conditions, impose the continuity of the velocity and the normal fluxes across the interior faces. In here, the jump operator along a face in Γ, with neighboring cells Ω_l and Ω_r, is defined as

$$[\![\odot]\!] = \odot_l + \odot_r.$$

following the notation introduced in Montlaur et al. (2008).

3.2.2 Strong form of the local and global problems

As mentioned for the standard FCFV formulation presented in Chap. 2, the hybrid pressure FCFV formulation is also based on the works on FCFV of Sevilla et al. (2018, 2019) and Giacomini and Sevilla (2020); Vieira et al. (2020); Vila-Pérez et al. (2022). Similary, the mixed formulation in (3.7) is solved in two steps, as proposed in seminal works on HDG performed by Cockburn and Gopalakrishnan (2009); Nguyen et al. (2010); Cockburn et al. (2010); Nguyen et al. (2011). First, $\texttt{n}_{\texttt{el}}$ *local problems* are defined, one per cell, to express the primal variables, $\boldsymbol{u}$, and p, and the mixed variable, $\boldsymbol{L}$ as a function of new independent variables, $\hat{\boldsymbol{u}}$ and $\hat{p}$, defined on the boundary of the cell.

Figure 3.1 shows a sketch of a triangular mesh with the variables defined on each cell and on each edge. As will be shown in this section, the global system of equations to be solved only involves the variables on the edges and the cell variables can then be recovered cell-by-cell. Given the hybrid variables $\hat{\boldsymbol{u}} \in \Gamma \cup \partial\Omega \setminus \Gamma_D$ and $\hat{p} \in \Gamma \cup \partial\Omega$ representing respectively the trace of the velocity and the pressure a the cell faces, a set of $\texttt{n}_{\texttt{el}}$ local problems is introduced to determine element-by-element $(\boldsymbol{L}, \boldsymbol{u}, p)$ for all $\boldsymbol{x} \in \Omega_e \subset \Omega$, namely

$$\begin{cases} \boldsymbol{L} + \left(\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u} - \frac{1}{3}(\boldsymbol{\nabla}\cdot\boldsymbol{u})\mathbf{I}_{\mathtt{n_{sd}}}\right) = \boldsymbol{0} & \text{in } \Omega_e \\ \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) + \boldsymbol{\nabla}\cdot\left(\frac{2}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right) = \boldsymbol{s} & \text{in } \Omega_e \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega_e \\ \boldsymbol{u} = \boldsymbol{u}_D & \text{on } \partial\Omega_e \cap \Gamma_D, \\ \boldsymbol{u} = \hat{\boldsymbol{u}} & \text{on } \partial\Omega_e \setminus \Gamma_D, \end{cases} \tag{3.8}$$

Hence, the local problem provides $(\boldsymbol{L}, \boldsymbol{u}, p)$, in each cell, in terms of the global unknowns $\hat{\boldsymbol{u}}$ and $\hat{p}$. Of course, to have a unique solution both the hybrid velocity and hybrid pressure must be compatible and they are determined by imposing the remaining equations in (3.7). This is done in the second step, where the global problem accounts for the transmission conditions and the remaining boundary conditions, namely,

$$\begin{cases} \left[\!\left[(\boldsymbol{u}\otimes\boldsymbol{u})\boldsymbol{n} + \left(\frac{2}{Re}\boldsymbol{L} + p\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n}\right]\!\right] = \boldsymbol{0} & \text{on } \Gamma, \\ [\![\boldsymbol{u}\otimes\boldsymbol{n}]\!] = \boldsymbol{0} & \text{on } \Gamma, \\ \boldsymbol{B}(\boldsymbol{u}, p, \boldsymbol{L}, \boldsymbol{g}) = \boldsymbol{0} & \text{on } \partial\Omega\backslash\Gamma_D, \end{cases} \tag{3.9}$$

Moreover, note that the second equation of (3.9) can be rewritten in terms of the orthogonal system of vectors $\{\boldsymbol{n}, \boldsymbol{t}_1, \dots, \boldsymbol{t}_{\mathtt{n_{sd}}-1}\}$ representing the normal and tangential directions to the internal mesh edges ($\mathtt{n_{sd}} = 2$) or faces ($\mathtt{n_{sd}} = 3$). More specifically the inter-cell continuity of the velocity is equivalent to

$$\begin{cases} [\![\boldsymbol{u}\cdot\boldsymbol{n}]\!] = 0 & \text{on } \Gamma, \\ (\boldsymbol{u}_l - \boldsymbol{u}_r)\boldsymbol{t}_k = 0 & \text{on } \Gamma \quad \text{for } k = 1, \dots, \mathtt{n_{sd}}-1. \end{cases} \tag{3.10}$$

Given the uniqueness of $\hat{\boldsymbol{u}}$ on the faces of the mesh and the condition imposed on $\boldsymbol{u}$ at the cell boundaries in the local problem, the continuity of the tangential component of the velocity, in the second equation of (3.10) is automatically fulfilled. On the contrary, the continuity of the mass flux across Γ, represented by the first equation of (3.10), does not follow straightforwardly as in traditional formulations of hybrid methods, see Giacomini et al. (2020); Sevilla et al. (2018). Indeed, the relaxation of the compressibility constraint (see Remark 3.1) is responsible for the velocity not fulfilling strongly such equation and for the normal component of $\boldsymbol{u}$ to violate the condition prescribed at the cell boundaries in the local problem (3.8). Hence, the global problem of FCFV formulation with the hybrid pressure needs to explicitly enforce the continuity of the mass flux on Γ and the appropriate value of the normal component of the velocity on $\partial\Omega$, that is, Eq. 3.9 is to be complemented with

$$\begin{cases} \boldsymbol{u}\cdot\boldsymbol{n} = \boldsymbol{u}_D\cdot\boldsymbol{n} & \text{on } \partial\Omega\cap\Gamma_D. \\ \boldsymbol{u}\cdot\boldsymbol{n} = \hat{\boldsymbol{u}}\cdot\boldsymbol{n} & \text{on } \partial\Omega\setminus\Gamma_D. \\ [\![\boldsymbol{u}\cdot\boldsymbol{n}]\!] = 0 & \text{on } \Gamma, \end{cases} \tag{3.11}$$

3.2.3 Integral form of the local and global problems

The integral form of the local problems, given by equations (3.8), for each cell is

$$\begin{cases} -\displaystyle\int_{\Omega_e}\boldsymbol{L}\,d\Omega = \int_{\partial\Omega_e\cap\Gamma_D}\frac{1}{2}\left(\boldsymbol{u}_D\otimes\boldsymbol{n}+\boldsymbol{n}\otimes\boldsymbol{u}_D\right)\,d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D}\frac{1}{2}\left(\hat{\boldsymbol{u}}\otimes\boldsymbol{n}+\boldsymbol{n}\otimes\hat{\boldsymbol{u}}\right)\,d\Gamma \\ \qquad\qquad -\displaystyle\int_{\partial\Omega_e\cap\Gamma_D}\frac{1}{3}\boldsymbol{u}_D\cdot\boldsymbol{n}\mathbf{I}_{\mathtt{n_{sd}}}\,d\Gamma - \int_{\partial\Omega_e\setminus\Gamma_D}\frac{1}{3}\hat{\boldsymbol{u}}\cdot\boldsymbol{n}\mathbf{I}_{\mathtt{n_{sd}}}\,d\Gamma, \\ \displaystyle\int_{\partial\Omega_e}\left(\widehat{\boldsymbol{u}\otimes\boldsymbol{u}+p\mathbf{I}_{\mathtt{n_{sd}}}}\right)\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega_e}\left(\widehat{\frac{2}{Re}\boldsymbol{L}} - \frac{2}{Re}\boldsymbol{L}\right)\boldsymbol{n}\,d\Gamma = \int_{\Omega_e}\boldsymbol{s}\,d\Omega, \\ \displaystyle\int_{\partial\Omega_e}\widehat{\boldsymbol{u}\cdot\boldsymbol{n}}\,d\Gamma = 0 \end{cases} \tag{3.12}$$

Note that the variational form of the momentum equation above is obtained after integrating twice by parts of the term containing the mixed variable, whereas the others (i.e. pressure and convective) terms are integrated by parts only once. This, as noted in Sevilla and Huerta (2016); Giacomini et al. (2020), preserves the symmetry of the local problem for the diffusive operator and, consequently, for the Stokes problem.

The integral form of the global problem is given by the continuity of the momentum fluxes, given by equation (3.9), and by the inter-cell continuity of the mass flux and the normal component of the velocity on the external boundary, see Eq. 3.11,

$$\begin{cases} \displaystyle\sum_{e=1}^{\mathtt{n_{el}}}\left\{\int_{\partial\Omega_e\setminus\partial\Omega}\widehat{(\boldsymbol{u}\otimes\boldsymbol{u}+p\mathbf{I}_{\mathtt{n_{sd}}})}\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega_e\setminus\partial\Omega}\widehat{\frac{2}{Re}\boldsymbol{L}}\,\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega_e\cap\partial\Omega\setminus\Gamma_D}\boldsymbol{B}(\boldsymbol{u},p,\boldsymbol{L})d\Gamma\right\} = \boldsymbol{0}, \\ \displaystyle\int_{\partial\Omega_e\setminus\partial\Omega}\widehat{\boldsymbol{u}\cdot\boldsymbol{n}}\,d\Gamma + \int_{\partial\Omega_e\cap\Gamma_D}\widehat{\boldsymbol{u}\cdot\boldsymbol{n}} - \boldsymbol{u}_D\cdot\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega_e\cap\partial\Omega\setminus\Gamma_D}\widehat{\boldsymbol{u}\cdot\boldsymbol{n}} - \hat{\boldsymbol{u}}\cdot\boldsymbol{n}\,d\Gamma = 0. \end{cases} \tag{3.13}$$

The numerical trace of mass flux is defined as follows

$$\widehat{(\boldsymbol{u}\otimes\boldsymbol{u}+p\mathbf{I}_{\mathtt{n_{sd}}})}\boldsymbol{n} := \begin{cases} \left(\boldsymbol{u}_D\otimes\boldsymbol{u}_D+\hat{p}\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} + \boldsymbol{\tau}^a(\boldsymbol{u}-\boldsymbol{u}_D) & \text{on } \partial\Omega_e\cap\Gamma_D, \\ \left(\hat{\boldsymbol{u}}\otimes\hat{\boldsymbol{u}}+\hat{p}\mathbf{I}_{\mathtt{n_{sd}}}\right)\boldsymbol{n} + \boldsymbol{\tau}^a(\boldsymbol{u}-\hat{\boldsymbol{u}}) & \text{elsewhere,} \end{cases} \tag{3.14a}$$

and for the diffusive flux

$$\widehat{\frac{2}{Re}\boldsymbol{L}\,\boldsymbol{n}} := \begin{cases} \dfrac{2}{Re}\boldsymbol{L}\,\boldsymbol{n}+\boldsymbol{\tau}^d(\boldsymbol{u}-\boldsymbol{u}_D) & \text{on } \partial\Omega_e\cap\Gamma_D, \\ \dfrac{2}{Re}\boldsymbol{L}\,\boldsymbol{n}+\boldsymbol{\tau}^d(\boldsymbol{u}-\hat{\boldsymbol{u}}) & \text{elsewhere,} \end{cases} \tag{3.14b}$$

Note that since the compressibility equation is only enforced in a weak sense in the present formulation, the numerical trace of the mass flux needs to be introduced. Thus, the flux of the normal component of the velocity across the faces or numerical trace of the mass flux reads

$$\widehat{\boldsymbol{u}\cdot\boldsymbol{n}} := \begin{cases} \boldsymbol{u}_D\cdot\boldsymbol{n} + \tau^p(p-\hat{p}) & \text{on } \partial\Omega_e\cap\Gamma_D, \\ \hat{\boldsymbol{u}}\cdot\boldsymbol{n} + \tau^p(p-\hat{p}) & \text{elsewhere,} \end{cases} \tag{3.14c}$$

An appropriate stabilisation, given by $\boldsymbol{\tau}^a$, and $\boldsymbol{\tau}^d$, is required to ensure that the numerical scheme is well-posed. The diffusive stabilisation is taken as

$$\boldsymbol{\tau}^d := \frac{\beta}{Re}\mathbf{I}_{\mathtt{n_{sd}}} \tag{3.15}$$

where β is a numerical parameter, following Sevilla et al. (2018).

Concerning the convective stabilization, new definitions proposed in this work and already mentioned in Chap. 2 are also employed here. The derivation of three different convective stabilisations, based on the Lax-Friedrichs (LF), Roe, and Harten-Lax-van Leer (HLL) Riemann solvers, is detailed in Appendix A. Here we introduce the scalar stabilization parameter τ^p related to numerical mass flux and defined to control the compressibility effect due to the relaxation of the continuity equation.

After introducing the expression of the numerical fluxes given by equation (3.14) into the local problem (3.12), the integral form of the local problem reads

$$\begin{cases} -\displaystyle\int_{\Omega_e}\boldsymbol{L}\,d\Omega = \int_{\partial\Omega_e\cap\Gamma_D}\frac{1}{2}(\boldsymbol{u}_D\otimes\boldsymbol{n}+\boldsymbol{n}\otimes\boldsymbol{u}_D)\,d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D}\frac{1}{2}(\hat{\boldsymbol{u}}\otimes\boldsymbol{n}+\boldsymbol{n}\otimes\hat{\boldsymbol{u}})\,d\Gamma \\ \qquad\qquad -\displaystyle\int_{\partial\Omega_e\cap\Gamma_D}\frac{1}{3}(\boldsymbol{u}_D\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}}\,d\Gamma - \int_{\partial\Omega_e\setminus\Gamma_D}\frac{1}{3}(\hat{\boldsymbol{u}}\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}}\,d\Gamma, \\ \displaystyle\int_{\partial\Omega_e}\boldsymbol{\tau}\boldsymbol{u}\,d\Gamma = \int_{\Omega_e}\boldsymbol{s}\,d\Omega + \int_{\partial\Omega_e\cap\Gamma_D}(\boldsymbol{\tau}-(\boldsymbol{u}_D\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}})\boldsymbol{u}_D\,d\Gamma \\ \qquad\qquad +\displaystyle\int_{\partial\Omega_e\setminus\Gamma_D}(\boldsymbol{\tau}-(\hat{\boldsymbol{u}}\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}})\hat{\boldsymbol{u}}\,d\Gamma - \int_{\partial\Omega_e}\hat{p}\,\boldsymbol{n}\,d\Gamma, \\ \displaystyle\int_{\partial\Omega_e}\tau^p p\,d\Gamma = -\int_{\partial\Omega_e\cap\Gamma_D}\boldsymbol{u}_D\cdot\boldsymbol{n}\,d\Gamma - \int_{\partial\Omega_e\setminus\Gamma_D}\hat{\boldsymbol{u}}\cdot\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega_e}\tau^p\hat{p}\,d\Gamma. \end{cases} \tag{3.16}$$

where $\boldsymbol{\tau} = \boldsymbol{\tau}^a + \boldsymbol{\tau}^d$.

Similarly, by introducing the numerical fluxes into the integral form of equation (3.13), the global problem, of determining the hybrid velocity and pressure, is obtained

$$\begin{cases} \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Bigg\{ \int_{\partial\Omega_e \setminus \partial\Omega} \frac{2}{Re} \boldsymbol{L}\boldsymbol{n}\, d\Gamma + \int_{\partial\Omega_e \setminus \partial\Omega} \boldsymbol{\tau}\boldsymbol{u}\, d\Gamma - \int_{\partial\Omega_e \setminus \partial\Omega} \boldsymbol{\tau}\hat{\boldsymbol{u}}\, d\Gamma \\ \displaystyle\qquad + \int_{\partial\Omega_e \cap \partial\Omega \setminus \Gamma_D} \hat{\boldsymbol{B}}(\boldsymbol{u}, \boldsymbol{L}, \hat{\boldsymbol{u}}, \hat{p}, \boldsymbol{\tau}, \boldsymbol{g}) d\Gamma \Bigg\} = \boldsymbol{0}, \\ \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Bigg\{ \int_{\partial\Omega_e} \tau^p (p - \hat{p}) d\Gamma \Bigg\} = 0. \end{cases} \tag{3.17}$$

It is worth noting that the first term of the convective flux in equation (3.14a) does not appear on internal faces within the global problem (3.17) due to the unique definition of the hybrid variables $\hat{\boldsymbol{u}}$ and $\hat{p}$ on internal faces, nonetheless, the contribution of those terms is present on the boundary operator $\hat{\boldsymbol{B}}$, defined in Eq. (3.18) for each boundary type. The hybrid pressure FCFV version of the boundary operator appearing in the global problem is given by

$$\hat{\boldsymbol{B}} := \begin{cases} \left(\dfrac{2}{Re} \boldsymbol{L} + \hat{\boldsymbol{u}} \otimes \hat{\boldsymbol{u}} + \hat{p} \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau}(\boldsymbol{u} - \hat{\boldsymbol{u}}) + \boldsymbol{g} & \text{on } \Gamma_N, \\ \left(\dfrac{2}{Re} \boldsymbol{L} + \hat{p} \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau}^d (\boldsymbol{u} - \hat{\boldsymbol{u}}) & \text{on } \Gamma_O, \\ \begin{cases} \boldsymbol{t}_k \cdot \left[\left(\dfrac{2}{Re} \boldsymbol{L} + \hat{p} \mathbf{I}_{\mathtt{n_{sd}}} \right) \boldsymbol{n} + \boldsymbol{\tau}^d (\boldsymbol{u} - \hat{\boldsymbol{u}}) \right] \\ \hat{\boldsymbol{u}} \cdot \boldsymbol{n} \end{cases} & \text{on } \Gamma_S, \end{cases} \tag{3.18}$$

As the Dirichlet boundary conditions are accounted for in the local problems, the FCFV boundary operators are only described for Neumann, outflow, and symmetry boundaries.

Remark 3.2 (Cell mass flux). The last equation of (3.16) integrates the numerical trace of the cell mass flux. In the limit of incompressibility, when $p \to \hat{p}$, it reduces to

$$J_e := \int_{\partial\Omega_e \cap \Gamma_D} \boldsymbol{u}_D \cdot \boldsymbol{n}\, d\Gamma + \int_{\partial\Omega_e \setminus \Gamma_D} \hat{\boldsymbol{u}} \cdot \boldsymbol{n}\, d\Gamma \tag{3.19}$$

This equation corresponds to the compatibility condition, or divergence-free condition, and is not exactly fulfilled in the hybrid pressure formulation, in contrast to the standard FCFV.

3.3 Voigt notation for the symmetric mixed variable

Before diving into the discrete forms of the local and global problems, we first introduce the Voigt notation, which is used here to describe the symmetric strain rate tensor $\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u}$ and consequently the mixed variable $\boldsymbol{L}$. The Voigt notation of the mixed variable strongly enforces the tensor symmetry, as explored in Giacomini et al. (2018), allowing optimal convergence and superconvergence properties even for low-order approximations. We evoke from Giacomini et al. (2018, 2020) the Voigt notation, where the symmetric $\boldsymbol{L}$ tensor of size $\mathtt{n_{sd}} \times \mathtt{n_{sd}}$ has its diagonal and off-diagonal components stored in a vector of length $\mathtt{m_{sd}} = \mathtt{n_{sd}}(\mathtt{n_{sd}}+1)/2$ (i.e., three components in 2D and six components in 3D). The strain-rate tensor $\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u}$ is written as follows, with $\mathbf{e}_V \in \mathbb{R}^{\mathtt{m_{sd}}}$.

$$\mathbf{e}_V := \begin{cases} \begin{bmatrix} e_{11} & e_{22} & e_{12} \end{bmatrix}^T & \text{in 2D} \\ \begin{bmatrix} e_{11} & e_{22} & e_{33} & e_{12} & e_{13} & e_{23} \end{bmatrix}^T & \text{in 3D} \end{cases} \tag{3.20}$$

with the definition above, the strain rate tensor is written in Voigt notation as $\mathbf{e}_V = \boldsymbol{\nabla}_{\mathtt{s}}\boldsymbol{u}$, with the $\mathtt{m_{sd}} \times \mathtt{n_{sd}}$ matrix introduced

$$\boldsymbol{\nabla}_{\mathtt{s}} := \begin{cases} \begin{bmatrix} \partial/\partial x_1 & 0 & \partial/\partial x_1 \\ 0 & \partial/\partial x_2 & \partial/\partial x_1 \end{bmatrix}^T & \text{in 2D} \\ \begin{bmatrix} \partial/\partial x_1 & 0 & 0 & \partial/\partial x_2 & \partial/\partial x_3 & 0 \\ 0 & \partial/\partial x_2 & 0 & \partial/\partial x_1 & 0 & \partial/\partial x_3 \\ 0 & 0 & \partial/\partial x_3 & 0 & \partial/\partial x_1 & \partial/\partial x_2 \end{bmatrix}^T & \text{in 3D} \end{cases} \tag{3.21}$$

From the above definition, the strain tensor $\boldsymbol{\nabla}^{\mathtt{s}}\boldsymbol{u}$ components can be retrieved from the Voigt notation counterpart by multiplying the off-diagonal terms by a factor of $1/2$.

The mixed variable $\boldsymbol{L}$ is expressed in Voigt notation with the help of the following auxiliary operators,

$$\mathbf{N} := \begin{cases} \begin{bmatrix} n_1 & 0 & n_2 \\ 0 & n_2 & n_1 \end{bmatrix}^T & \text{in 2D} \\ \begin{bmatrix} n_1 & 0 & 0 & n_2 & n_3 & 0 \\ 0 & n_2 & 0 & n_1 & 0 & n_3 \\ 0 & 0 & n_3 & 0 & n_1 & n_2 \end{bmatrix}^T & \text{in 3D} \end{cases} \tag{3.22}$$

$$\mathbf{D} := \begin{cases} \dfrac{1}{Re}\begin{bmatrix} 2\mathbf{I}_{\mathtt{n_{sd}}} & \mathbf{0}_{\mathtt{n_{sd}}\times 1} \\ \mathbf{0}^T_{\mathtt{n_{sd}}\times 1} & 1 \end{bmatrix} & \text{in 2D} \\ \dfrac{1}{Re}\begin{bmatrix} 2\mathbf{I}_{\mathtt{n_{sd}}} & \mathbf{0}_{\mathtt{n_{sd}}} \\ \mathbf{0}_{\mathtt{n_{sd}}} & 1 \end{bmatrix} & \text{in 3D} \end{cases} \qquad \mathbf{E} := \begin{cases} \begin{bmatrix} 1, & 1, & 0 \end{bmatrix}^T & \text{in 2D} \\ \begin{bmatrix} 1, & 1, & 1 & 0, & 0, & 0 \end{bmatrix}^T & \text{in 3D} \end{cases} \tag{3.23}$$

where $\mathbf{N} \in \mathbb{R}^{\mathtt{m_{sd}}\times\mathtt{n_{sd}}}$, and $\mathbf{D} \in \mathbb{R}^{\mathtt{m_{sd}}\times\mathtt{m_{sd}}}$ are auxiliary operators storing the face normal vector components and the constitutive parameters, respectively. The vector $\mathbf{E} \in \mathbb{R}^{\mathtt{msd}}$ allows writing the discrete divergence of $\boldsymbol{u}$ in Voigt notation.

3.3.1 Strong forms with mixed variable in Voigt notation

The strong form of the local problem in (3.8) rewritten with the mixed variable $\boldsymbol{L}$ in Voigt notation reads

$$\begin{cases} \boldsymbol{L} + \Big(\boldsymbol{\nabla}_{\mathtt{s}}\boldsymbol{u} - \frac{1}{3}(\mathbf{E}^T\boldsymbol{\nabla}_{\mathtt{s}}\boldsymbol{u})\Big) = \mathbf{0} & \text{in } \Omega_e \\ \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) + \boldsymbol{\nabla}_{\mathtt{s}}{}^T\mathbf{D}\boldsymbol{L} + \boldsymbol{\nabla}\cdot(p\mathbf{I}_{\mathtt{n_{sd}}}) = \boldsymbol{s} & \text{in } \Omega_e \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega_e \\ \boldsymbol{u} = \boldsymbol{u}_D & \text{on } \partial\Omega_e \cap \Gamma_D, \\ \boldsymbol{u} = \hat{\boldsymbol{u}} & \text{on } \partial\Omega_e \setminus \Gamma_D, \end{cases} \tag{3.24}$$

The strong form of the global problem in (3.9) rewritten with the mixed variable $\boldsymbol{L}$ in Voigt notation reads

$$\begin{cases} \Big[\!\!\Big[(\boldsymbol{u}\otimes\boldsymbol{u})\boldsymbol{n} + \mathbf{N}^T\mathbf{D}\boldsymbol{L} + p\,\boldsymbol{n}\Big]\!\!\Big] = \mathbf{0} & \text{on } \Gamma, \\ [\![\boldsymbol{u}\otimes\boldsymbol{n}]\!] = \mathbf{0} & \text{on } \Gamma, \\ \boldsymbol{B}(\boldsymbol{u}, p, \boldsymbol{L}, \boldsymbol{g}) = \mathbf{0} & \text{on } \partial\Omega\setminus\Gamma_D, \end{cases} \tag{3.25}$$

3.3.2 Integral forms with mixed variable in Voigt notation

The integral for of the local problem in (3.16) rewritten with the mixed variable $\boldsymbol{L}$ in Voigt notation reads

$$\begin{cases} -\displaystyle\int_{\Omega_e}\boldsymbol{L}\,d\Omega = \int_{\partial\Omega_e\cap\Gamma_D}\mathbf{N}\boldsymbol{u}_D\,d\Gamma + \int_{\partial\Omega_e\setminus\Gamma_D}\mathbf{N}\hat{\boldsymbol{u}}\,d\Gamma \\ \qquad\qquad\qquad - \displaystyle\int_{\partial\Omega_e\cap\Gamma_D}\mathbf{E}\frac{\mathbf{E}^T}{3}\boldsymbol{u}_D\,d\Gamma - \int_{\partial\Omega_e\setminus\Gamma_D}\mathbf{E}\frac{\mathbf{E}^T}{3}\hat{\boldsymbol{u}}\,d\Gamma, \\ \displaystyle\int_{\partial\Omega_e}\boldsymbol{\tau}\boldsymbol{u}\,d\Gamma = \int_{\Omega_e}\boldsymbol{s}\,d\Omega + \int_{\partial\Omega_e\cap\Gamma_D}(\boldsymbol{\tau} - (\boldsymbol{u}_D\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}})\boldsymbol{u}_D\,d\Gamma \\ \qquad\qquad + \displaystyle\int_{\partial\Omega_e\setminus\Gamma_D}(\boldsymbol{\tau} - (\hat{\boldsymbol{u}}\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}})\hat{\boldsymbol{u}}\,d\Gamma - \int_{\partial\Omega_e}\hat{p}\,\boldsymbol{n}\,d\Gamma, \\ \displaystyle\int_{\partial\Omega_e}\tau^p p\,d\Gamma = -\int_{\partial\Omega_e\cap\Gamma_D}\boldsymbol{u}_D\cdot\boldsymbol{n}\,d\Gamma - \int_{\partial\Omega_e\setminus\Gamma_D}\hat{\boldsymbol{u}}\cdot\boldsymbol{n}\,d\Gamma + \int_{\partial\Omega_e}\tau^p\hat{p}\,d\Gamma. \end{cases} \tag{3.26}$$

where $\boldsymbol{\tau} = \boldsymbol{\tau}^a + \boldsymbol{\tau}^d$.

The integral for of the global problem in (3.17) rewritten with the mixed variable $\boldsymbol{L}$ in

Voigt notation reads

$$\begin{cases} \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Bigg\{ \int_{\partial\Omega_e\setminus\partial\Omega} \mathbf{N}^T\mathbf{D}\boldsymbol{L}\, d\Gamma + \int_{\partial\Omega_e\setminus\partial\Omega} \boldsymbol{\tau}\boldsymbol{u}\, d\Gamma - \int_{\partial\Omega_e\setminus\partial\Omega} \boldsymbol{\tau}\hat{\boldsymbol{u}}\, d\Gamma \\ \qquad\qquad + \displaystyle\int_{\partial\Omega_e\cap\partial\Omega\setminus\Gamma_D} \hat{\boldsymbol{B}}(\boldsymbol{u},\boldsymbol{L},\hat{\boldsymbol{u}},\hat{p},\boldsymbol{\tau},\boldsymbol{g}) d\Gamma \Bigg\} = \mathbf{0}, \\ \displaystyle\sum_{e=1}^{\mathtt{n_{el}}} \Bigg\{ \int_{\partial\Omega_e} \tau^p (p-\hat{p}) d\Gamma \Bigg\} = 0. \end{cases} \tag{3.27}$$

Similarly, the hybrid pressure FCFV version of the boundary operator of (3.18) will read as

$$\hat{\boldsymbol{B}} := \begin{cases} \mathbf{N}^T\mathbf{D}\boldsymbol{L} + (\hat{\boldsymbol{u}}\otimes\hat{\boldsymbol{u}} + \hat{p}\mathbf{I}_{\mathtt{n_{sd}}})\,\boldsymbol{n} + \boldsymbol{\tau}(\boldsymbol{u}-\hat{\boldsymbol{u}}) + \boldsymbol{g} & \text{on } \Gamma_N, \\ \mathbf{N}^T\mathbf{D}\boldsymbol{L} + \hat{p}\boldsymbol{n} + \boldsymbol{\tau}^d(\boldsymbol{u}-\hat{\boldsymbol{u}}) & \text{on } \Gamma_O, \\ \begin{cases} \boldsymbol{t}_k \cdot \Big[\mathbf{N}^T\mathbf{D}\boldsymbol{L} + \hat{p}\boldsymbol{n} + \boldsymbol{\tau}^d(\boldsymbol{u}-\hat{\boldsymbol{u}})\Big] \\ \hat{\boldsymbol{u}}\cdot\boldsymbol{n} \end{cases} & \text{on } \Gamma_S, \end{cases} \tag{3.28}$$

3.4 Hybrid pressure face-centred finite volume discretisation

To simplify the presentation, the set of indices for the faces of a cell is denoted by $\mathcal{A}_e := \{1,\dots,\mathtt{n}_{\mathtt{fa}}^e\}$. In addition, the set of indices for the faces of a cell on the Dirichlet boundary, interior to the domain and on the boundary of the domain are denoted by $\mathcal{D}_e := \{j \in \mathcal{A}_e \mid \Gamma_{e,j} \cap \Gamma_D \neq \emptyset\}$, $\mathcal{I}_e := \{j \in \mathcal{A}_e \mid \Gamma_{e,j} \cap \partial\Omega = \emptyset\}$ and $\mathcal{E}_e := \{j \in \mathcal{A}_e \mid \Gamma_{e,j} \cap \partial\Omega \neq \emptyset\}$, respectively. Given the imposition of Dirichlet boundary conditions in the local problems, it is also convenient to denote by $\mathcal{B}_e := \mathcal{A}_e \setminus \mathcal{D}_e$ and $\mathcal{C}_e := \mathcal{E}_e \setminus \mathcal{D}_e$ the set of indices for the faces of a cell not on a Dirichlet boundary and for the external faces not on a Dirichlet boundary. Finally, it is useful to introduce the indicator function of a set $\mathcal{S}$, i.e.

$$\chi_{\mathcal{S}}(i) = \begin{cases} 1 & \text{if } \ i \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases} \tag{3.29}$$

3.4.1 Spatial discretisation

The hybrid pressure FCFV method introduces a constant approximation of the primal, $\boldsymbol{u}$, and p, and mixed, $\boldsymbol{L}$ variables in each cell as well as a constant approximation of the hybrid variables, $\hat{\boldsymbol{u}}$ and $\hat{p}$ on each cell face. The value of the primal and mixed variables in each cell is denoted by $\mathbf{u}_e$, p_e, and, $\mathbf{L}_e$, whereas the value of the hybrid variables on the j-th face of a cell is denoted by $\hat{\mathbf{u}}_j$ and $\hat{\mathrm{p}}_j$.

With the introduced notation, we follow with the discretization of the local and global problems. The discretization of the local problems given by Equation (3.26) reads

$$\begin{cases} -|\Omega_e|\mathbf{L}_e = \displaystyle\sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|(\mathbf{N}_j-\mathbf{E}\frac{\mathbf{E}^T}{3}\mathbf{N}_j)\boldsymbol{u}_{D,j}+\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|(\mathbf{N}_j-\mathbf{E}\frac{\mathbf{E}^T}{3}\mathbf{N}_j)\hat{\mathbf{u}}_j, \\ \displaystyle\sum_{j\in\mathcal{A}_e}|\Gamma_{e,j}|\boldsymbol{\tau}_j\mathbf{u}_e = |\Omega_e|\mathbf{s}_e+\sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j-(\boldsymbol{u}_{D,j}\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{u}_{D,j} \\ \qquad\qquad\qquad\displaystyle+\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\Big(\boldsymbol{\tau}_j-(\hat{\mathbf{u}}_j\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\hat{\mathbf{u}}_j-\sum_{j\in\mathcal{A}_e}|\Gamma_{e,j}|\boldsymbol{n}_j\hat{\mathrm{p}}_j, \\ \displaystyle\sum_{j\in\mathcal{A}_e}|\Gamma_{e,j}|\tau_j^p\mathrm{p}_e = -\sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\boldsymbol{u}_{D,j}\cdot\boldsymbol{n}_j-\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\hat{\mathbf{u}}_j\cdot\boldsymbol{n}_j+\sum_{j\in\mathcal{A}_e}|\Gamma_{e,j}|\tau_j^p\hat{\mathrm{p}}_j, \end{cases} \tag{3.30}$$

where $|\Omega_e|$ is the area/volume of the cell Ω_e and $|\Gamma_{e,j}|$ is the length/area of the edge/face $\Gamma_{e,j}$. Similarly, the discrete form of the global problem of (3.27) is

$$\begin{cases} \displaystyle\sum_{e=1}^{\mathtt{n_{el}}}|\Gamma_{e,i}|\Big\{\Big(\mathbf{N}_i^T\mathbf{D}\mathbf{L}_e+\boldsymbol{\tau}_i\mathbf{u}_e-\boldsymbol{\tau}_i\hat{\mathbf{u}}_i\Big)\chi_{\mathcal{I}_e}(i) \\ \qquad\qquad +\hat{\mathbf{B}}_i(\mathbf{u}_e,\mathbf{L}_e,\hat{\mathbf{u}}_i,\hat{\mathrm{p}}_i,\boldsymbol{\tau}_i,\mathbf{g}_i)\chi_{\mathcal{C}_e}(i)\Big\} = \mathbf{0} \quad \text{for } i\in\mathcal{B}_e, \\ \displaystyle\sum_{e=1}^{\mathtt{n_{el}}}|\Gamma_{e,i}|\Big\{\tau_i^p(\mathrm{p}_e-\hat{\mathrm{p}}_i)\Big\} = 0 \quad \text{for } i\in\mathcal{A}_e \end{cases} \tag{3.31}$$

where the FCFV discrete form of the boundary operators is given by

$$\hat{\mathbf{B}}_i := \begin{cases} \Big(\mathbf{N}_i^T\mathbf{D}\mathbf{L}_e+\hat{\mathbf{u}}_i\otimes\hat{\mathbf{u}}_i+\hat{\mathrm{p}}_i\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{n}_i+\boldsymbol{\tau}_i(\mathbf{u}_e-\hat{\mathbf{u}}_i)+\mathbf{g}_i & \text{on } \Gamma_N, \\ \Big(\mathbf{N}_i^T\mathbf{D}\mathbf{L}_e+\hat{\mathrm{p}}_i\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{n}_i+\boldsymbol{\tau}_i^d(\mathbf{u}_e-\hat{\mathbf{u}}_i) & \text{on } \Gamma_O, \\ \left\{\begin{matrix} \mathbf{t}_{k,i}\cdot\Big[\Big(\mathbf{N}_i^T\mathbf{D}\mathbf{L}_e+\hat{\mathrm{p}}_i\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{n}_i+\boldsymbol{\tau}_i^d(\mathbf{u}_e-\hat{\mathbf{u}}_i)\Big] \\ \hat{\mathbf{u}}_i\cdot\boldsymbol{n}_i \end{matrix}\right\} & \text{on } \Gamma_S, \end{cases} \tag{3.32}$$

Remark 3.3 (Discrete cell mass flux). The discrete form of the element mass flux of (3.19) reads

$$\mathrm{J}_e := \sum_{j\in\mathcal{D}_e}|\Gamma_{e,j}|\boldsymbol{u}_{D,j}\cdot\boldsymbol{n}_j+\sum_{j\in\mathcal{B}_e}|\Gamma_{e,j}|\hat{\mathbf{u}}_j\cdot\boldsymbol{n}_j \tag{3.33}$$

3.5 Hybrid pressure face-centred finite volume solution

The hybrid pressure FCFV solver consists of a non-linear problem in terms of the global variables increment only, namely $\Delta\hat{\boldsymbol{u}}$, and $\Delta\hat{p}$. This is possible by noticing that the discrete form of the local problem in (3.30) is linear in terms of the local variables $\boldsymbol{L}$, $\boldsymbol{u}$, and p.

Recalling the discrete form of the global problem in 3.31, where the residuals $\mathbf{R}_{\hat{u}}$,and $\mathbf{R}_{\hat{p}}$ can be obtained by assembling the contributions of

$$\begin{cases} \mathbf{R}_{e,i,\hat{u}} := |\Gamma_{e,i}| \Big\{ \Big(\mathbf{N}_i^T \mathbf{D} \mathbf{L}_e + \boldsymbol{\tau}_i \mathbf{u}_e - \boldsymbol{\tau}_i \hat{\mathbf{u}}_i \Big) \chi_{\mathcal{I}_e}(i) \\ \qquad\qquad + \hat{\mathbf{B}}_i(\mathbf{u}_e, \mathbf{L}_e, \mathrm{p}_e, \hat{\mathbf{u}}_i, \hat{\mathrm{p}}_i, \boldsymbol{\tau}_i, \mathbf{g}_i) \chi_{\mathcal{C}_e}(i) \Big\} & \text{for } i \in \mathcal{B}_e, \\ \mathrm{R}_{e,i,\hat{p}} := |\Gamma_{e,i}| \Big\{ \tau_i^p (\mathrm{p}_e - \hat{\mathrm{p}}_i) \Big\}, & \text{for } i \in \mathcal{A}_e, \end{cases} \tag{3.34}$$

The elimination of the local variables from the residuals expressions of (3.34) can be done by substituting the explicit expression of $\boldsymbol{L}$, $\boldsymbol{u}$, and p, from (3.30). This leads to

$$\begin{cases} \mathbf{R}_{e,i,\hat{u}} := |\Gamma_{e,i}| \Big\{ \Big(-|\Omega_e|^{-1} \sum_{j \in \mathcal{D}_e} |\Gamma_{e,j}| \mathbf{N}_i^T \mathbf{D} (\mathbf{N}_j - \mathbf{E} \frac{\mathbf{E}^T}{3} \mathbf{N}_j) \boldsymbol{u}_{D,j} \\ \qquad - |\Omega_e|^{-1} \sum_{j \in \mathcal{B}_e} |\Gamma_{e,j}| \mathbf{N}_i^T \mathbf{D} (\mathbf{N}_j - \mathbf{E} \frac{\mathbf{E}^T}{3} \mathbf{N}_j) \hat{\mathbf{u}}_j \\ \qquad + \sum_{j \in \mathcal{D}_e} |\Gamma_{e,j}| \boldsymbol{\tau}_i \boldsymbol{\alpha}_e^{-1} \Big(\boldsymbol{\tau}_j - (\boldsymbol{u}_{D,j} \cdot \boldsymbol{n}_j) \mathbf{I}_{\mathtt{n_{sd}}} \Big) \boldsymbol{u}_{D,j} \\ \qquad + \sum_{j \in \mathcal{B}_e} |\Gamma_{e,j}| \boldsymbol{\tau}_i \boldsymbol{\alpha}_e^{-1} \Big(\boldsymbol{\tau}_j - (\hat{\mathbf{u}}_j \cdot \boldsymbol{n}_j) \mathbf{I}_{\mathtt{n_{sd}}} \Big) \hat{\mathbf{u}}_j - \sum_{j \in \mathcal{A}_e} |\Gamma_{e,j}| \boldsymbol{\tau}_i \boldsymbol{\alpha}_e^{-1} \boldsymbol{n}_j \hat{\mathrm{p}}_j \\ \qquad + \boldsymbol{\tau}_i \boldsymbol{\alpha}_e^{-1} |\Omega_e| \mathbf{s}_e - \boldsymbol{\tau}_i \hat{\mathbf{u}}_i \Big) \chi_{\mathcal{I}_e}(i) + \hat{\mathbf{B}}_i(\hat{\mathbf{u}}_i, \hat{\mathrm{p}}_i, \boldsymbol{\tau}_i, \boldsymbol{\tau}_j, \mathbf{g}_i) \chi_{\mathcal{C}_e}(i) \Big\} & \text{for } i \in \mathcal{B}_e, \\ \mathrm{R}_{e,i,\hat{p}} := |\Gamma_{e,i}| \Big\{ -\tau_i^p \gamma_e^{-1} \sum_{j \in \mathcal{D}_e} |\Gamma_{e,j}| \boldsymbol{u}_{D,j} \cdot \boldsymbol{n}_j - \tau_i^p \gamma_e^{-1} \sum_{j \in \mathcal{B}_e} |\Gamma_{e,j}| \hat{\mathbf{u}}_j \cdot \boldsymbol{n}_j \\ \qquad + \tau_i^p \gamma_e^{-1} \sum_{j \in \mathcal{A}_e} |\Gamma_{e,j}| \tau_j^p \hat{\mathrm{p}}_j - \tau_i^p \hat{\mathrm{p}}_i \Big\}, & \text{for } i \in \mathcal{A}_e, \end{cases} \tag{3.35}$$

with $\boldsymbol{\alpha}_e$, and γ_e being defined as

$$\boldsymbol{\alpha}_e := \sum_{j \in \mathcal{A}_e} |\Gamma_{e,j}| \boldsymbol{\tau}_j, \quad \gamma_e := \sum_{j \in \mathcal{A}_e} |\Gamma_{e,j}| \tau_j^p$$

The same elimination of local degrees of freedom is applied to the boundary operator $\hat{\mathbf{B}}_i$, as explicitly stated in (3.35). With the residuals written only in terms of the global degrees of freedom, the linearization is performed, leading the linear system of equations to be solved at each Newton-Raphson iteration m

$$\mathbb{K}^m \Delta \boldsymbol{\Lambda} = -\mathbf{R}_\Lambda^m, \tag{3.36}$$

where

$$\mathbb{K}^m = \begin{bmatrix} \mathbf{K}_{\hat{u}\hat{u}} & \mathbf{K}_{\hat{u}\hat{p}} \\ \mathbf{K}_{\hat{p}\hat{u}} & \mathbf{K}_{\hat{p}\hat{p}} \end{bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_{\hat{p}} \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{p}} \end{Bmatrix}, \quad \mathbf{U} = \begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \\ \mathbf{p} \end{Bmatrix}, \tag{3.37}$$

Type	$\mathtt{n_{eq}}$	$\mathtt{n_{eq}^{FCFV}}$	$\mathtt{n_{eq}}/\mathtt{n_{eq}^{FCFV}}$
TRI	$9N$	$8N$	1.13
QUA	$6N$	$5N$	1.20
TET	$40N$	$35N$	1.14
HEX	$12N$	$10N$	1.20
PRI	$20N$	$17N$	1.18
PYR	$16N$	$68/5N$	1.18

Table 3.1: Size of the global linear system of equations $\mathtt{n_{eq}}$ per element type in terms of number of nodes N. 2D and 3D elements. A comparison between the hybrid pressure FCFV and the standard FCFV formulations is shown.

with $\mathbf{K}_{ab}$ denoting the tangent matrix obtained by assembling the cell contributions of $(\mathbf{K}_{ab})_e := \partial \mathbf{R}_{e,a}/\partial b$ and $\Delta \circledcirc^m = \circledcirc^{m+1} - \circledcirc^m$ the solution increment from the Newton-Raphson iteration m to $m+1$. The local solution $\mathbf{U}$ is retrieved as a post-process after obtaining the global solution $\boldsymbol{\Lambda}$ at the end of the Newton-Raphson solver. The blocks composing the matrix $\mathbb{K}$, arising from the exact Newton-Raphson linearization, are detailed in Appendix C, Section C.3.

3.6 Computational aspects

This section briefly discusses some computational aspects to be considered when implementing the proposed hybrid pressure FCFV for solving the incompressible Navier-Stokes equation. Attention will be drawn to the comparison of the hybrid pressure FCFV formulation against the standard FCFV formulation presented in Chap. 2.

3.6.1 Size of local and global problems

The cost of each Newton-Raphson iteration is dominated by the cost of solving the global problem. The size of the sparse non-symmetric linear system of equations (3.36) is $\mathtt{n_{eq}} = \mathtt{n_{fa}}\mathtt{n_{sd}} + \mathtt{n_{fa}}$, where $\mathtt{n_{fa}}$ is the total number of edges/faces in the mesh. The standard FCFV formulation on the other hand requires a global system of equation of size $\mathtt{n_{eq}^{FCFV}} = \mathtt{n_{fa}}\mathtt{n_{sd}} + \mathtt{n_{el}}$, with $\mathtt{n_{el}}$ being the total number of elements in the mesh. In Table 3.1, the number of equations $\mathtt{n_{eq}}$, for the hybrid pressure FCFV and standard FCFV approaches, is shown for 2D and 3D element types considering a mesh with a given N nodes. The global system size of the hybrid pressure formulation is around 10%-20% bigger than the one of the standard formulation, as seen in Table 3.1.

The size of each local problem is $\mathtt{n_{sd}}(\mathtt{n_{sd}}+1)+1$ for the hybrid pressure FCFV formulation. The standard FCFV formulation has a size of $\mathtt{n_{sd}}(\mathtt{n_{sd}}+1)$. The extra degree of freedom in the hybrid pressure formulation comes from the elemental pressure p_e that needs to be evaluated.

Type	$\mathtt{n_{nz}}$	$\mathtt{n_{nz}^{FCFV}}$	$\mathtt{n_{nz}}/\mathtt{n_{nz}^{FCFV}}$
TRI	$135N$	$84N$	1.61
QUA	$126N$	$72N$	1.75
TET	$1120N$	$750N$	1.49
HEX	$528N$	$333N$	1.59
PRI	$720N$	$465N$	1.55
PYR	$576N$	$372N$	1.55

Table 3.2: Number of non-zeros $\mathtt{n_{nz}}$ in the global sparse matrix per element type in terms of number of nodes N. 2D and 3D elements. A comparison between the hybrid pressure FCFV and the standard FCFV formulations is shown

For the standard formulation, p_e is eliminated with $\mathrm{p}_e = \rho_e$. The local problems of both formulations can be trivially solved in parallel as no communication is required between cells. Furthermore, solving a local problem does not require solving a linear system of equations of size $\mathtt{n_{sd}}(\mathtt{n_{sd}}+1)+1$ or $\mathtt{n_{sd}}(\mathtt{n_{sd}}+1)$. After solving the global system, (3.36) or (2.56), for the hybrid pressure or standard formulation, the local variables can be recovered in a post-process step, as discussed in sections 3.5 and 2.5.1.

The solution of the global system is performed using the parallel multifrontal sparse direct solver `MUMPS`, from Amestoy et al. (2001, 2019). The interface to the `MUMPS` solver is made through the `PETSc` library of Balay et al. (2023).

3.6.2 Number of non-zeros in the sparse global matrix

A comparison of both hybrid pressure and standard FCFV formulations can be drawn in terms of the amount of memory required to allocate the global sparse matrix. Assuming that the number of exterior edges/faces in the mesh is negligible when compared to the number of interior edges/faces, the number of non-zero entries in the global system of hybrid pressure FCFV formulation is

$$\mathtt{n_{nz}} \simeq \sum_{e=1}^{\mathtt{n_{el}}} (\mathtt{n_{fa}^{\mathit{e}}})^2(\mathtt{n_{sd}}+1)^2 - \mathtt{n_{fa}}(\mathtt{n_{sd}}+1)^2. \tag{3.38}$$

with $\mathtt{n_{fa}^{\mathit{e}}}$ being the number of faces in the element e and $\mathtt{n_{fa}}$ the total number of faces in the mesh. In contrast, the number of non-zero entries of the global problems associated with the standard FCFV formulation is

$$\mathtt{n_{nz}^{FCFV}} \simeq \sum_{e=1}^{\mathtt{n_{el}}} \mathtt{n_{fa}^{\mathit{e}}}\mathtt{n_{sd}}(\mathtt{n_{fa}^{\mathit{e}}}\mathtt{n_{sd}}+2) - \mathtt{n_{fa}}\mathtt{n_{sd}^2}, \tag{3.39}$$

In Table 3.2, the number of non-zeros $\mathtt{n_{nz}}$ in the sparse global matrix, hybrid pressure, and standard FCFV formulations, is shown for 2D and 3D element types considering a mesh with a given N nodes. The number of non-zeros for the hybrid pressure formulation is around 50%-60% bigger than the one for the standard formulation, as seen in Table 3.2.

Chapter 4

Numerical benchmarks for laminar incompressible flows

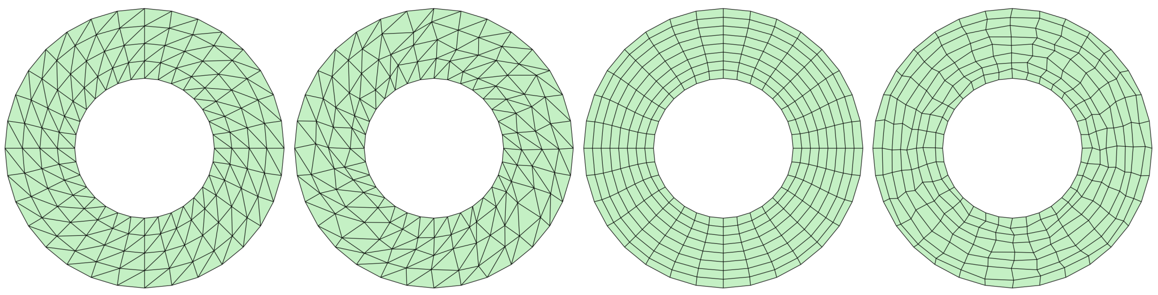
This chapter presents a set of numerical examples selected to test the optimal approximation properties of the presented FCFV methodologies for laminar incompressible flows. Attention will be given to the comparison of the standard FCFV of Chap. 2, published in Vieira et al. (2024)[1], and the hybrid pressure FCFV[2] of Chap. 3. Also, the comparison of the performance of the proposed convective stabilisations in terms of numerical dissipation is performed. Examples include steady and transient cases in laminar regimes for both 2D and 3D settings.

When errors concerning analytical or reference solutions are reported, E denotes the relative error, and $\|\cdot\|_{L^2}$ denotes the $\mathcal{L}^2(\Omega)$ norm for variables lying on the cell-centres and the $\mathcal{L}^2(\Gamma)$ norm for the variables lying on the face-centres.

The values of stabilisation parameters β and τ^p appearing in the numerical diffusive fluxes, (2.50) and (3.14b), and numerical mass flux, in (3.14c), are respectively taken as 10 and 0.1, for all examples. The convective stabilisation parameter ϵ, appearing in the Riemann solvers of Appendix A, is taken as 0.05 for all examples.

[1]The results presented in this chapter regarding the standard FCFV formulation are part of the laminar incompressible Navier-Stokes results published in: L. M. Vieira, M. Giacomini, R. Sevilla, and A. Huerta. **The Face-Centred Finite Volume Method for Laminar and Turbulent Incompressible Flows**. Computer and Fluids. 279:106339, 2024.

[2]The hybrid pressure FCFV results presented in this chapter are part of the paper: M. Giacomini, D. Cortellessa, L. M. Vieira, R. Sevilla, and A. Huerta. **A Hybrid Pressure Formulation of the Face-Centred Finite Volume Method for Viscous Laminar Incompressible Flows**. Submitted to the International Journal for Numerical Methods in Engineering.



(a) Regular Tri. (b) Distorted Tri. (c) Regular Quads. (d) Distorted Quads.

Figure 4.1: Couette flow: First refinement level, namely 16x16, of regular and distorted (a,b) Triangular and (c,d) quadrilateral meshes used for the convergence study.

4.1 Couette flow

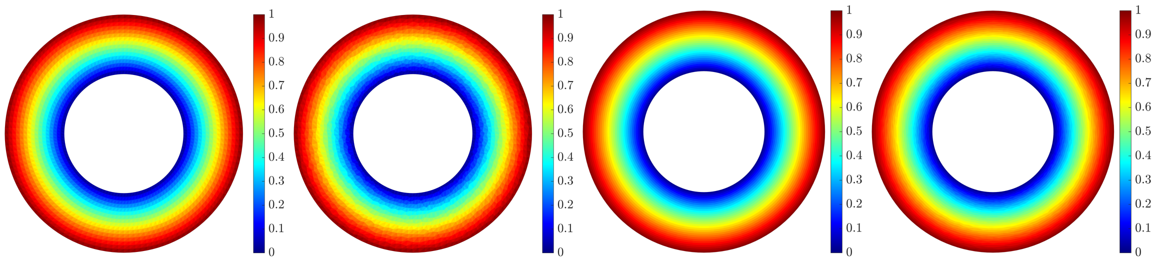
The first example is the so-called Couette flow, see Childs (2011), which involves the flow in an annulus with imposed angular velocity on the whole boundary. The inner and outer radii are $R_i = 1$ and $R_o = 2$, respectively, and the imposed angular velocities are $\Omega_i = 0$ and $\Omega_o = 0.5$, respectively. This example tests the optimal rate of convergence of the standard FCFV and the hybrid pressure FCFV for laminar flows under mesh refinement. In polar coordinates, the exact solution is

$$\begin{cases} u_r = 0, \qquad u_\phi = C_1 r + C_2 \dfrac{1}{r}, \\ p = C_1^2 \dfrac{r^2}{2} + 2C_1C_2 \log(r) - \dfrac{C_2^2}{2r^2} + C, \end{cases} \tag{4.1}$$

where $C_1 = (\Omega_o R_o^2 - \Omega_i R_i^2)/(R_o^2 - R_i^2)$, $C_2 = (\Omega_i - \Omega_o) R_i^2 R_o^2/(R_o^2 - R_i^2)$ and C is a constant such that the pressure at the outer radius is equal to 1. As the exact solution does not depend upon the viscosity, the Reynolds number is selected as $Re = 1$.

Triangular and quadrilateral structured meshes are considered and the effect of cell distortion on the accuracy of the computations is evaluated. Fig. 4.1 shows two structured triangular and quadrilateral meshes and the corresponding meshes where the internal nodes have been randomly moved, as done in Sevilla et al. (2018); Vila-Pérez et al. (2023), to test the effect of the cell distortion. Five meshes are considered for this study. The i-th mesh has a total of $(8 \times 2^i) \times (8 \times 2^i)$ cells for both quadrilateral and triangular grids. To perform a mesh convergence analysis, the meshes are uniformly refined, and the error in all quantities, namely cell velocity, face velocity, gradient of the velocity, and pressure, is measured.

In this example, the HLL convection stabilisation is employed, but further numerical experiments, not reported for brevity, show that the results with the LF or the Roe stabilisation are almost identical. This is expected due to the diffusion-dominated character of the



(a) Regular Tri (b) Distorted Tri (c) Regular Quads (d) Distorted Quads

Figure 4.2: Couette flow: Module of velocity for refinement level 3, namely 64x64, regular and distorted (a,b) triangular and (b,c) quadrilateral meshes with the standard FCFV using the HLL convective stabilization

solution. Due to the simplicity of this example, the steady problem is directly solved in all cases without marching in pseudo-time, using uniform flow as the initial guess. A tolerance of 10^{-12} is imposed for the nonlinear Newton-Raphson iterations (see B.3).

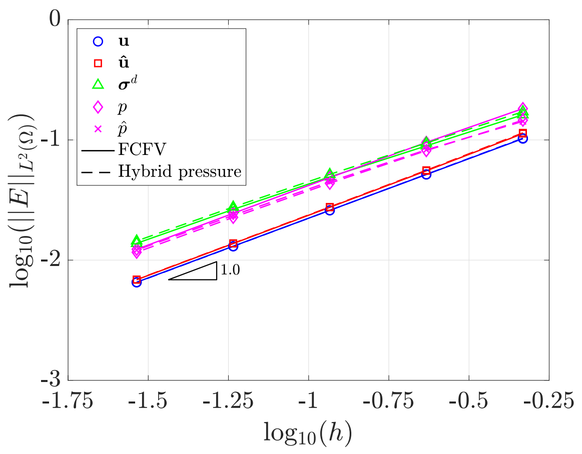
In Fig. 4.2, the module of velocity field solution is shown for the third level of refinement for regular and distorted triangular and quadrilateral meshes with the standard FCFV.

To ensure a fair comparison between the hybrid pressure FCFV formulation and the standard FCFV formulation in terms of h-convergence for all variables, it is preferable to replace the error of the mixed variable $\boldsymbol{L}$ with the error of the deviatoric part of the Cauchy-Stress tensor, denoted as $\boldsymbol{\sigma}^d$. This adjustment is necessary because the definition of $\boldsymbol{L}$ differs between the formulations: in the standard FCFV, $\boldsymbol{L}$ represents the velocity gradient, whereas, in the hybrid pressure FCFV, it corresponds to a scaled version of the deviatoric part of the Cauchy-Stress tensor. Recall that in the standard FCFV formulation, the deviatoric part of the Cauchy-Stress tensor can be recovered using $\boldsymbol{\sigma}^d = (\boldsymbol{L} + \boldsymbol{L}^T)/Re$, considering that the velocity field satisfies the pointwise divergence-free condition exactly in this formulation.

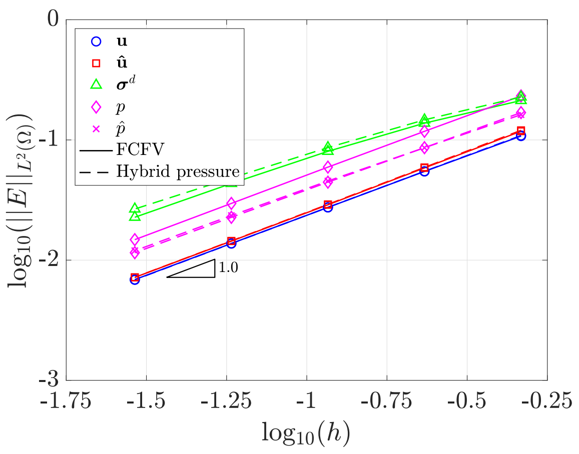
Fig. 4.3 shows the convergence of the error for all variables on regular and distorted triangular grids. The figure displays the optimal error convergence under mesh refinement for both the standard FCFV and the hybrid pressure FCFV with the HLL Riemann solver.

It can be observed, for both standard FCFV and hybrid pressure FCFV, that the effect of grid distortion is minimal in the velocity and hybrid velocity, while slightly less accurate results are obtained for the deviatoric part of the Cauchy-stress tensor. The effect of the distortion on the pressure is less pronounced for the hybrid pressure FCFV formulation.

Similar conclusions can be drawn from the results for quadrilateral grids, shown in Fig. 4.4. Once again, the standard FCFV shows a prevalent loss of accuracy in the pressure field due to cell distortion, whereas the hybrid pressure formulation maintains nearly unchanged pressure accuracy. Furthermore, the effect of cell distortion on pressure accuracy is more pronounced with quadrilateral meshes, particularly in the standard FCFV case. The extra accuracy

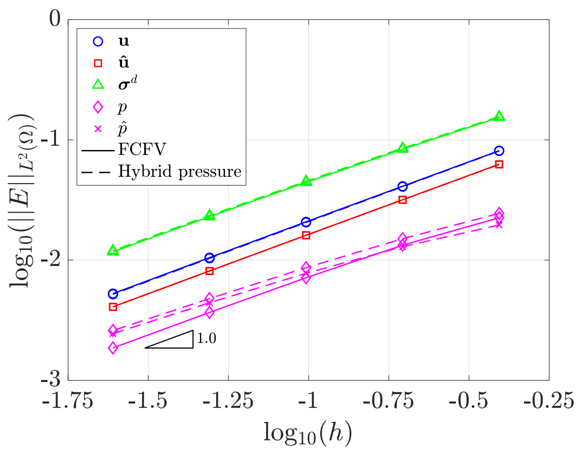


(a) Regular Triangles

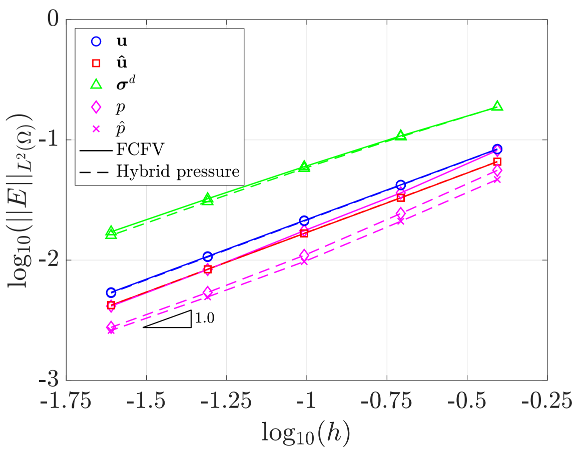


(b) Distorted Triangles

Figure 4.3: Couette flow: Mesh convergence of the error of the cell velocity, face velocity, deviatoric part of the Cauchy stress tensor, pressure, and face pressure for regular and distorted triangular meshes using the HLL convective stabilization. The hybrid pressure FCFV and the standard FCFV are compared.



(a) Regular Quadrilaterals



(b) Distorted Quadrilaterals

Figure 4.4: Couette flow: Mesh convergence of the error of the cell velocity, face velocity, deviatoric part of the Cauchy stress tensor, pressure, and face pressure for regular and distorted quadrilateral meshes using the HLL convective stabilization. The hybrid pressure FCFV and the standard FCFV are compared.

obtained with quadrilateral meshes in this example is due to the nature of the analytical solution, depending only on the radial coordinate, and the specific selection of meshes.

Both formulations present comparable computational costs for a given accuracy on all variables. However, we notice a slightly more accurate pressure field for the hybrid pressure

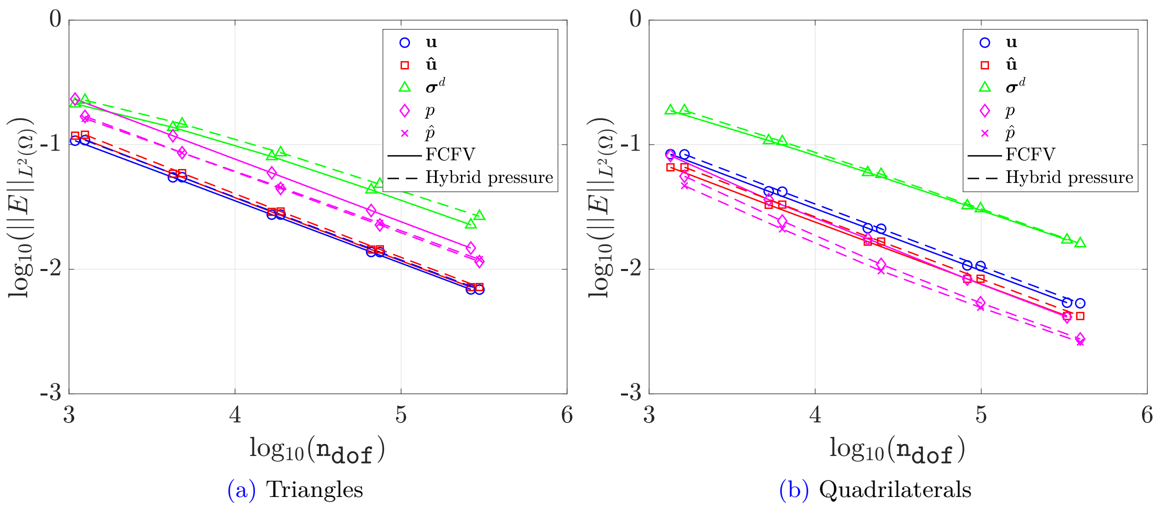


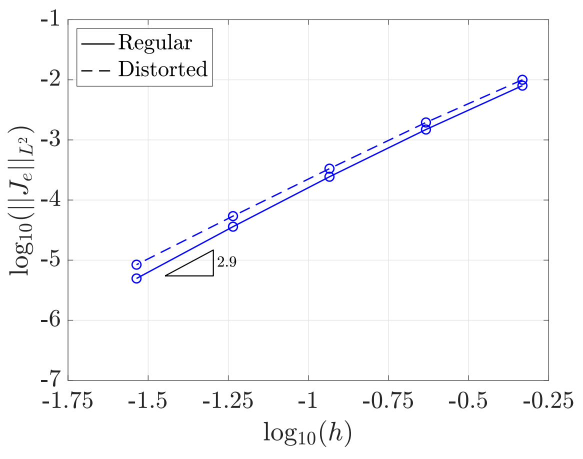
Figure 4.5: Couette flow: Error of the cell velocity, face velocity, deviatoric part of the Cauchy stress tensor, pressure, and face pressure as a function of the number of degrees of freedom for distorted triangular and quadrilateral meshes using the HLL convective stabilization. The hybrid pressure FCFV and the standard FCFV are compared.

formulation for a given number of degrees of freedom. This can be appreciated in Fig. 4.5, where the error on all variables is plotted against the number of degrees of freedom of the global system of equations, namely $\mathtt{n_{dof}}$, for distorted triangular and quadrilateral grids.

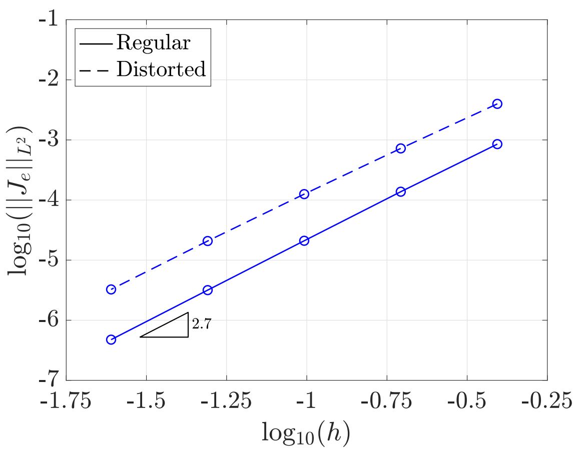
4.1.1 On the hybrid pressure FCFV incompressibility

An important aspect of the hybrid pressure FCFV formulation is the relaxation on the velocity field divergence-free condition, induced by pressure jumps on the numerical mass flux, introduced in (3.14c). Since the formulation is intended for incompressible flows, one can expect that as $h \to 0$ the pressure jumps also tend to zero, and thus the incompressibility condition on the velocity field is fulfilled. We measure the fulfilment of the compressibility condition with the $\mathcal{L}^2(\Omega)$ norm of the cell mass flux J_e, defined in Remark 3.2 and with its discrete form shown in Remark 3.3. Note that J_e for the standard FCFV formulation is exactly fulfilled to zero due to the divergence-free and compatibility conditions.

In Fig. 4.6 the h-convergence of the cell mass flux is shown, and it can be observed the superconvergence of J_e for regular and distorted, triangular and quadrilateral grids. We argue that the superconvergence is attained because J_e is a scaling of the hybrid velocity $\hat{\boldsymbol{u}}$ by the cell face size $|\Gamma_{e,j}|$, see (3.33). Since $\hat{\boldsymbol{u}}$ converges linearly with h and $|\Gamma_{e,j}|$ is directly proportional to h, at least second-order convergence is expected. We also notice that the grid distortion does not affect the order of convergence in all cases. Figures 4.7 and 4.8 show maps of the logarithm of the absolute value of the cell mass flux, J_e, for the first and

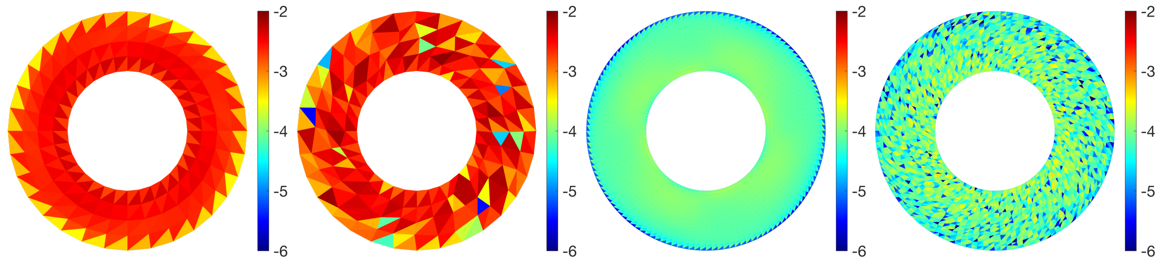


(a) Triangles



(b) Quadrilaterals

Figure 4.6: Couette flow: Mesh convergence of the $\mathcal{L}^2(\Omega)$ norm of the cell mass flux across the faces for regular and distorted triangular and quadrilateral meshes with the hybrid pressure FCFV using the HLL convective stabilization.



(a) Grid 1 - Regular (b) Grid 1 - Distorted (c) Grid 3 - Regular (d) Grid 3 - Distorted

Figure 4.7: Couette flow: Cell mass flux, $\log_{10}(|\mathrm{J}_e|)$, across the faces for regular and distorted triangular grids (a,b) refinement level 1 and (c,d) refinement level 3 with the hybrid pressure FCFV using the HLL convective stabilization.

third levels of grid refinement on regular and distorted triangular and quadrilateral grids. It can be observed that the maximum J_e decreases significantly from the first to the third level of grid refinement. For the first level of refinement, the maximum error is around 10^{-2} for triangular meshes and 10^{-3} for quadrilateral meshes. For the third level, the maximum error reduces to approximately 10^{-4} for triangles and 10^{-5} for quadrilateral meshes. While errors on distorted grids are higher than on regular grids, they remain at least one order of magnitude below the grid size h, which is about 4×10^{-1} and 2×10^{-2} for the first and third refinement levels, respectively. Furthermore, due to the uniqueness of the hybrid velocity, the total mass flux, calculated by summing the contributions of each J_e, is zero at machine precision, confirming the global conservation of mass attained by the method.

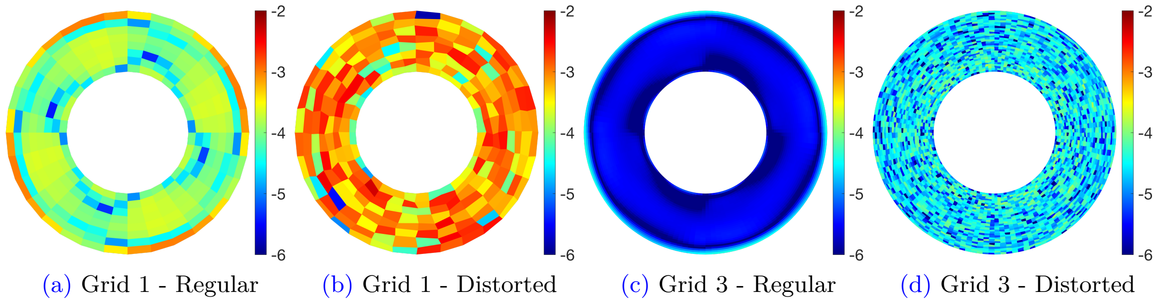


Figure 4.8: Couette flow: Cell mass flux, $\log_{10}(|\mathrm{J}_e|)$, across the faces for regular and distorted quadrilateral grids (a,b) refinement level 1 and (c,d) refinement level 3 with the hybrid pressure FCFV using the HLL convective stabilization.

	48x48	96x96	192x192	384x384	768x768
$\mathtt{n_{el}}$	4,608	18,432	73,728	294,912	1,179,648
h_0	$1.00 \cdot 10^{-2}$	$5.00 \cdot 10^{-3}$	$2.50 \cdot 10^{-3}$	$1.25 \cdot 10^{-3}$	$6.25 \cdot 10^{-4}$
r	1.059	1.028	1.014	1.007	1.003

Table 4.1: Structured triangular grid set for the Lid-Driven Cavity flow.

4.2 Lid-driven cavity flow

The second problem is the lid-driven cavity flow, defined in $\Omega = (0,1)^2$. It is a pure Dirichlet boundary condition problem with the velocity at the boundaries prescribed as

$$\begin{cases} \boldsymbol{u} = (1,\ 0) & \text{for} \quad x_1 \in [0\ 1], \quad x_2 = 1, \\ \boldsymbol{u} = (0,\ 0) & \text{for} \quad x_1 \in [0\ 1], \quad x_2 \in [0\ 1), \end{cases} \tag{4.2}$$

This example compares the accuracy of the different convection stabilizations derived in Appendix A and assesses the mesh distortion's influence on convection-dominated problems for the standard FCFV formulation. In the second stage, the standard FCFV formulation, and the hybrid pressure formulation are compared. To this end, regular and distorted triangular meshes are employed. Five meshes are considered for the convergence study, as described in Table 4.1, where $\mathtt{n_{el}}$, h_0 and r are the grid number of elements, the first layer at wall height, and growth ratio, respectively. Figure 4.9 shows the first regular and distorted triangular meshes.

As in the previous example, the steady problem is directly solved, without marching in pseudo-time, with a 10^{-12} tolerance and uniform flow as the initial guess. Since no analytical solution is available, the accuracy of the computations is measured against a reference solution computed with the Taylor-Hood (Q2Q1) element and the streamline-upwind Petrov-Galerkin (SUPG) method, as in Donea and Huerta (2003). The reference solution is

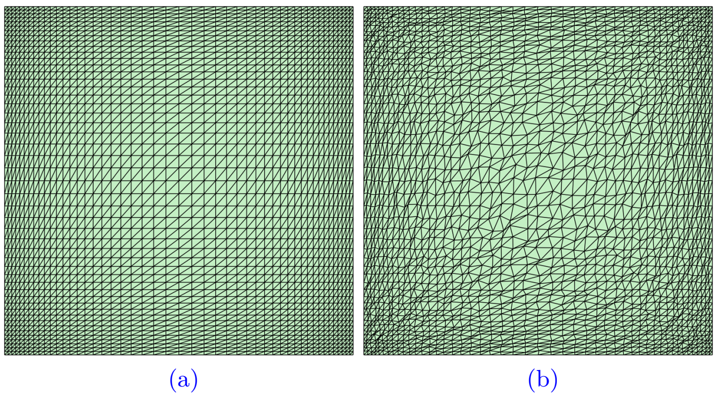


Figure 4.9: Lid-driven cavity flow: First refinement level of (a) Regular and (b) distorted triangular meshes used for the convergence study.

computed on a mesh of 700×700 cells that introduces local refinement near the boundaries, with the height of the first cell being 9.6×10^{-5}. It is worth noting that the reference solution corresponds to the so-called leaky cavity, due to the strong imposition of the incompatible boundary conditions at the top left and right corners. With the FCFV approach, the incompatible boundary conditions do not represent an issue because the velocity is imposed at the barycentre of the faces. To account for this discrepancy between the FCFV and reference solutions, the $\mathcal{L}^2$ norm of the error, for all variables, is measured excluding the regions $[0, 0.05] \times [0.95, 1]$ and $[0.95, 1] \times [0.95, 1]$.

4.2.1 Effect of the Rieman solver

The Reynolds number is taken as $Re = 1000$, and the grid convergence study described is performed using the standard FCFV formulation. The results of the mesh convergence study are depicted in Fig. 4.10 and show again the expected first-order convergence of the error for all variables, as the mesh is refined. For this example, the Roe and HLL convective stabilisations provide very similar accuracy, whereas the LF stabilisation is less accurate. This is observed for all variables and on both regular and distorted meshes. It is also worth noting that using distorted meshes provides accuracy comparable to regular meshes, even for this example where resolving the boundary layers is crucial. To further assess the accuracy of the results, Fig. 4.11 displays the profiles of the velocity and pressure fields along the centrelines, using the fourth regular mesh and different convective stabilisations. The results of the proposed FCFV formulation are compared to the reference solution and illustrate the extra accuracy obtained by using the Roe and HLL stabilisations, when compared to the LF stabilisation.

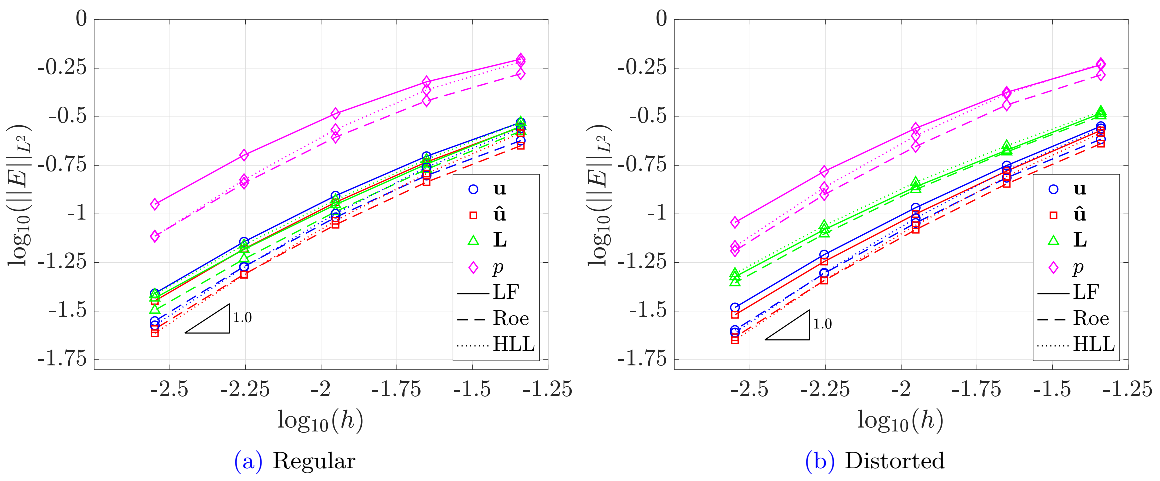


Figure 4.10: Lid-driven cavity flow: Mesh convergence of the error of the cell velocity, face velocity, gradient of the velocity, and pressure for (a) regular and (b) distorted triangular meshes using different convective stabilization with the standard FCFV formulation at $Re = 1000$.

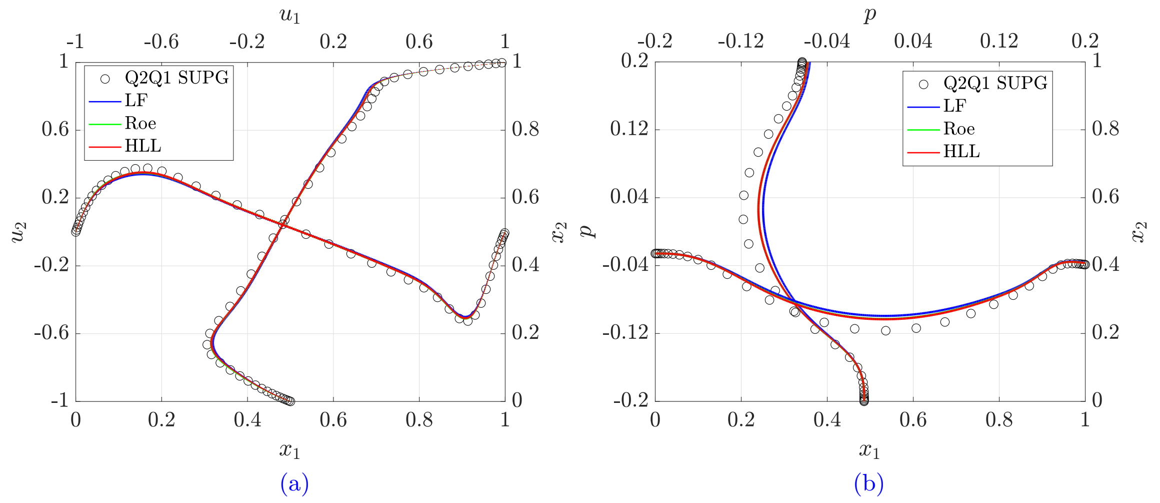


Figure 4.11: Lid-driven cavity flow: Profiles of (a) velocity and (b) pressure along the centrelines, using the fourth regular mesh and different convection stabilisation and standard FCFV formulation.

4.2.2 Cavity flow with the hybrid pressure FCFV

To access the performance of the hybrid pressure FCFV formulation against the standard FCFV formulation we solve the same cavity problem at two Reynolds numbers, namely $Re = [1000, 3200]$, using the convective HLL stabilization.

The results of the mesh convergence study at $Re = 1000$ and $Re = 3200$ are shown in Fig. 4.12, where the error against the Taylor-Hood (Q2Q1) reference is depicted for regular

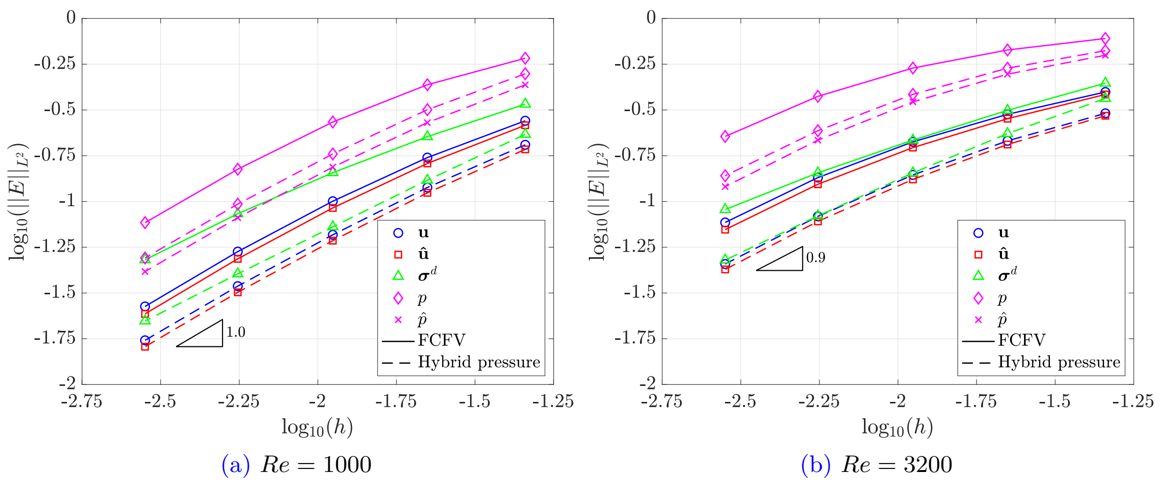


Figure 4.12: Lid-driven cavity flow: Mesh convergence of the error of the cell velocity, face velocity, deviatoric part of the Cauchy stress tensor, pressure, and face pressure for (a) $Re = 1000$ and (b) $Re = 3200$ with the standard FCFV and hybrid pressure FCFV using the HLL convective stabilization and regular grids.

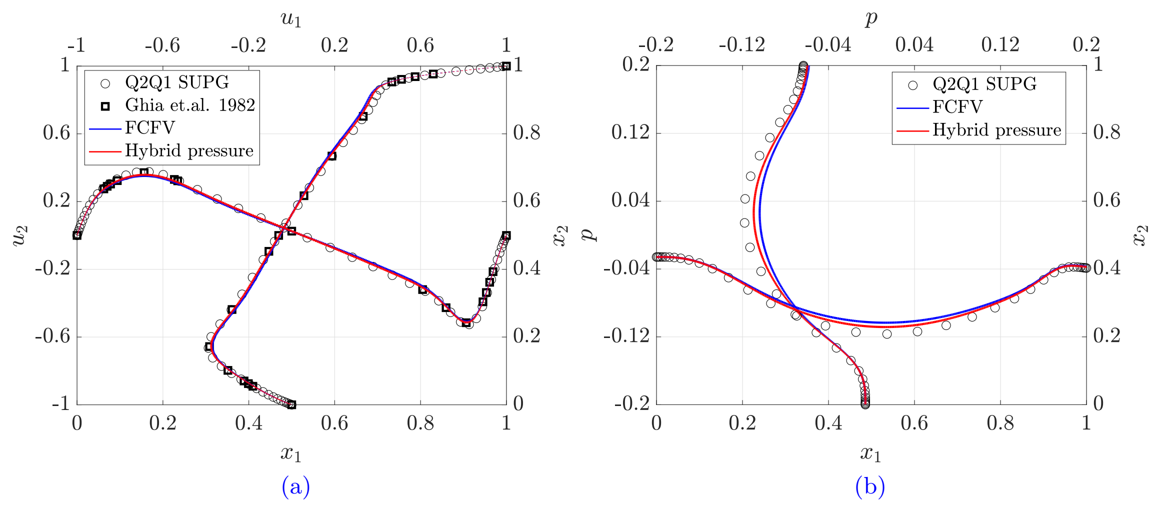


Figure 4.13: Lid-driven cavity flow: Profiles of (a) velocity and (b) pressure along the centrelines, using the fourth regular mesh with the standard FCFV and hybrid pressure FCFV using the HLL convective stabilization at $Re = 1000$.

grids. It can be observed that for all variables, and both Re numbers, the hybrid pressure formulation presents better accuracy. Both formulations deliver an optimal rate of convergence at $Re = 1000$ and an almost optimal convergence rate for $Re = 3200$. The latter can be improved with an additional grid refinement level.

The Fig. 4.13, displays the profiles of the velocity and pressure fields along the centrelines

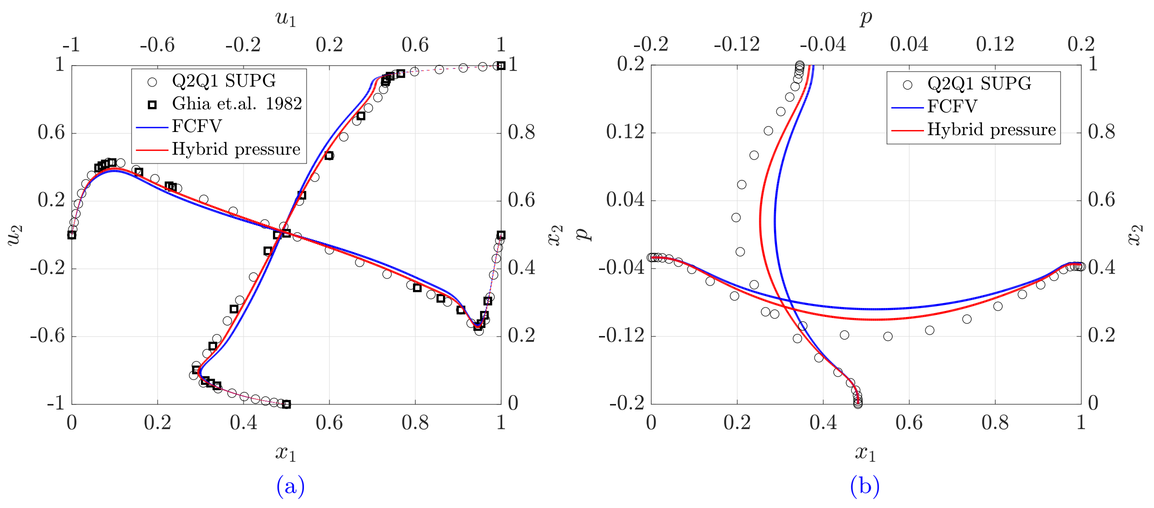


Figure 4.14: Lid-driven cavity flow: Profiles of (a) velocity and (b) pressure along the centrelines, using the fourth regular mesh with the standard FCFV and hybrid pressure FCFV using the HLL convective stabilization at $Re = 3200$.

at $Re = 1000$, using the fourth regular mesh and different HLL convection stabilization. The reference Taylor-Hood (Q2Q1) and results from Ghia et al. (1982) are reported for the velocity profile. The difference between the two FCFV formulations is more pronounced in the pressure profiles. This can be further seen in Fig. 4.14, where the same profiles are shown at $Re = 3200$. Also, at such Re, the velocity profile computed with the standard FCFV deviates from the references at intermediate x_1 and x_2 positions from the wall.

4.3 Numerical diffusion of the convective stabilisations

The next example, taken from Cheung et al. (2015), is used to evaluate the dissipative effect of the three convective stabilisations proposed in Appendix A, using the standard FCFV formulation, without any loss in the generalization of the results for the hybrid pressure FCFV formulation.

The computational domain is $\Omega = (0,1)^2$ and the manufactured solution is given by

$$\begin{cases} u_1 = t^4 \sin(2\pi x_2)\Big(1 - \cos(2\pi x_1)\Big), \\ u_2 = -t^4 \sin(2\pi x_1)\Big(1 - \cos(2\pi x_2)\Big), \\ p = t^4\Big(\cos(\pi x_1) + \cos(\pi x_2)\Big), \end{cases} \tag{4.3}$$

Dirichlet boundary conditions corresponding to the analytical solution are imposed in the whole domain boundary. The Reynolds number is $Re = 10^5$. For such a high Reynolds number, the stabilisation of the FCFV is dominated by the convective stabilisation because the diffusive stabilisation is of the order of $1/Re$, as detailed in (2.51).

The solution is advanced until a final time $T = 1$ using the BDF2 time integrator scheme and $\Delta t = 0.001$. The time step has been selected to ensure that all the error is dominated by the spatial discretisation allowing a clear comparison of the three different stabilisations. To assess the dissipative effect of the stabilisation, the quantity

$$E(T) = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x}, T) \cdot \boldsymbol{u}(\boldsymbol{x}, T) d\Omega, \tag{4.4}$$

proportional to the kinetic energy, is evaluated and compared to the exact value, which is 1.5.

Three meshes are considered to perform the study. The i-th quadrilateral mesh is a structured mesh of $(32 \times 2^i) \times (32 \times 2^i)$ cells, whereas the i-th triangular mesh is obtained after splitting into four triangles each quadrilateral of a mesh with $(16 \times 2^i) \times (16 \times 2^i)$ cells. Therefore, the i-th triangular and quadrilateral meshes contain the same number of cells and a further comparison of the dissipative effect of the stabilisation for different element shapes can be performed.

Table 4.2 reports the relative error on $E(1)$ for three different triangular and quadrilateral meshes using the three different convective stabilisations.

	Triangles			Quadrilaterals		
Mesh	LF	Roe	HLL	LF	Roe	HLL
1	8.5×10^{-2}	5.6×10^{-2}	2.8×10^{-2}	9.4×10^{-2}	6.6×10^{-2}	3.3×10^{-2}
2	4.6×10^{-2}	3.0×10^{-2}	1.5×10^{-2}	5.2×10^{-2}	3.7×10^{-2}	1.8×10^{-2}
3	2.4×10^{-2}	1.5×10^{-2}	7.7×10^{-3}	2.7×10^{-2}	1.9×10^{-2}	9.8×10^{-3}

Table 4.2: Relative error on $E(1)$ for three different triangular and quadrilateral meshes and using the three different convective stabilisations.

The results show that the HLL stabilisation consistently produces the most accurate results, whereas the LF stabilisation always produces the least accurate results. This is observed for all meshes and both element types. More precisely the error using Roe stabilisation induces an error 1.5 times lower than the error using LF. When using the HLL stabilisation the error is two times lower than the error obtained with the Roe stabilisation. This conclusion is observed for both triangular and quadrilateral cells. Comparing the element types marginal differences are observed, with the triangles providing a slightly better accuracy for all meshes and all three stabilisations.

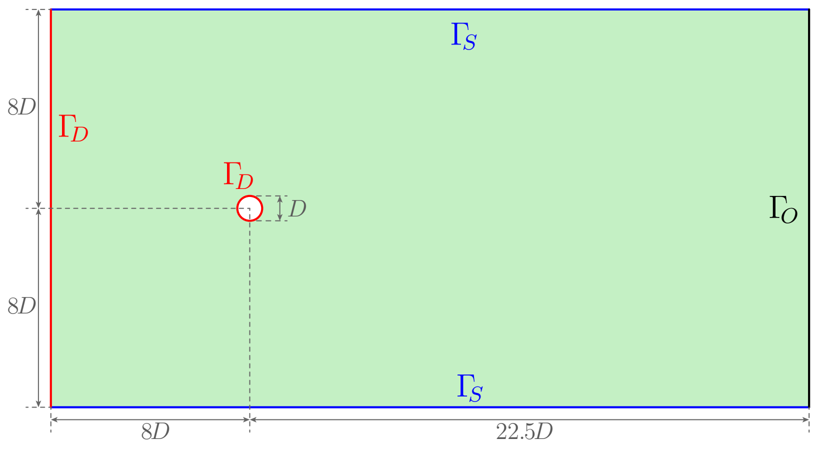


Figure 4.15: Unsteady laminar flow past a circular cylinder: Problem setup.

Grid	$\mathtt{n_{el}}$	h_{wake}	h_{far}	h_0	r	h_{wall}/h_0
1	73,930	0.030 - 0.075	1.0	1.0×10^{-2}	1.1	10
2	135,273	0.020 - 0.050	1.0	0.5×10^{-2}	1.1	10

Table 4.3: Unstructed grids for the Flow Past Cylinder problem

4.4 Unsteady flow past a cylinder

The next example considers the unsteady flow past a circular cylinder of diameter D at $Re = 100$. This classical benchmark is used to assess the performance of the proposed method with different convective stabilisations for a transient laminar flow, using the standard FCFV formulation. The setup of the problem is depicted in Fig. 4.15, including the relevant dimensions and the boundary conditions. The Dirichlet boundary consists of the inflow boundary, where a constant horizontal velocity of magnitude one is imposed, and the cylinder, where a no-slip condition is enforced. The outlet boundary features an outflow condition while in the remaining two boundaries symmetry conditions are imposed.

Two unstructured triangular meshes are considered. The grids are designed with a prescribed inflation layer around the cylinder and a wake refinement with an increasing grid size towards the outflow boundary. In Table 4.3, the detail of the unstructured grids used is shown. Here, $\mathtt{n_{el}}$, h_{wake}/D, h_{far}, h_0, r, and h_{wall}/h_0 denotes the grid number of elements, the variable wake grid size, the far-field grid size, the first layer height at the wall, the inflation layer growth ratio, and the element aspect ratio at the wall, respectively. The second mesh, with 134,801 cells, is shown in Fig. 4.16. For the time integration, the BDF2 scheme is employed with a time-step $\Delta t = 0.1$, and the initial condition is taken as the steady-state solution with $Re = 10$. A tolerance of 10^{-6} is used for the nonlinear problems and the average number of Newton-Raphson iterations across all time steps is two for the

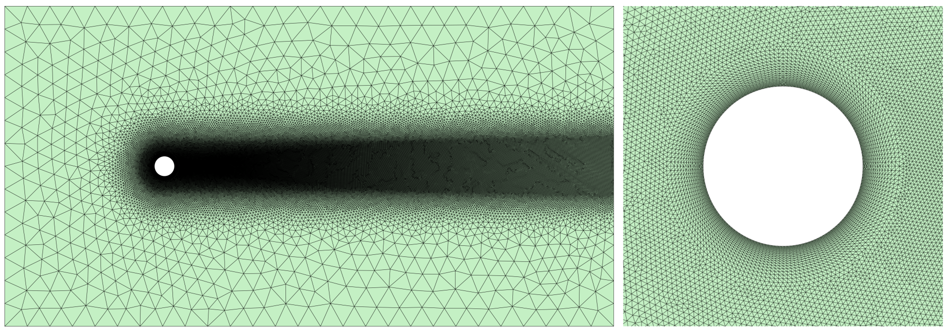


Figure 4.16: Unsteady laminar flow past a circular cylinder: (a) Second mesh with 135,273 cells and (b) detail of the inflation layer.

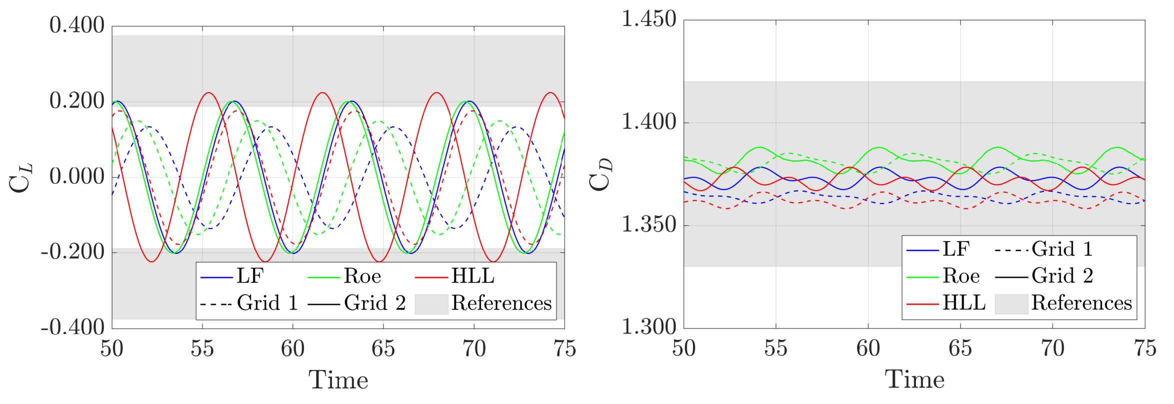


Figure 4.17: Unsteady laminar flow past a circular cylinder: (a) Lift and (b) drag coefficients as a function of time for the two meshes and the three convective stabilisations.

HLL stabilisation and one for the Roe and LF stabilisations.

To assess the accuracy of the simulations, the computed lift (C_L) and drag (C_D) coefficients and the Strouhal number (S_t) are compared against some results found in the literature. The mentioned quantities of interest are respectively computed as $C_L = 2|F_{x_2}|/(\rho U_\infty^2 D)$, $C_D = 2|F_{x_1}|/(\rho U_\infty^2 D)$, and $S_t = fD/U_\infty$. With F_{x_1} and F_{x_2} being the force along the x_1 and x_2 directions, ρ and U_∞ the reference density and velocity, D the reference diameter, and f the lift force oscillatory frequency. Figure 4.17 shows the lift and drag coefficients as a function of time for the two meshes and the three convective stabilisations, where the shadowed area represents the range of values reported in the literature, to cite a few Tezduyar et al. (1991);Tezduyar et al. (1992); Kjellgren (1997); Rajani et al. (2009); Kadapa et al. (2016, 2020). For the references considered the number of elements varies between 5,000 and 18,000, but it is important to note that in these references linear, quadratic, and higher-order

FCFV results						
	C_L amplitude		Mean C_D		S_t	
Stabilisation	Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2
LF	0.142	0.202	1.365	1.373	0.148	0.154
Roe	0.159	0.201	1.381	1.382	0.150	0.155
HLL	0.177	0.225	1.362	1.372	0.154	0.159
Literature results						
Reference	C_L amplitude		Mean C_D		S_t	
Tezduyar et al. (1991)	[0.188, 0.375]		[1.35, 1.41]		[0.156, 0.171]	
Tezduyar et al. (1992)	[0.370, 0.375]		[1.38, 1.40]		[0.166, 0.170]	
Kjellgren (1997)	[0.250, 0.330]		[1.34, 1.37]		[0.160, 0.170]	
Rajani et al. (2009)	0.253		1.335		0.157	
Kadapa et al. (2016)	[0.338, 0.362]		[1.39, 1.42]		[0.165, 0.173]	
Kadapa et al. (2020)	[0.293, 0.338]		–		[0.159, 0.169]	

Table 4.4: Unsteady laminar flow past a circular cylinder: Amplitude of the lift coefficient, mean value of the drag, and Strouhal number using both meshes and the three different stabilisations and numerical reference results.

elements are considered for the simulations. The results reported in the literature are obtained for a variety of numerical schemes including finite volumes, stabilized finite elements, and immersed methods with B-Splines.

A detailed comparison is provided in table 4.4, showing the amplitude of the lift coefficient, the mean value of the drag, and the Strouhal number using both meshes and the three different stabilisations. Results reported in the literature for the three quantities of interest are also included.

For this example, the HLL stabilisation provides the most accurate results. With the first mesh, the lift coefficient is almost within the range reported in the literature, whereas the Roe and LF stabilisations require the second grid to provide similar accuracy. Using the second mesh, the results of all convective stabilisations lie within the range reported in the literature. The extra accuracy of the HLL stabilisation is also observed when comparing the Strouhal number to reference values. The results obtained on the first grid with the HLL stabilisation are almost within the range reported in the references and almost identical to those obtained with LF and Roe in the second mesh. For the mean value of the drag, the results obtained on both meshes and with the three stabilisations lie within the range reported in the literature.

Figure 4.18 shows a snapshot of the magnitude of the velocity and pressure fields at time $t = 65$, when the solution has reached a periodic state, illustrating the ability of the proposed method to capture the von Kármán vortex street.

Figure 4.19 shows a closer snapshot of the magnitude of the velocity and pressure fields at time $t = 65$ along with flow isolines, showing the capability of method in capture complex

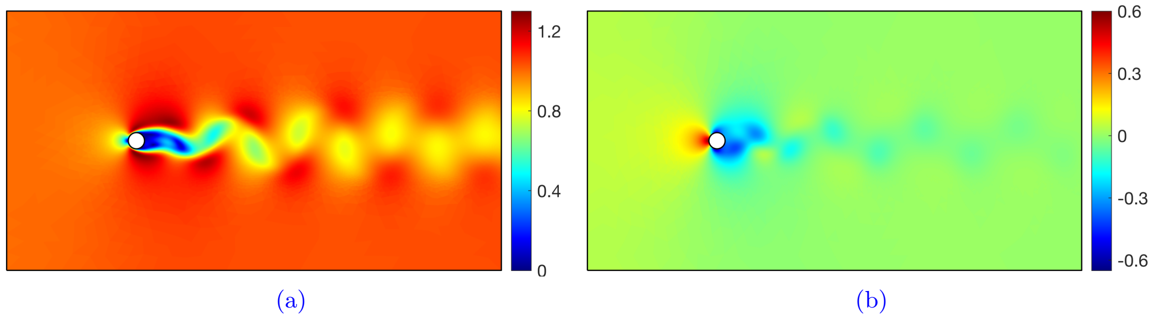


Figure 4.18: Unsteady laminar flow past a circular cylinder: Snapshots of the (a) magnitude of the velocity and (b) pressure at $t = 65$.

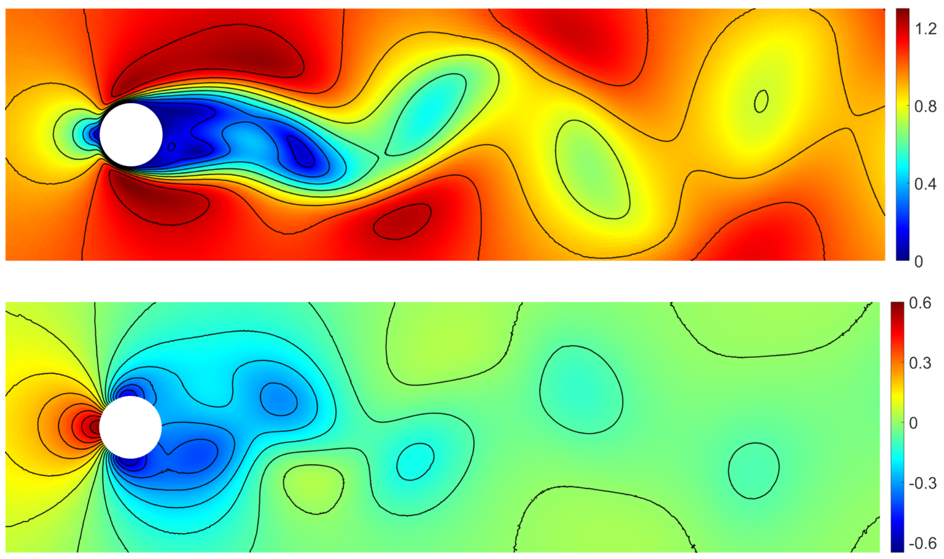


Figure 4.19: (Top) Norm of velocity field and (Bottom) pressure field with isolines, for grid 2 and HLL stabilization at $t = 65$.

flow physics and gradients within the von Kármán vortex street. The displayed simulation corresponds to the standard FCFV solution in the second mesh using the HLL stabilisation.

4.5 3D manufactured source problem

The first 3D example involves a manufactured source problem solved in a computational domain such that $\Omega = (0,1)^3$ and the source term is selected such that the analytical solution is

$$\begin{cases} u_1 = b + (x_3 - y)\sin(x_1 - b), \\ u_2 = a - x_2(x_3 - 0.5x_2)\cos(x_1 - b) - x_2(x_1 - 0.5x_2)\cos(x_3 - b), \\ u_3 = b + (x_1 - x_2)\sin(x_3 - b), \\ \;p = x(1 - x_1) + x_2(1 - x_2) + x_3(1 - x_3) \end{cases} \tag{4.5}$$

where $a = 1.0$, and $b = 0.5$. As it is a synthetic problem, inspired on Roy et al. (2002, 2007); Lodares et al. (2022), with no physical meaning and the exact solution does not depend upon the viscosity, the Reynolds number is selected as $Re = 1$. The analytical traction $\boldsymbol{g}$ is imposed at the Neumann boundary, namely $\Gamma_N = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_3 = 0\}$, and analytical Dirichlet condition $\boldsymbol{u}_D$ is imposed on the remaining boundaries, namely $\Gamma_D = \partial\Omega \setminus \Gamma_N$.

In this example, grid convergence properties of the standard and hybrid pressure FCFV formulation will be studied for the four 3D grid element types, namely tetrahedrons (TET), hexahedrons (HEX), prisms (PRI), and pyramids (PYR). As found out by numerical tests, the results are marginally affected by the convective stabilization type, thus only HLL stabilization is shown for all grid types. As already mentioned in Section 4.1, this is an expected finding given the diffusion domination of the problem that leads to a higher order of magnitude for the diffusive stabilization compared to the convective one, indeed, from Eq. (2.51), the diffusive stabilization is of order β/Re, with $\beta = 10$, whereas the convective stabilization is at maximum of order 1.

The domain is discretized with five regular uniform grids for each element type: tetrahedrons, hexahedrons, prisms, and pyramids. The i-th hexahedrons grid has $2^i \times 2^i \times 2^i$ regular cells, while the Tetrahedrons, Prisms, and Pyramids grids are obtained by splitting each cell of the hexahedrons grid into 24, 4, and 6 cells respectively. In Fig. 4.20, a cut through the second grid refinement level for all element types considered is shown.

The steady-state computation is performed without pseudo-time marching with analytical

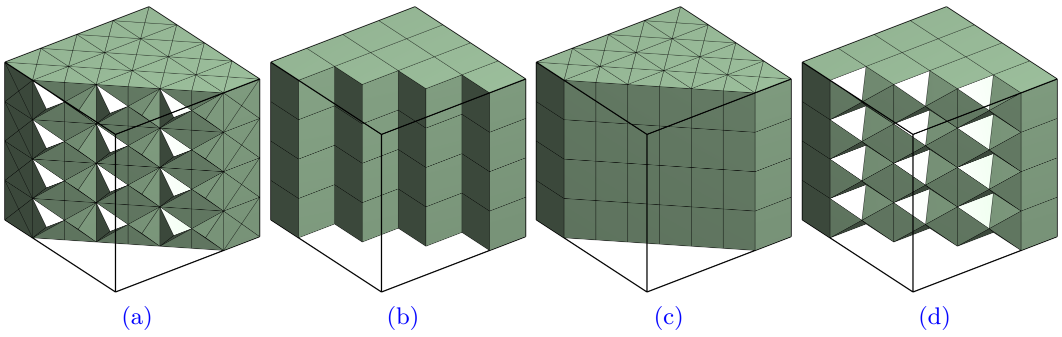
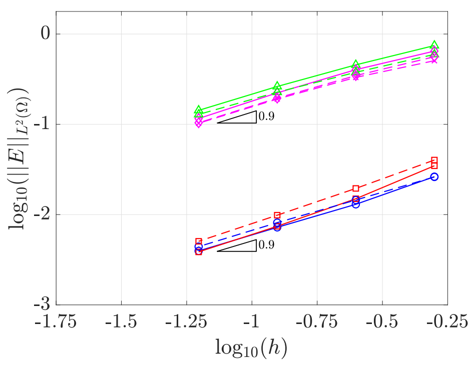
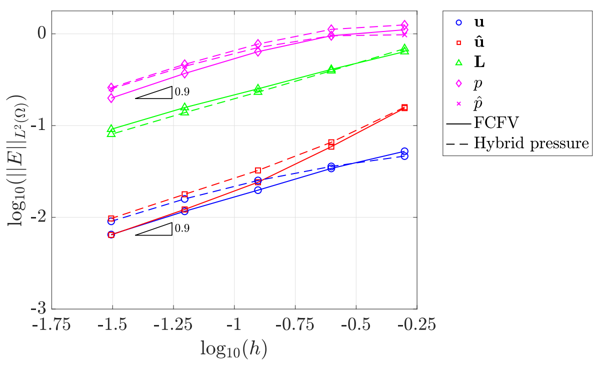


Figure 4.20: 3D manufactured problem: Cut through of the second grid refinement level (a) Tetrahedrons, (b) Hexahedrons, (c) Prisms, and (d) Pyramids meshes.

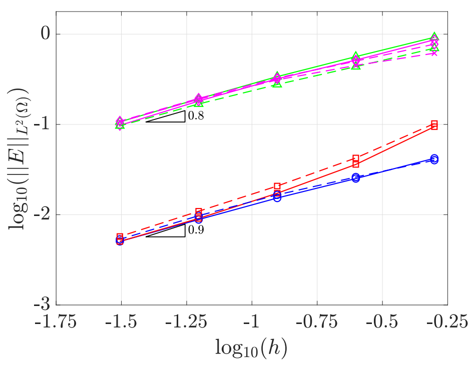


(a) Tetrahedrons

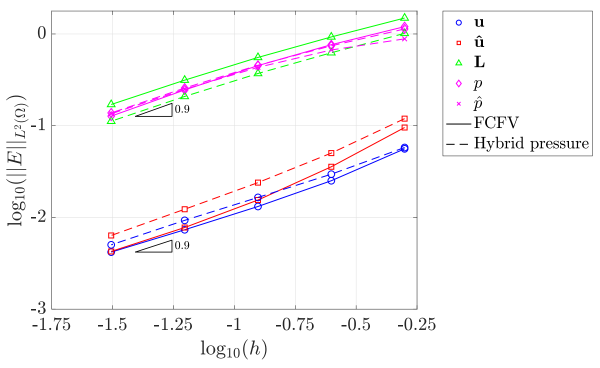


(b) Hexahedrons

Figure 4.21: 3D manufactured problem: Mesh convergence of the error of the cell velocity, face velocity, gradient of the velocity, pressure, and face pressure for (a) Tetrahedrons and (b) Hexahedrons meshes. The FCFV and the hybrid pressure FCFV are compared.



(a) Prisms



(b) Pyramids

Figure 4.22: 3D manufactured problem: Mesh convergence of the error of the cell velocity, face velocity, gradient of the velocity, pressure, and face pressure for (a) Prisms and (b) Pyramids meshes. The FCFV and the hybrid pressure FCFV are compared.

initial conditions and a tolerance of 10^{-12} on the scaled residuals. The mesh convergence study for the FCFV variables, namely cell velocity, face velocity, gradient of the velocity, pressure, and face pressure, on tetrahedrons and hexahedrons grids are shown in Fig. 4.21 for the standard FCFV and hybrid pressure FCFV. The same set of results for Prisms and Pyramids grids are shown in Fig. 4.22. It can be observed that the tetrahedron grid presents more accurate results on the primal variables, followed by the pyramids, prims, and hexahedral. This order is directly related to the number of degrees of freedom of each element type for a given refinement level. The hexahedral grid seems to deliver the most accurate velocity gradient computations, possible due to the cartesian characteristic of the grid and the specific analytical manufactured solution chosen. On the other hand, the hexahedral grid

Grid	$\mathtt{n_{el}}$	h_{wake}	h_{wall}	h_{far}
1	16,108	0.32 - 1.20	0.32	1.2
2	32,550	0.16 - 0.80	0.16	1.2
3	110,567	0.08 - 0.40	0.08	1.2
4	473,710	0.04 - 0.20	0.04	1.2
5	2,204,960	0.02 - 0.10	0.02	1.2

Table 4.5: Unstructed grids for the Flow Past Sphere problem

delivers poorer pressure computation due to the lower number of local and global degrees of freedom compared to the other grid types.

Concerning the standard FCFV and hybrid pressure FCFV, both methods deliver the same theoretical order of convergence for all variables with similar accuracy.

4.6 Flow past a sphere

To access the capabilities of the standard FCFV and the hybrid pressure FCFV to solve a more complex problem, the flow past a sphere will be studied. This is a well-known example in literature and has already been solved with the FCFV formulation for Stokes flow in Sevilla et al. (2018) and Vieira et al. (2020).

The domain of interest consists of $\Omega = \Big([-10D\ 20D]\times[-10D\ 10D]\times[-10D\ 10D]\Big)\setminus\mathcal{B}_1(0)$, where $\mathcal{B}_1(0)$ is a ball with unit diameter D centred at the origin. The Reynolds number is selected as $Re = 20$. In such a regime, the flow field is steady and axisymmetric, as appointed by several authors, to mention a few Taneda (1956); Tabata and Itakura (1998); Johnson and Patel (1999); Schlichting and Gersten (2000); Crivellini et al. (2013). This axisymmetry is explored, and the FCFV solution is computed only in one-fourth of the domain of interest, such that the computational domain reduces to $\Omega = \Big([-10D\ 20D]\times[0\ 10D]\times[0\ 10D]\Big)\setminus\mathcal{B}_1(0)$.

At the inflow boundary, namely $\Gamma_{D,in} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_1 = 0\}$, the Dirichlet boundary condition is imposed such that $\boldsymbol{u}_D = (1,\ 0,\ 0)$. At the outflow boundary, namely $\Gamma_O = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_1 = 20\}$, outflow boundary condition is imposed. At the sphere wall, $\Gamma_{D,wall} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | \left(x_1^2 + x_2^2 + x_3^2\right) = 1\}$, no-slip condition is imposed. On the four remaining boundaries, parallel to the bulk of the flow, symmetry conditions are set.

Five unstructured Tetahedral grids are designed such that a wall size and a wake refinement are set with the element size varying from h_{wall} at the sphere vicinity to h_{wake} at the outflow boundary. Away from the wake zone, a far-field size h_{far} is prescribed. Starting from grid 1, each next grid is a uniform refinement of the previous at the one-fourth sphere wall and wake regions. In Table 4.5, the five grids data are shown. For grid 4, an overview and zoom around the one-fourth sphere are depicted in Fig. 4.23.

No inflation layer is used in those grids, which means that isotropic unstructured uni-

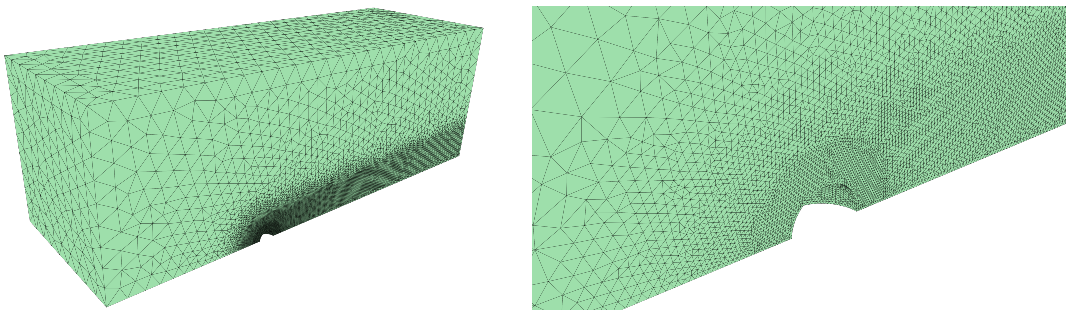

(a) (b)

Figure 4.23: Laminar flow past a sphere: (a) Fourth mesh with 473,710 cells and (b) zoom around one-fourth sphere.

FCFV results					
	C_D				
Formulation	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5
FCFV	2.967	3.028	2.931	2.864	2.810
Hybrid pressure	2.878	2.837	2.808	2.805	2.788
Literature results					
Reference	C_D				
Crivellini et al. (2013)	[2.882, 2.719]				
Tabata and Itakura (1998)	2.724				
Schlichting and Gersten (2000)	2.790				
Roos and Willmarth (1971)	2.820				

Table 4.6: Laminar flow past a sphere: Total drag coefficient at the sphere surface using the five meshes and the two FCFV formulations with HLL convective stabilization. Numerical, Crivellini et al. (2013); Tabata and Itakura (1998), and experimental, Schlichting and Gersten (2000); Roos and Willmarth (1971), reference values are provided.

form elements are employed around the one-fourth sphere surface. This is justified by the fact that at such low Reynolds, the boundary layer is not as sharp as those found at high Reynolds configurations. Also, as shown numerically and experimentally by Johnson and Patel (1999), and experimentally by Taneda (1956); Roos and Willmarth (1971), the flow past the sphere at $Re < 24$ is perfectly laminar and no recirculation or vortex ring is formed downstream of the sphere. Other authors corroborate this with numerical experiments, to mention Crivellini et al. (2013); Tabata and Itakura (1998); Preacher et al. (1970). The steady-state computation is performed without pseudo-time marching with uniform initial conditions and a tolerance of 10^{-12} on the scaled residuals.

The drag force around the sphere is the quantity of interest in this example, which is computed with $C_D = 2|F_{x_1}|/(\rho U_\infty^2 A)$, with F_{x_1} being the drag for along the x_1 direction, ρ and U_∞ the reference density and velocity, and A the one-fourth sphere cross-section area.

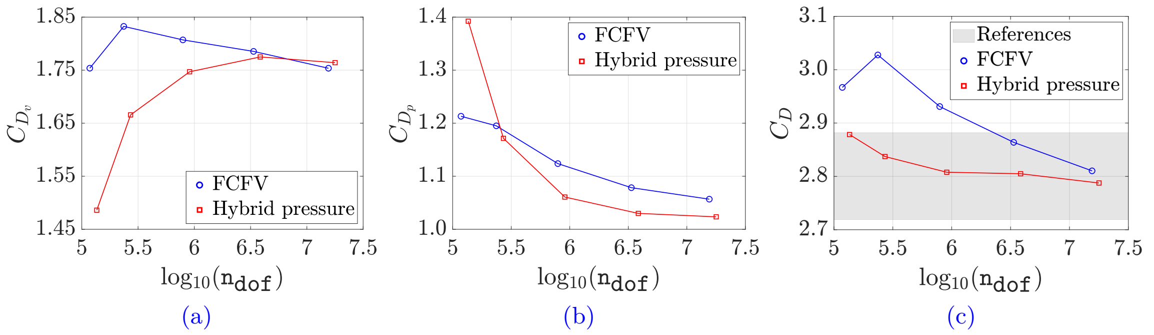
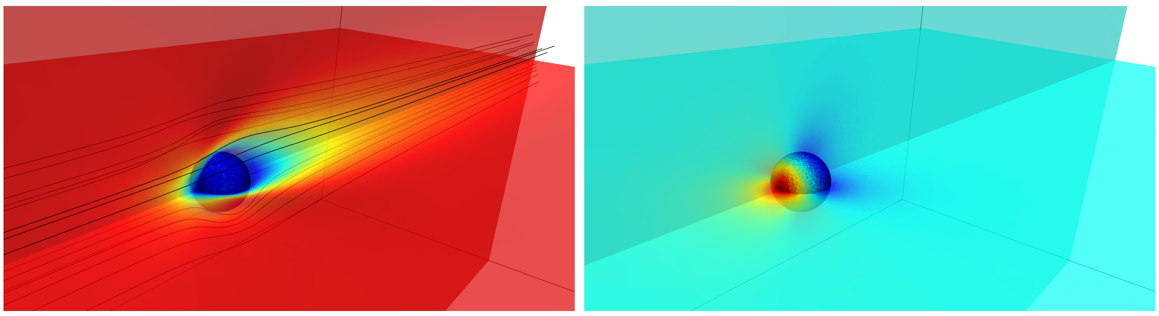


Figure 4.24: Laminar flow past a circular sphere drag coefficients: (a) Viscous drag, (b) Pressure drag, and (c) Total drag as a function of the number of degrees of freedom for standard FCFV and hybrid pressure FCFV with HLL convective stabilization. The reference values range is shown for the total drag.

The latter is computed as $A = \pi D^2/16$. The quantity of interest calculated on the five meshes with the standard FCFV and the hybrid pressure FCFV, using the HLL convective stabilization, is compared against numerical and experimental reference data on Table 4.6. It can be observed that while for the hybrid pressure FCFV formulation, the C_D computed with all grids lies within the reference range, for the standard FCFV the QoI falls inside the reference range only from grids 4 and 5.

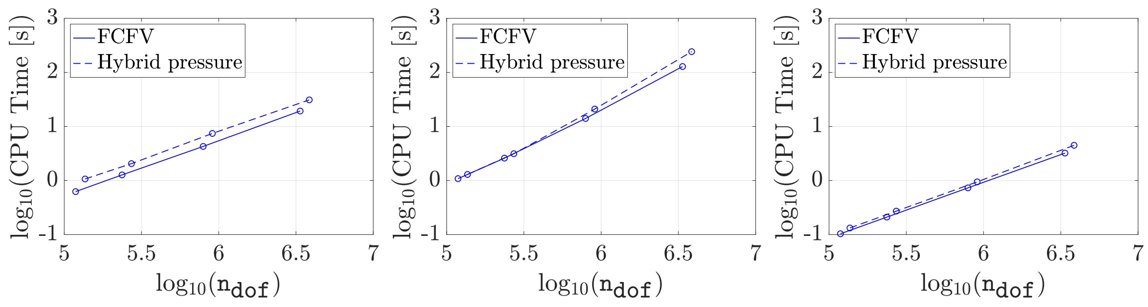
To further illustrate the performance comparison of both formulations to predict the drag coefficient around the sphere, we split the C_D into a viscous contribution C_{D_v}, arising from the deviatoric part of Cauchy-stress tensor, and a pressure contribution C_{D_p}, arising from the volumetric counterpart. In Fig. 4.24 we plot C_{D_v}, C_{D_p} and the total drag coefficient C_D against the number of degrees of freedom in the global system of equations. It can be observed in Fig. 4.24(a) that both standard FCFV and hybrid pressure FCFV deliver similar values for the viscous drag C_{D_v} for the two finer grids. On the other hand, for the pressure drag $C_{D,p}$, depicted in Fig. 4.24(b), considerable deviation between both formulations is observed. The observations suggest that the better performance of the hybrid pressure in predicting the total drag C_D, in Fig. 4.24(c), is dominated by a better prediction of the pressure drag. The additional degrees of freedom related to the pressure render the hybrid pressure formulation superior performance, which seems to be more pronounced in 3D flow settings.

Additionally, from the viscous and pressure drag computations, we observe that asymptotic grid convergence is attained by the hybrid pressure formulation, while the standard FCFV formulation does not appear to have reached grid convergence for the computed quantities of interest (QoI). The magnitude of the velocity field with streamlines and the pressure field around the sphere and downstream the wake is shown in Fig. 4.25 for the hybrid pressure formulation on the fourth grid. Clearly, the method can reproduce the flow



(a) (b)

Figure 4.25: Laminar flow past a circular cylinder: detail of the (a) magnitude of the velocity field with streamlines and (b) pressure field, in the vicinity of the sphere on grid 4 with the hybrid pressure FCFV and HLL convective stabilization .



(a) Build and assemble (b) Global solve (c) Local problem

Figure 4.26: Average CPU time: (a) Building and assembling of the global matrix, (b) Solving the global system of equations, and (c) Solving the local problem as a function of the number of degrees of freedom for standard FCFV and hybrid pressure FCFV.

features of the problem.

For a better comparison of the standard FCFV and hybrid pressure FCFV in terms of computational cost, in Fig. 4.26, the average CPU time for the build and assemble of the global matrix, for the solution of the global system of equations, and for the solution of the local problems is depicted for the first four grids. While the build and assemble as well as the local problem solution are done serially, the solution of the global system of equations is performed in a parallel manner using the MUltifrontal Massively Parallel sparse direct Solver (MUMPS) with 16 MPI processors. From the build and assemble CPU time results, in Fig. 4.26(a), we can observe that the extra degrees of freedom in the hybrid pressure formulation, and consequently more matrix blocks to be built and assembled, slightly increase the computational cost when compared to the one for the standard FCFV. The local problem solution, in Fig. 4.26(c) has virtually the same computational cost in both formulations.

The solution of the global system of equations, handled by MUMPS, shown in Fig. 4.26(b), appears to become more costly for the hybrid pressure formulation as the number of degrees of freedom exceeds 10^6. This is due to the increased number of non-zeros in the sparse matrix of the hybrid pressure global system and sparsity pattern. However, for the example presented, we can see that at a much lower cost, the hybrid pressure formulation delivers more accurate QoI values than the standard FCFV, indeed, the QoI results of the hybrid pressure FCFV with the grid 3 are comparable to those of the standard FCFV with grid 5, as observed in Table 4.6 and Fig. 4.24(c).

Chapter 5

Numerical benchmarks for turbulent incompressible flows

This chapter presents a set of numerical examples selected to test the optimal approximation properties of the standard FCFV methodology, described in Chap. 2, for turbulent incompressible flows, modelled with the RANS-SA equations[1]. The performance comparison of the proposed convective stabilisations in terms of numerical dissipation is performed. Attention will also be given to the comparison of the monolithic and staggered approaches to solving the RANS-SA coupling and their behaviour under the time relaxation schemes proposed in Appendix B. Examples include steady and pseudo-transient cases in turbulent regimes for both 2D and 3D settings.

When errors concerning analytical or reference solutions are reported, E denotes the relative error, and $\| \cdot \|_{\mathcal{L}^2}$ denotes the $\mathcal{L}^2(\Omega)$ norm for variables lying on the cell-centres and the $\mathcal{L}^2(\Gamma)$ norm for the variables lying on the face-centres.

The value of stabilisation parameter β, appearing in the numerical diffusive fluxes, (2.20b) and (2.21b), is taken as 10 for all examples. The convective stabilisation parameters ϵ and $\tilde{\epsilon}$ appearing in the Riemann solvers of Appendix A, are taken as 0.05 and 0.01, respectively, for all examples.

5.1 2D Manufactured source problems

Devising a manufactured source problem to verify the implementation of the RANS equations coupled with the Spalart-Allmaras turbulence model has been studied by several authors. While the use of trigonometric-based manufactured solutions (MSs), that do not mimic the

[1]Some of the results presented in this chapter are from the turbulent incompressible RANS-SA results published in: L. M. Vieira, M. Giacomini, R. Sevilla, and A. Huerta. **The Face-Centred Finite Volume Method for Laminar and Turbulent Incompressible Flows** Computer and Fluids. 279:106339, 2024.

realistic behaviour of the flow field in the studied regime, is widely accepted in the community, as explored by Roy et al. (2002, 2007); Navah and Nadarajah (2018), some authors focus on developing MSs that resemble realistic flow fields, to cite a few Eça et al. (2007); L. Eça and Pelletier (2007); Oliver et al. (2012); Eça et al. (2016); Navah and Nadarajah (2020). Although these solutions may lack physical meaning, when viewed as a purely mathematical exercise, trigonometric-based functions can be employed such that all terms of the RANS-SA equations are engaged, enabling the verification of code correctness and the theoretical order of accuracy.

On the other hand, creating realistic RANS-SA manufactured problems presents numerical challenges due to the difficulty in designing a solution that aligns with both the non-linear behaviour of the Spalart-Allmaras model near the wall and the characteristics of the wall-bounded turbulent velocity profile, as extensively discussed by Eça et al. (2007); L. Eça and Pelletier (2007); Navah and Nadarajah (2020). Despite these challenges, such manufactured solutions can offer deeper insights into the behaviour of RANS solvers under realistic flow conditions, such as the accurate computation of boundary layers in high Reynolds number turbulent flows.

Given the above considerations, in the next two sections, we propose using two manufactured source problems to illustrate the behaviour of the FCFV RANS-SA solver on distorted and stretched grids. The grids used in these examples serve as a twofold mimic of typical CFD computation grids: they are unstructured with some distortion in the bulk flow and highly stretched near the wall within the boundary layer. In both manufactured source examples, the HLL Riemann solver from Appendix A is used without any loss of generality, as further tests have shown minimal differences in numerical diffusion between the Riemann solvers for the two cases considered.

5.1.1 Manufactured problem MS1: Effect of grid distortion

Following the manufactured solution proposed by Peters and Evans (2019) and the ideas of using smooth trigonometric functions to build the MSs, as in Lodares et al. (2022); Tong et al. (2017); Tiberga et al. (2020), we devise this MS to mimic a bulk flow computation with regular and distorted grids. The problem is solved in a computational domain such that $\Omega = (0,1)^1$ and the source term is selected such that the analytical solution is

$$\begin{cases} u_1 = -2x_1^2 e^{x_1}(x_2 - x_2^2)(2x_2 - 1)(x_1 - 1)^2 \\ u_2 = -x_1 x_2^2 e^{x_1}(x_1^2 + 3x_1 - 2)(x_1 - 1)(x_2 - 1)^2 \\ \;p = x_1(1 - x_1) + x_2(1 - x_2) \\ \;\tilde{\nu} = \tilde{\nu}_{\max} \sin(\pi x_1) \sin(\pi x_2) \end{cases} \tag{5.1}$$

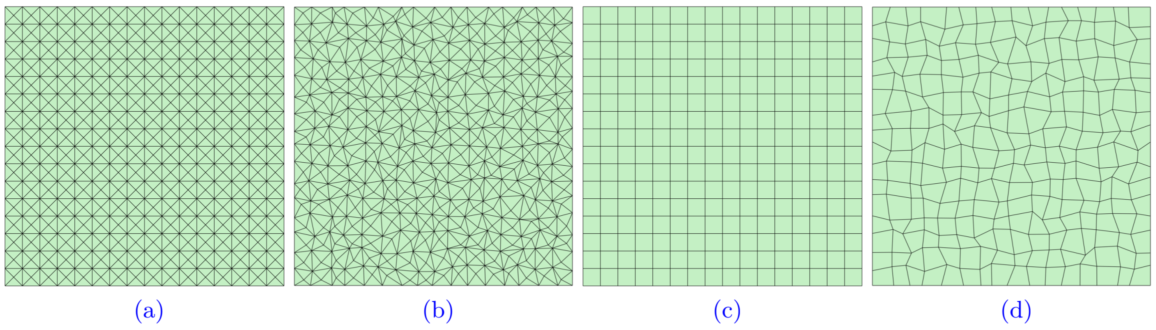


Figure 5.1: 2D manufactured source problem MS1: First refinement level of regular and distorted (a,b) triangular and (c,d) quadrilateral meshes used for the convergence study.

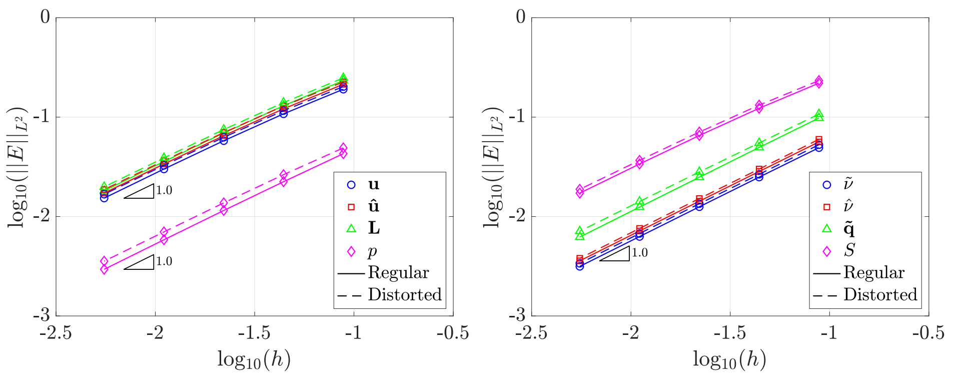
where $\tilde{\nu}_{\max} = 10$. As it is a synthetic problem with no physical meaning and the exact solution does not depend upon the viscosity, the Reynolds number is selected as $Re = 1$. The analytical traction $\boldsymbol{g}$ is imposed at the Neumann boundary, namely $\Gamma_N = \{(x_1, x_2) \in \mathbb{R}^3 | x_2 = 0\}$, and analytical Dirichlet conditions $\boldsymbol{u}_D$ and $\tilde{\nu}_D$ are imposed on the remaining boundaries, namely $\Gamma_D = \partial\Omega \setminus \Gamma_N$.

The domain is discretized with five regular and distorted triangular and quadrilateral grids. The i-th quadrilateral grid has $(8 \times 2^i) \times (8 \times 2^i)$ regular cells, while the grid with triangles is obtained by dividing each cell of the quadrilateral grid into 4 cells. In Fig. 5.1, the first grid refinement level is depicted.

No pseudo-time marching scheme is required for the computations, instead, a pure Newton-Raphson solver is performed with analytical initial conditions and a tolerance of 10^{-12} on the scaled residuals. The monolithic strategy described in Section 2.4.1 is used to solve the RANS-SA coupling. For details on the residuals used for convergence see B.3.

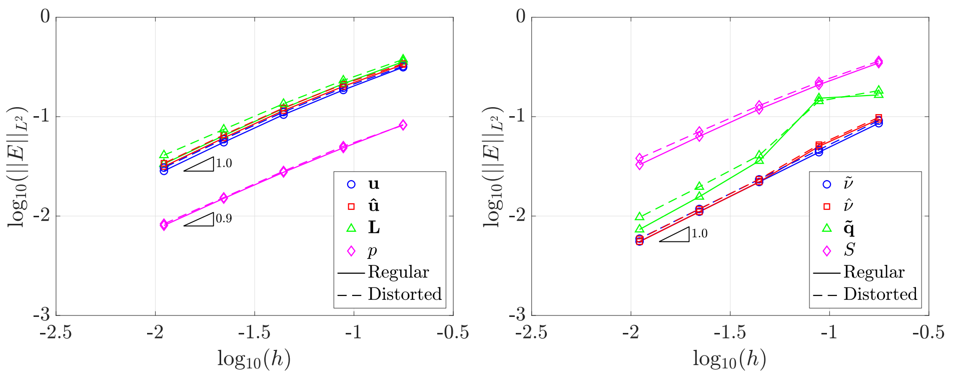
The mesh convergence for the RANS flow variables, namely cell velocity, face velocity, the gradient of the velocity, and pressure, for the triangular regular and distorted grids, is shown in Fig. 5.2(a). In Fig. 5.2(b) the mesh convergence of the variables related to the Sparlat-Allamaras equation discretization, namely cell turbulent variable, face turbulent variable, and gradient of the turbulent variable, are shown. Additionally, the grid convergence of the vorticity magnitude, S, is displayed. Assuring that the vorticity magnitude converges with the same order of magnitude as all FCFV variables is fundamental since this quantity, computed from the velocity gradient (see Remark 2.3), is used to calculate the Sparlat-Allamaras source terms.

From the results in Fig. 5.2 we can observe that the grid distortion effect is minimal in the order of convergence and accuracy of the results. As expected in both cases all variables converge with the theoretical first-order rate. Similar conclusions can be drawn for the quadrilateral grids, with the results shown in Fig. 5.3. The pressure field is more accu-



(a) RANS (b) SA

Figure 5.2: 2D manufactured source problem MS1: Mesh convergence of the error of (a) the cell velocity, face velocity, gradient of the velocity, and pressure, and (b) cell turbulent variable, face turbulent variable, gradient of the turbulent variable, and vorticity magnitude for regular and distorted triangles.



(a) RANS (b) SA

Figure 5.3: 2D manufactured problem MS1: Mesh convergence of the error of (a) the cell velocity, face velocity, gradient of the velocity, and pressure, and (b) cell turbulent variable, face turbulent variable, gradient of the turbulent variable, and vorticity magnitude for regular and distorted quadrilaterals.

rately captured for triangles and quadrilateral grids simply because the chosen manufactured pressure has a much simpler analytical expression than the other flow variables.

5.1.2 Manufactured problem MS2: Effect of grid stretching

In this section we introduce a modified version of the manufactured solution proposed by Eça et al. (2007); L. Eça and Pelletier (2007), to verify wall-bounded turbulent incompressible flows. The FCFV behaviour under stretched grids, commonly used in CFD applications, will be studied. Thus, the modified version proposed is solved in a computational domain

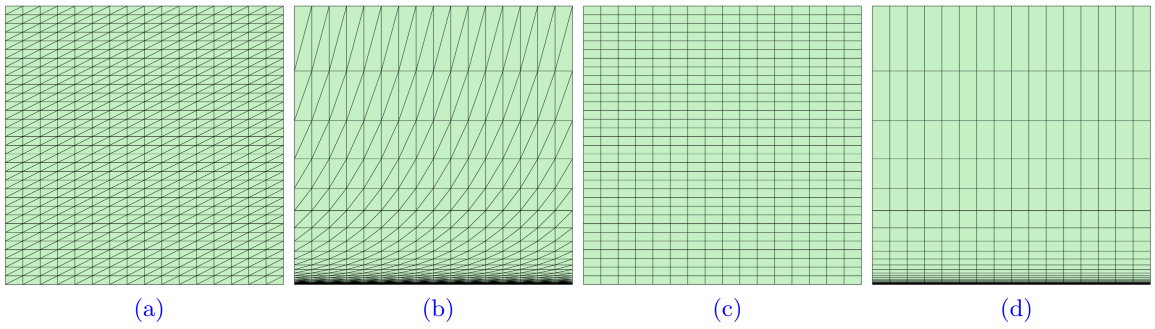


Figure 5.4: 2D manufactured source problem MS2: First refinement level of regular and stretched ($s = 1000$) (a,b) triangular and (c,d) quadrilateral meshes used for the convergence study.

such that $\Omega = \big([0.6\ 1.0] \times [0.0\ 0.4]\big)$ and the source term is selected such that the analytical solution reads

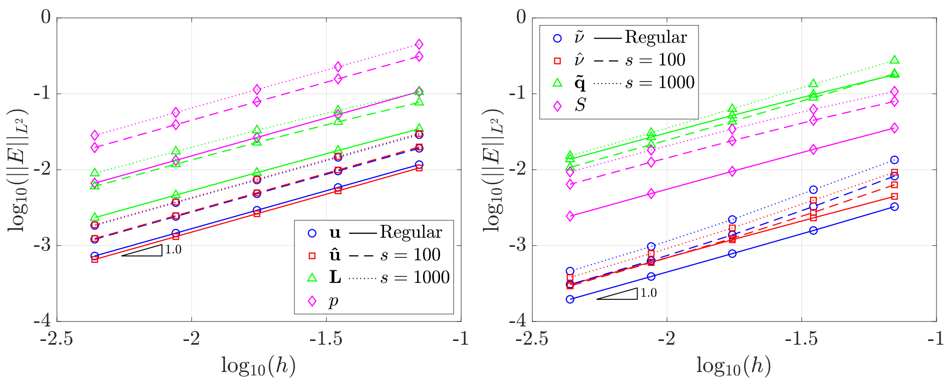
$$\begin{cases} u_1 = erf(\eta) \\ u_2 = \dfrac{1}{\sigma\sqrt{\pi}}(1 - e^{-\eta^2}) \\ p = 0.5 \log(2x_1 - x_1^2 + 0.25) \log(4x_2^3 - 3x_2^2 + 1.25) \\ \tilde{\nu} = \tilde{\nu}_{\max}(\theta + \sqrt{2}\ \eta_\nu e^{0.5-\eta_\nu^2}) \end{cases} \tag{5.2}$$

where $\eta = \sigma x_2/x_1$, and $\eta_\nu = \sigma_\nu x_2/x_1$. With $\sigma = 4$, $\sigma_\nu = 2.5\sigma$, $\tilde{\nu}_{\max} = 10^3$, and $Re = 10^6$. The modification in the baseline version of Eça et al. (2007); L. Eça and Pelletier (2007) is the introduction of a threshold parameter $\theta = 10$.

Despite efforts to devise a physically sound manufactured solution, as noted by Navah and Nadarajah (2020), the proposal by Eça et al. (2007); L. Eça and Pelletier (2007) does not replicate the logarithmic layer in the velocity profile, instead resembling a laminar profile. It also fails to capture the expected behaviour of the SA non-linear source functions near the wall, which can lead to numerical convergence issues and a loss of convergence order in grid refinement tests. Consequently, the threshold parameter θ is introduced as a modification to mitigate numerical issues associated with the SA non-linear functions near the wall.

The analytical normal diffusive flux, namely $\boldsymbol{g}_d$, is imposed at the Neumann boundary, namely $\Gamma_N = \{(x_1, x_2) \in \mathbb{R}^3 | x_1 = 1\}$. With $\boldsymbol{g}_d$ standing for the diffusive part of the Neumann traction $\boldsymbol{g}$. This is because numerical experiments have shown that imposing the analytical Neumann traction $\boldsymbol{g}$ leads to spurious oscillations at the outflow boundary. The analytical Dirichlet conditions $\boldsymbol{u}_D$ and $\tilde{\nu}_D$ are imposed on the remaining boundaries, namely $\Gamma_D = \partial\Omega \setminus \Gamma_N$.

The domain is discretized with five regular and stretched triangular and quadrilateral grids. In Fig. 5.4, the first grid refinement level is depicted for regular and stretched ($s = 1000$) triangular and quadrilateral grids. The i-th quadrilateral grid has $(8 \times 2^i) \times (16 \times 2^i)$



(a) RANS (b) SA

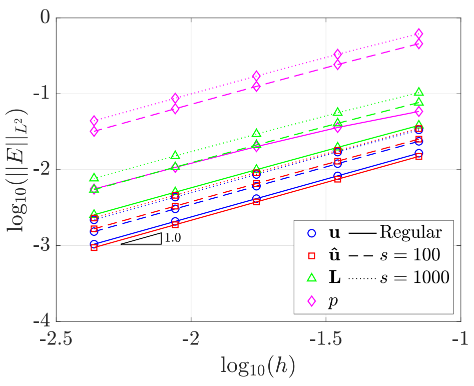
Figure 5.5: 2D manufactured source problem MS2: Mesh convergence of the error of (a) the cell velocity, face velocity, gradient of the velocity, and pressure, and (b) cell turbulent variable, face turbulent variable, gradient of the turbulent variable, and vorticity magnitude for regular and stretched triangles.

regular cells, while the grid with triangles is obtained by dividing each cell of the quadrilateral grid into 2 cells. The grid stretching is characterized by the stretching factor s, which is the grid aspect ratio at the bottom boundary. Two stretching factors are considered, $s = [100,\ 1000]$. For the stretched grids, the cell size is increased from the wall with a constant growth ratio.

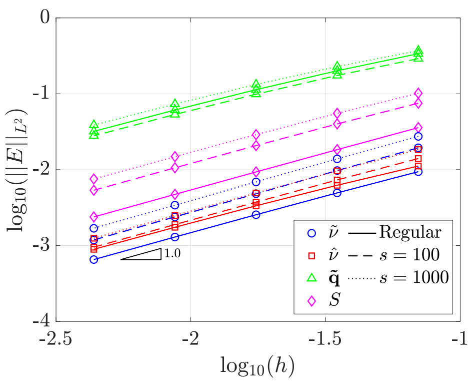
As in the previous example, the steady-state computation is performed without pseudo-time marching with analytical initial conditions and a tolerance of 10^{-12} on the scaled residuals. The monolithic strategy described in Section 2.4.1 is used to solve the RANS-SA coupling.

The grid convergence of the RANS and the SA discrete variables is shown in Fig. 5.5 for regular and stretched triangular grids. As the grid is stretched, the accuracy of all RANS variables, Fig. 5.5(b), is reduced. At first glance, this may sound counterintuitive, however, this is because the analytical velocity and pressure fields do not fully resemble a turbulent wall-bounded flow field, where the gradients are clustered at the wall. Instead, as stressed by Navah and Nadarajah (2020), the velocity profile resembles the one of a laminar flow, having the broader zone of gradients that are better captured with uniform grids. This loss of accuracy is more pronounced in the pressure field since its gradient is more spread in the domain. For the Sparlat-Allmaras variables, a similar effect is observed for the turbulent variable, however, its gradient is better captured with the grid with a stretching factor $s = 100$. This is because the gradients of the turbulent variable are closer to the wall, where the cells are clustered, and the stretching factor of $s = 100$ seems to offer a better balance between wall and bulk flow grid refinement.

In spite of the loss in accuracy, no matter the stretching factor, the order of convergence



(a) RANS



(b) SA

Figure 5.6: 2D manufactured source problem MS2: Mesh convergence of the error of (a) the cell velocity, face velocity, gradient of the velocity, and pressure, and (b) cell turbulent variable, face turbulent variable, gradient of the turbulent variable, and vorticity magnitude for regular and stretched quadrilaterals.

remains unchanged at the theoretical first-order for all RANS and SA variables, showing the robustness of the solver in dealing with stretched grids. Similar conclusions are drawn for the quadrilateral grids, with results shown in Fig. 5.6. The apparent better accuracy for the triangular grids compared to the quadrilateral ones is due to the extra degree of freedom added within each quadrilateral.

5.2 Turbulent flow in a channel

To verify the FCFV RANS-SA implementation and its capability to capture the turbulent flow mean velocity profile, modelled by the law of the wall, introduced by Von Kármán (1939), or single equation formulae extensively discussed in Zhang et al. (2022), the wall-bounded turbulent flow in a channel is studied. This problem is an object of study by several authors in the realm of DNS computations, for example, Moser et al. (1999); del Álamo and Jiménez (2003); Lee and Moser (2015). This is because, besides having a simple setup, the problem provides important insights into the confined turbulent flow features and behaviour, which are present in many CFD applications.

The HLL convective stabilization will be employed and the FCFV solver behavior under the Newton-Raphson-based strategy, introduced in Appendix B.2, will be studied.

The Reynolds number in terms of the friction velocity u_τ is chosen as $Re_\tau = 550$, where $Re_\tau = u_\tau L/\nu$. For this example, the reference length L is half the channel height H, such that $H = 2L$, as done in Moser et al. (1999); del Álamo and Jiménez (2003); Lee and Moser (2015); Kessels (2016).

The computational domain is such that $\Omega = \Big([0\ 1]\times[0\ 2]\Big)$. As a pressure-driven turbulent

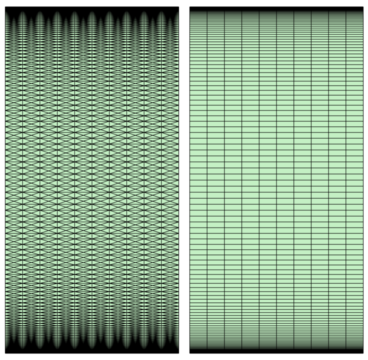


Figure 5.7: Turbulent flow in a channel: grid refinement level 1 for (a) Triangular and (b) quadrilateral meshes.

flow between parallel walls, it is modelled with the help of periodic boundary conditions at the channel inlet and outlet sections, namely $\Gamma_P = \{(x_1, x_2) \in \mathbb{R}^3 | x_1 = 0 \cup x_1 = 1\}$. This strategy is important to avoid boundary conditions affecting the bulk of the flow and is also adopted in Moser et al. (1999); del Álamo and Jiménez (2003); Lee and Moser (2015); Kessels (2016). The top and bottom walls are modelled with a Dirichlet non-slip condition.

Since it is a pure Dirichlet problem, the pressure field is constrained to have a zero mean value. Given the pressure constraint and the fact that the Dirichlet boundaries are only imposing non-slip conditions (i.e., zero-valued fields), it is necessary to add a driven force to the momentum equation to allow the flow to develop from the zero values at the wall. To this end, a unitary source term $\boldsymbol{s} = (1, 0)$ is added to the right-hand side of the RANS momentum equation to mimic the pressure gradient in the x_1 direction. From a momentum balance, it can be shown that this leads to a unitary friction velocity, namely $u_\tau = 1$.

The computational domain is discretised with a set of five structured triangular and quadrilateral grids, with refinement close to the wall, allowing the proper capture of the flow gradients. Fig. 5.7 displays the coarsest triangular and quadrilateral meshes. The i-th quadrilateral mesh has $10 \times (64 \times 2^i)$ cells, whereas the corresponding triangular meshes are obtained by splitting each quadrilateral into four triangles. Given the one-dimensional nature of the flow, no refinement in streamwise direction is performed between the i-th and i+1-th grids. The grid height at the wall for the i-th mesh is $h_0^i = h_0/2^i$. The spacing is selected as $h_0 = 1.6 \times 10^{-3}$ so that the average y^+ at the wall is below 0.2 for all grids.

The problem is solved using the HLL convective stabilization. The Newton-Raphson-based pseudo-time marching strategy described in Appendix B.2 is applied, using a uniform field corresponding to the free-stream values as the initial condition. The monolithic strategy described in Section 2.4.1 is used to solve the RANS-SA coupling. The solution is advanced until the steady state is reached with a tolerance of 10^{-10}. Within each time step,

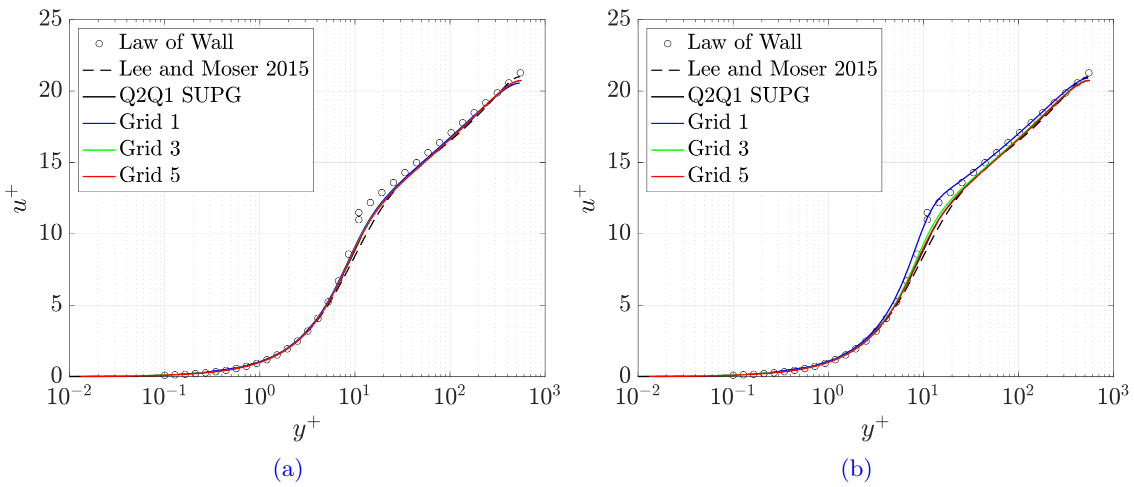


Figure 5.8: Turbulent flow in a channel: Dimensionless value of u^+ as a function of y^+ at $x_1 = 0.5$ for (a) triangular and (b) quadrilateral meshes, compared to the law of wall from Von Kármán (1939), DNS solution from Lee and Moser (2015), and a reference numerical solution from Kessels (2016).

the Newton-Raphson residual tolerance is also set to 10^{-10} while the maximum number of Newton-Raphson solves allowed is set to 20.

The mean velocity profile at the channel mid-cross section is compared with the log of wall of Von Kármán (1939), DNS data from Lee and Moser (2015), available at Lee and Moser (2013), and also against Taylor-Hood (Q2Q1) RANS-SA computations from Kessels (2016), as shown in Fig. 5.8. The results of Kessels (2016) are computed with 2048 cells perpendicular to the flow and 2 cells in the stream-wise direction.

From the results in Fig. 5.8, we can observe that the FCFV solution is almost indistinguishable from the reference numerical solution of Kessels (2016) for all grids shown when triangular cells are considered. For the quadrilateral grid, the first level of refinement struggles to capture the turbulent velocity profile in the transition zone (i.e., buffer layer zone). However, the other two quadrilateral grids align closely with the numerical reference. The better accuracy achieved by the triangular meshes is due to the extra degrees of freedom obtained when splitting each quadrilateral into four triangles. It is important to note that both the numerical FCFV computations and those by Kessels (2016) use an SA modelling approach. Therefore, their results are not expected to fully coincide with the DNS results of Lee and Moser (2015), primarily due to intrinsic modelling errors and also because the computations are 2D and do not account for the 3D flow structures present in the DNS computations. The Sparlat-Allmaras working viscosity, computed with FCFV, and the numerical reference are shown in Fig. 5.9. While the first quadrilateral grid seems to perform better than the triangular counterpart, the second and third refinement levels show slightly less accurate results than those computed with triangular grids.

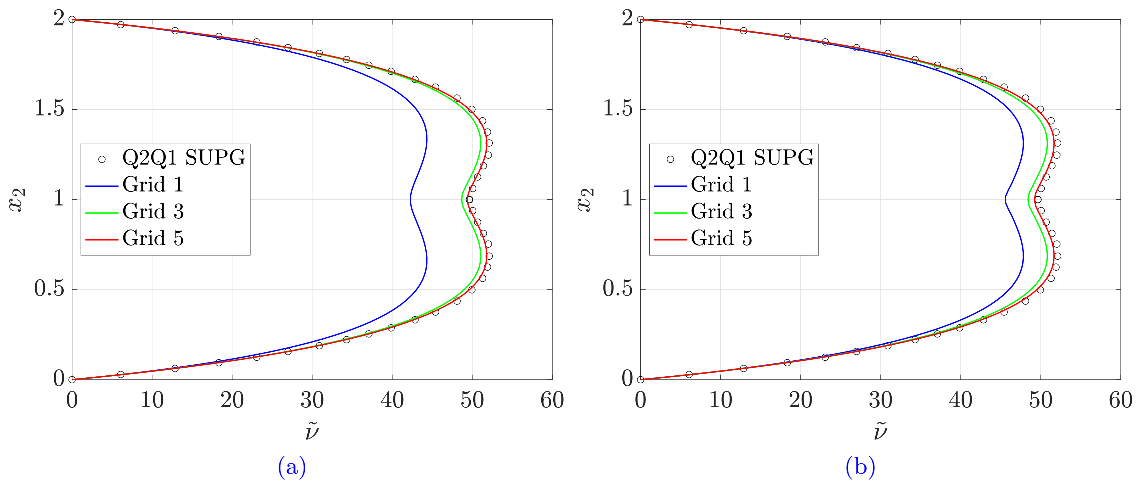


Figure 5.9: Turbulent flow in a channel: Sparlat-Allmaras working variable at $x_1 = 0.5$ for (a) triangular and (b) quadrilateral meshes, compared to a reference numerical solution from Kessels (2016).

5.2.1 Application of the Newton-Raphson-based relaxation strategy

As aforementioned, the results shown in the previous section were computed employing the pseudo-time marching strategy described Appendix B.2, which is based on the total number of Newton-Raphson solves within each pseudo-time step. This strategy was devised to be a more conservative approach when compared to the traditional relaxation strategies like SER, described in Appendix B.1.

The evolution of the steady state residuals and the CFL using the standard parameters, also described in Appendix B.2, are shown in Fig. 5.10 for the first triangular and quadrilateral meshes. We observe that both triangular and quadrilateral grids have a steady increment on the CFL, meaning that the solver did not have issues achieving convergence within each time step. The total number of time steps required for convergence is 21 and 18 for triangular and quadrilateral grids, respectively. We remember that, differently from other relaxation strategies, here the convergence of the Newton solver within each time step is sought. This means that the total computational cost is proportional to the sum of the total number of Newton-Raphson solves performed within each time step.

In Fig. 5.12 (a), the total number of Newton-Raphson solves required on each time step of the computation of Fig. 5.10 is shown. The overall number of solves is 88 and 77 for triangular and quadrilateral grids, respectively. It can be observed that both triangular and quadrilateral meshes present a spike in the number of solves around time step 10, due to the fast increment of CFL. This is reflected in the residuals of the turbulent variable $\tilde{\nu}$, as seen in Fig. 5.10(a), where the residuals tend to increase. From the physical point of view, as

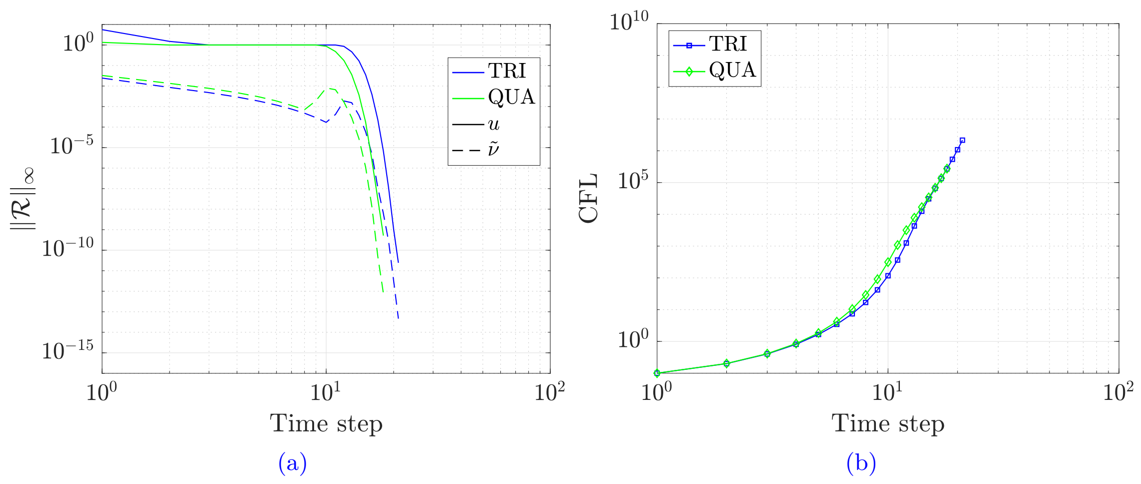


Figure 5.10: Turbulent flow in a channel: Evolution of the (a) residuals and (b) CFL during the pseudo-time marching for the computation using the first triangular and quadrilateral meshes with the baseline parameters of B.2.

CFL evolves, the solution evolves and more turbulence is generated from the uniform initial condition, leading the solver to struggle to minimise the residuals.

Note that in the first time step, in Fig. 5.12(a), the total number of solves is high, at 10 for triangles and 9 for quadrilaterals, due to the incompatible uniform initial condition set. Regarding the Newton-Raphson order of convergence, the nominal quadratic convergence is only achieved when the initial guess is close to the basin of attraction of the solution. Thus, in such relaxation strategies, when the solution changes substantially between time steps, one can expect a loss of convergence order, similarly as observed here on the time steps where the number of Newton-Raphson solves peaks. On the time steps where the number of Newton-Raphson solves is around 4-5, in Fig. 5.12, the measured order of convergence for the RANS and SA equations is around 1.8-1.9.

To further illustrate the CFL increment effect and the robustness of the employed relaxation strategy, we change the value of the parameter $\mathbf{r}_{max}$ from 2.0 to 4.0. This parameter controls the maximum CFL increment between successive time steps. The results of this computation for the first triangular and quadrilateral meshes are shown in Fig. 5.11. As we can see, while increment on $\mathbf{r}_{max}$ reduces the total number of time steps for convergence from 21 to 16 for triangles, for quadrilaterals it remains unchanged at 18. Additionally, in Fig. 5.11(b), we observe that the CFL growth is not smooth anymore, and the relaxation strategy reduces the CFL due to the solver's difficulty converging at the respective time step. This is shown by Fig. 5.12(b), where the number total of Newton-Raphson solvers peaks at the maximum allowed number of Newton-Raphson solves, namely 20, within the time steps 6 and 7 for quadrilaterals and at time step 7 for triangles. In response to it, the CFL adap-

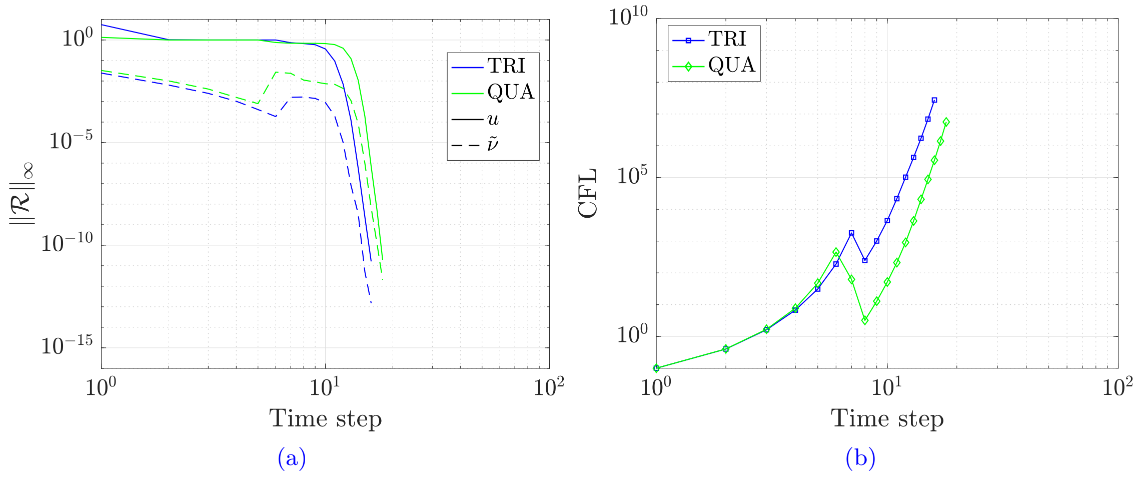
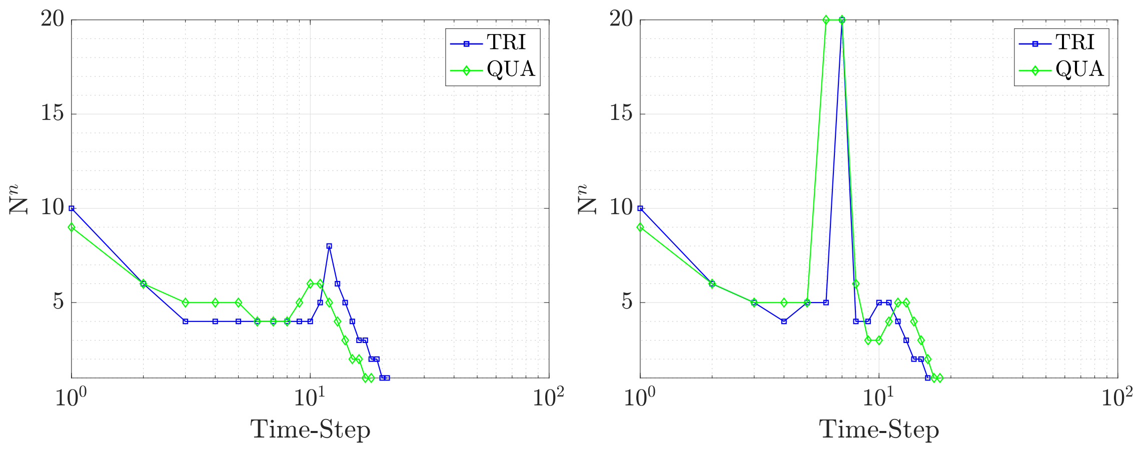


Figure 5.11: Turbulent flow in a channel: Evolution of the (a) residuals of RANS and SA equations and (b) CFL during the pseudo-time marching for the computation using the first triangular and quadrilateral meshes with the parameter $\mathrm{r}_{max} = 4.0$.



(a) Baseline $\mathrm{r}_{max} = 2.0$ (b) $\mathrm{r}_{max} = 4.0$

Figure 5.12: Turbulent flow in a channel: Number of Newton-Raphson solves per time step n, namely $\mathbf{N}^n$, using (a) the baseline $\mathrm{r}_{max} = 2.0$ and (b) $\mathrm{r}_{max} = 4.0$ for the pseudo-time marching scheme computed with the first triangular and quadrilateral meshes.

tive algorithm reduces the CFL recovering convergence of the Newton-Raphson solver within the subsequent time steps. Further, while the total number of solves for the triangular grid reduces from 88 to 85, for the quadrilateral grid the total cost increases from 77 to 107. From this, we can conclude that increasing the CFL increment rate can be beneficial up to a certain point where it starts to preclude the solver's convergence ability, leading to a higher computational cost.

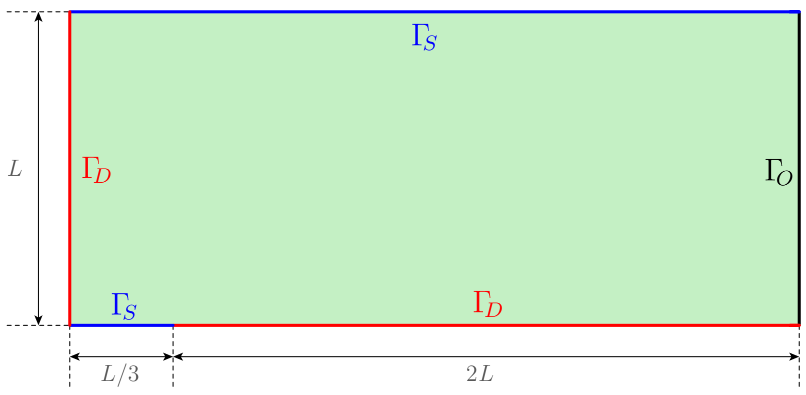


Figure 5.13: Turbulent flow over a flat plate: Problem setup.

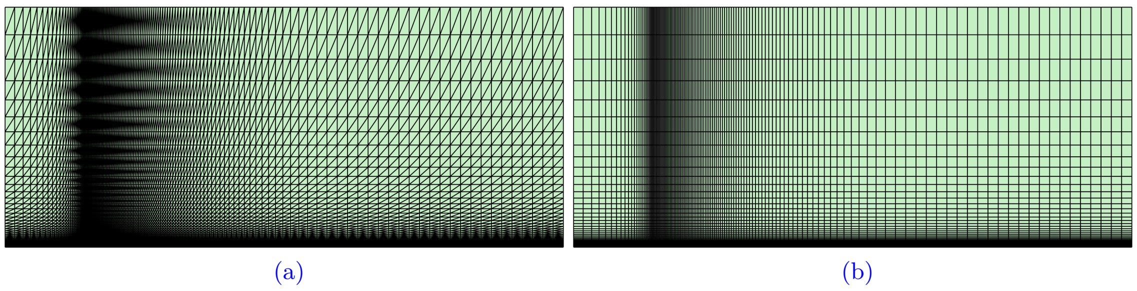
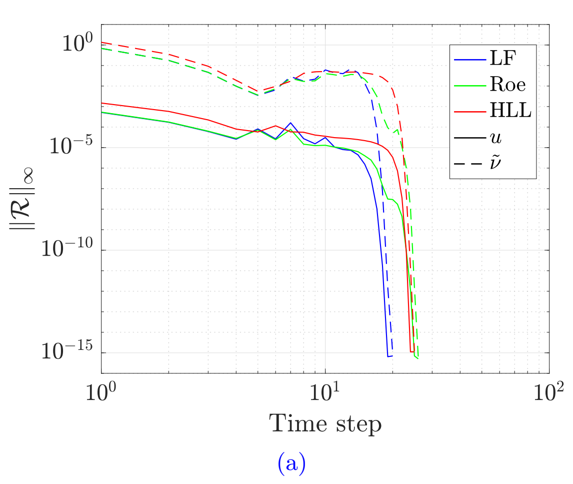


Figure 5.14: Turbulent flow over a flat plate: (a) Triangular and (b) quadrilateral coarse meshes.

5.3 Turbulent flow over a flat plate

The next example considers the steady turbulent flow over a flat plate at $Re = 5 \times 10^6$, a classical benchmark to test the accuracy and robustness of steady turbulent flow solvers. In this context, the FCFV solver will be tested with the different convective stabilisation defined in Appendix A. The robustness concerning grid distortion and a comparison between monolithic and staggered approaches on the CFL limit when employing the SER relaxation strategy from Appendix B.1 will be provided. The problem setup is depicted in figure 5.13, including the relevant dimensions and the boundary conditions. The Dirichlet boundary consists of the inflow boundary, where a constant horizontal velocity of magnitude one is imposed, and the plate, where a no-slip boundary condition is imposed. The inflow SA viscosity is taken as $\tilde{\nu} = 3$, assuming that the flow is fully developed at the inlet, as recommended by Spalart and Allmaras (1992); Allmaras et al. (2012).

Three triangular and quadrilateral meshes, typically employed with NASA solvers CLF3D and FUN3D, as in Rumsey (2014); Jespersen et al. (2016), are used to test the accuracy and robustness of the proposed FCFV. Fig. 5.14 displays the coarsest triangular and quadrilat-



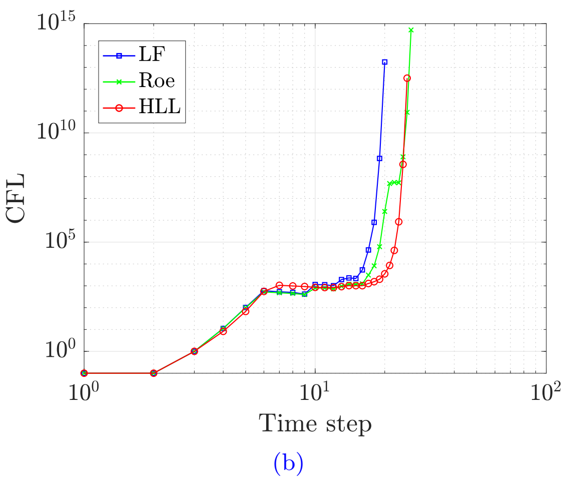


Figure 5.15: Turbulent flow over a flat plate: Evolution of the (a) residuals of RANS and SA equations and (b) CFL during the pseudo-time marching for the computation using the second quadrilateral mesh.

eral meshes. The i-th quadrilateral mesh has $(68 \times 2^i + 1) \times (48 \times 2^i + 1)$ cells, whereas the corresponding triangular meshes are obtained by splitting each quadrilateral into two triangles. The maximum aspect ratio of the cells near the wall in the first mesh is 2.14×10^5 and it is halved for each successive mesh refinement. The spacing is selected so that the average y^+ at the wall is close to 1 and 0.1 for the coarser and finest meshes, respectively.

The SER pseudo-time marching strategy described in Appendix B.1 is applied, using a uniform field corresponding to the free-stream values as the initial condition. As no time accuracy is required, the SER strategy employing only one Newton-Raphson iteration per time step is suitable. The RANS-SA coupling is handled by the monolithic approach described in Section 2.4.1. The solution is advanced until the steady state is reached with a tolerance of 10^{-12}. For details on the residuals used for convergence see B.3.

Figure 5.15 reports the evolution of the residuals of the RANS and SA equations, measured in the maximum norm, and the CFL during the pseudo-time marching for the computation using the coarsest quadrilateral mesh and for the three different convective stabilisation. The results show a very similar performance for the three stabilisations considered. During the first two time steps, a low CFL is required due to the non-physical transient effects caused by the initial condition not satisfying the boundary conditions, but after only six time steps, the CFL is already near 10^3. The CFL stays between 10^3 and 10^5 until the solution develops, and during the last time steps, the CFL exceeds 10^{12}, showing the robustness and stability of the proposed FCFV method. The behaviour is very similar using finer meshes or triangular cells.

To assess the accuracy of the FCFV computations, the skin friction along the plate

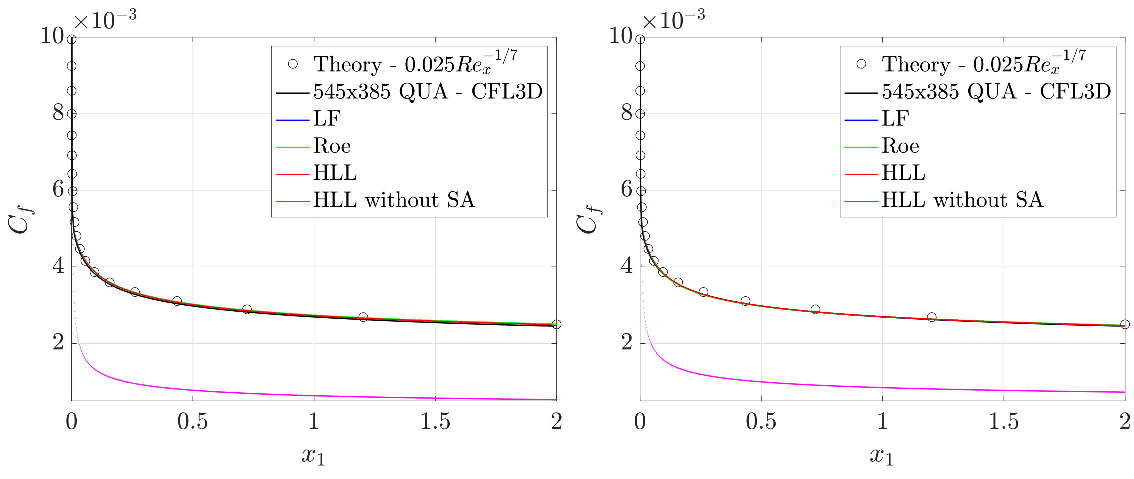


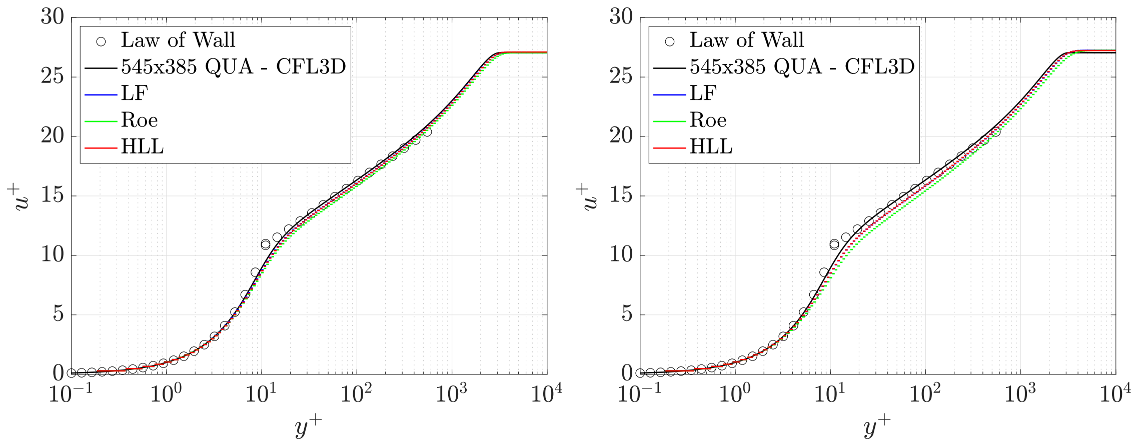
Figure 5.16: Turbulent flow over a flat plate: Skin friction along the plate for (a) the second triangular mesh and (b) the second quadrilateral mesh, compared to the Blasius solution and a reference numerical solution. The solution for the HLL convective stabilisation without the SA model is also shown.

is compared in figure 5.16 to two references, corresponding to the Blasius solution and the numerical solution obtained with the CFL3D solver in Rumsey (2014); Jespersen et al. (2016).

The FCFV computations are performed on the second triangular and quadrilateral meshes, with $273 \times 193 \times 2$ cells and with 273×193 cells, respectively, and employing the three different convective stabilisations. The numerical reference solution is computed on the finest quadrilateral mesh, featuring 545×385 cells, with the CFL3D solver. The FCFV results show excellent agreement with the reference behaviour. For the quadrilateral mesh, the results using different convective stabilisations are almost indistinguishable, whereas on the triangular mesh, a slightly better performance of the HLL stabilisation is observed.

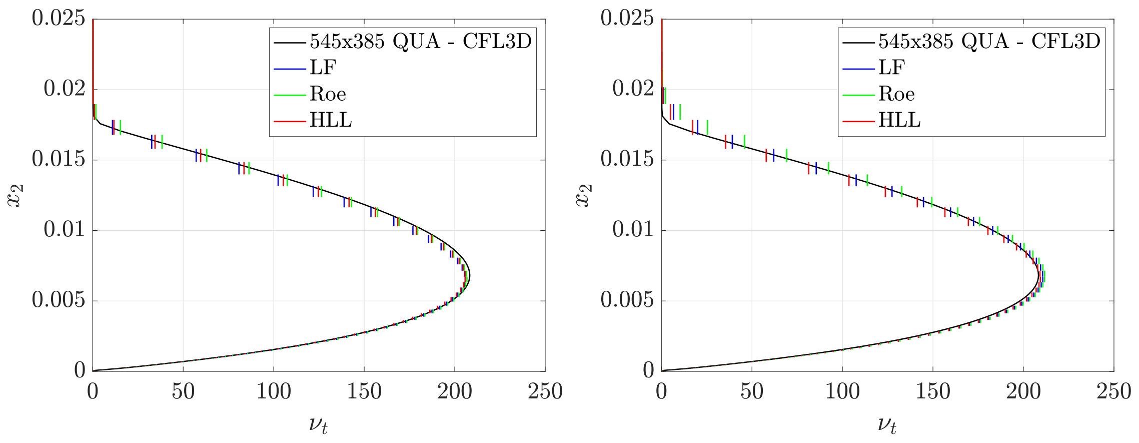
This figure also presents the results obtained using the HLL stabilisation without activating the SA turbulence model. The findings clearly demonstrate that a turbulence model is essential for achieving accurate results, even considering the additional dissipation inherent in the first-order FCFV method compared to higher-order methods. Furthermore, the solution without the activation of the SA model tends to converge towards the Blasius laminar solution. Similar conclusions regarding convective stabilisation are obtained when comparing the dimensionless value of u^+ as a function of y^+ at $x_1 = 0.97008$, as shown in figure 5.17, and the turbulent viscosity at the same location, as shown in figure 5.18.

To further analyse the accuracy of the results, table 5.1 reports the computed drag coefficient using the three available triangular and quadrilateral meshes and the three different convective stabilisations. The results of Rumsey (2014); Jespersen et al. (2016), obtained using the CFL3D and FUN3D solvers, are also included and show that the FCFV results are



(a) Triangles (b) Quadrilaterals

Figure 5.17: Turbulent flow over a flat plate: Dimensionless value of u^+ as a function of y^+ at $x_1 = 0.97008$ for (a) the second triangular mesh and (b) the second quadrilateral mesh, compared to the law of wall and a reference numerical solution.



(a) Triangles (b) Quadrilaterals

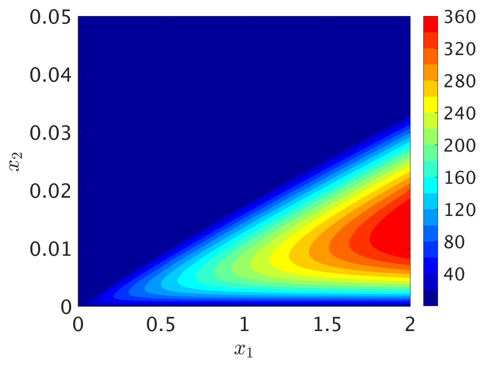
Figure 5.18: Turbulent flow over a flat plate: Turbulent viscosity at $x_1 = 0.97008$ for (a) the second triangular mesh and (b) the second quadrilateral mesh, compared to a reference numerical solution.

less than one drag count away from the reference results. The extra accuracy of the FCFV results using triangular meshes is only due to the extra degrees of freedom introduced when splitting the quadrilaterals into two triangles.

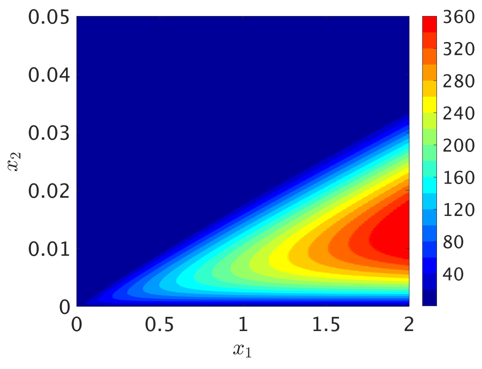
In Fig. 5.19, the turbulent viscosity contour near the flat plate wall, in the range $(x_1, x_2) = \big([0,\ 2] \times [0,\ 0.05]\big)$, is plotted for the 545x385 quadrilateral grid. The result with the FCFV using the HLL convective stabilisation is compared with the FUND3D solution on the same grid, and good agreement between the two solutions is observed.

FCFV results						
	Triangles			Quadrilaterals		
Stabilisation	Mesh 1	Mesh 2	Mesh 3	Mesh 1	Mesh 2	Mesh 3
LF	0.00290	0.00290	0.00290	0.00281	0.00287	0.00289
Roe	0.00290	0.00291	0.00291	0.00280	0.00287	0.00289
HLL	0.00289	0.00290	0.00290	0.00281	0.00287	0.00289
Reference results						
Reference	Triangles			Quadrilaterals		
CFL3D	–	–	–	0.00287	0.00286	0.00286
FUN3D	–	–	–	0.00284	0.00285	0.00285

Table 5.1: Turbulent flow over a flat plate: Drag coefficient computed using triangular and quadrilateral meshes and the three different stabilisations, and reference results.



(a) FUN3D



(b) FCFV HLL

Figure 5.19: Turbulent Viscosity ν_t field for the quadrilateral 545x385 grid: (a) FUN3D and (b) FCFV with HLL convective estabilization.

5.3.1 Robustness on distorted grids

To assess the robustness of the proposed FCFV method concerning mesh quality, a random perturbation to the interior nodes is introduced. The resulting coarse, distorted triangular and quadrilateral meshes are shown in figure 5.20. The skin friction using the HLL convective stabilisation on the second regular and distorted triangular and quadrilateral meshes is compared in figure 5.21. The results show very little influence of the substantial mesh distortion introduced, demonstrating the insensitivity of the FCFV method to mesh quality and the overall robustness of the proposed solver, even for high Reynolds number turbulent flows.

To further illustrate the robustness of the method with respect to cell distortion, figure 5.22 compares the dimensionless value of u^+ as a function of y^+ at $x_1 = 0.97008$ and the turbulent viscosity at the same location when using regular and distorted triangular meshes. In both cases, the HLL convective stabilisation is used, and the results on distorted meshes

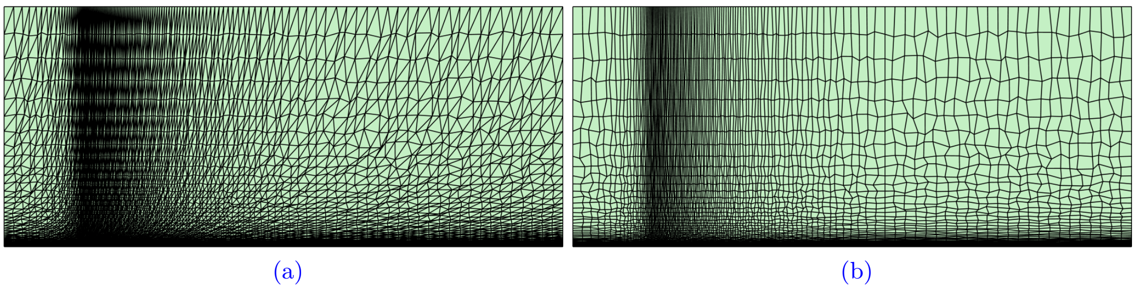


Figure 5.20: Turbulent flow over a flat plate: Distorted (a) triangular and (b) quadrilateral coarse meshes.

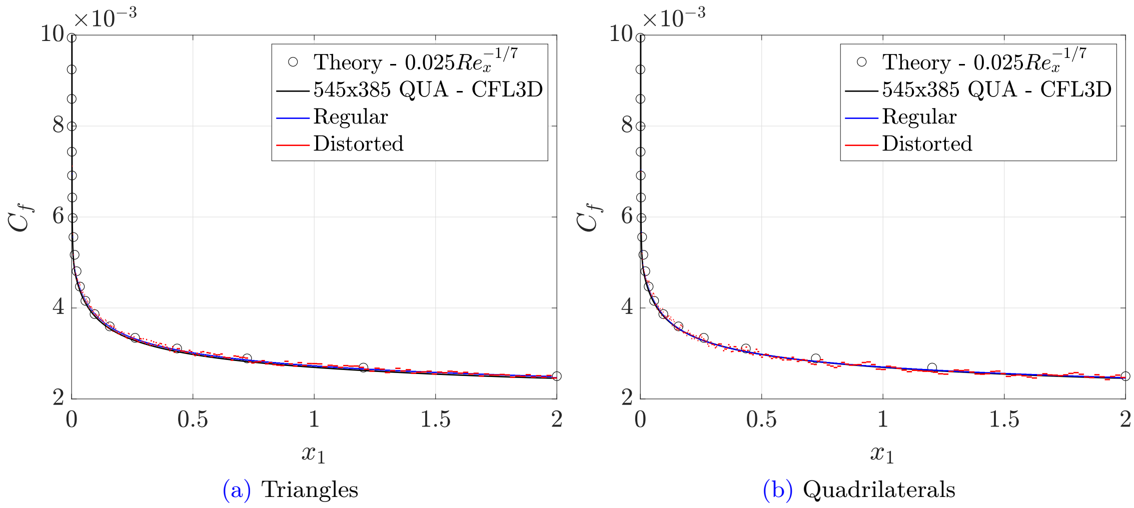
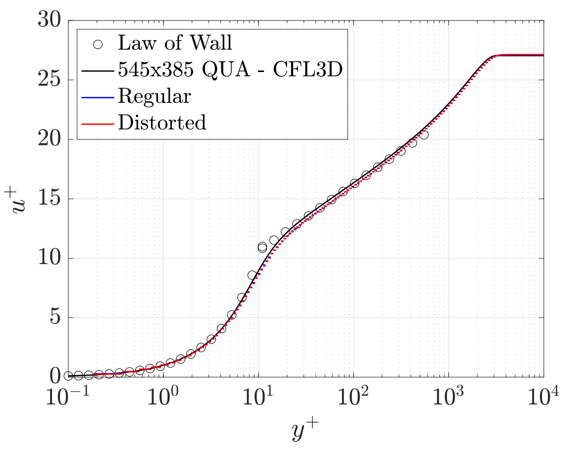
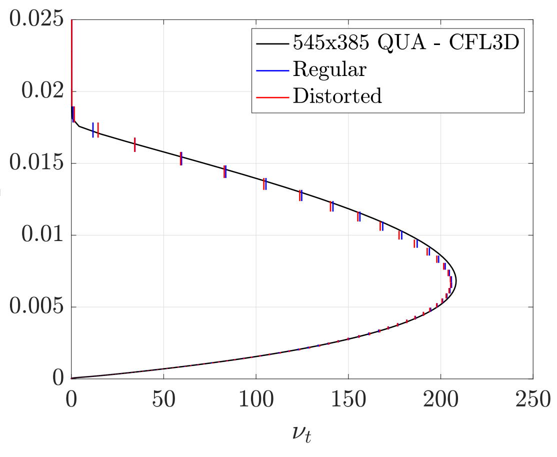


Figure 5.21: Turbulent flow over a flat plate: Skin friction on the second (a) triangular and (b) quadrilateral distorted meshes compared to the computation with regular meshes, the Blasius solution and a reference numerical solution.

are almost identical to the ones obtained with regular meshes. The unnoticeable difference induced by the mesh distortion is corroborated by comparing the drag coefficient computed with distorted meshes. For instance, the drag coefficient obtained using the second distorted triangular mesh is 0.00291, which is less than one drag count away from the result with the second triangular mesh, 0.00290. The same conclusions are obtained for quadrilateral meshes and other levels of mesh refinement.



(a)



(b)

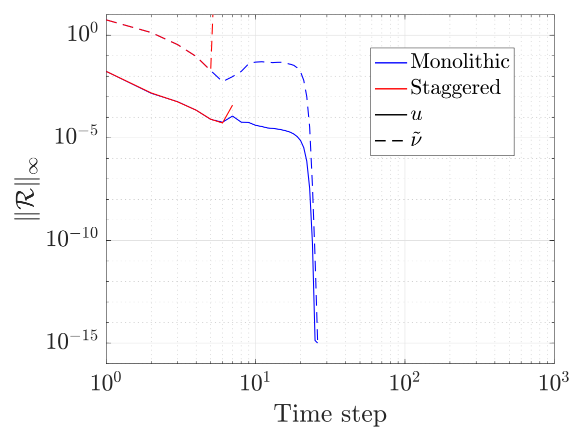
Figure 5.22: Turbulent flow over a flat plate: (a) Dimensionless value of u^+ as a function of y^+ and (b) turbulent viscosity, at $x_1 = 0.97008$, on the second triangular distorted mesh compared to the computation with regular meshes and a reference numerical solution.

5.3.2 CFL evolution limit on monolithic and staggered approaches

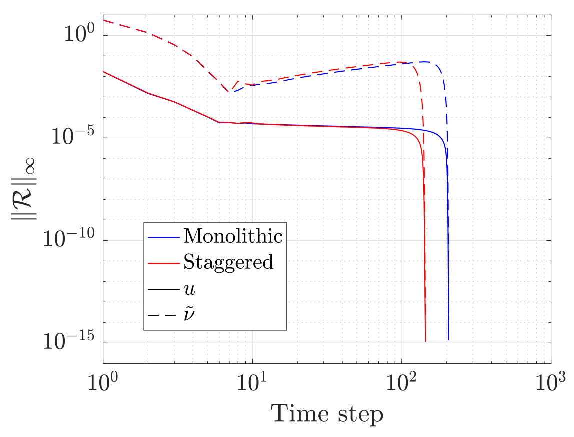
All the above results have been computed using a monolithic strategy to solve the RANS-SA system. A comparison of the performance and robustness of this approach with a staggered algorithm described in 2.4.2 is performed in this section.

When using the staggered FCFV approach, numerical experiments show that a high CFL increment speed can lead to divergence of the non-linear solver. To illustrate this phenomenon, let us solve the turbulent flow over a flat plate with the staggered approach using the SER strategy, described in B.1, with different values for the parameter γ_{max}, the one controlling the CFL increment rate. Figure 5.23 shows the evolution of the residuals of the RANS and SA equations during the pseudo-time marching.

The results correspond to the second quadrilateral mesh and using the HLL stabilisation. When using the value of $\gamma_{\max} = 2$, the monolithic approach converges in less than 30 time steps, whereas the staggered approach diverges after six time steps, as displayed in figure 5.23(a). This illustrates a lack of stability induced by the staggered approach. A simple remedy consists of using a more conservative approach to increase the CFL number, for instance, lowering the value of $\gamma_{\max}$. The results with $\gamma_{\max} = 1.2$ in figure 5.23(b) show that the staggered approach converges. As expected, the more conservative increase of the CFL results in convergence, but with a substantially larger number of time steps.



(a) Baseline $\gamma_{\max} = 2.0$



(b) $\gamma_{\max} = 1.2$

Figure 5.23: Turbulent flow over a flat plate: Evolution of the residuals of RANS and SA equations with (a) $\gamma_{\max} = 2$ and (b) $\gamma_{\max} = 1.2$.

5.4 Turbulent unsteady flow past a cylinder

The next example considers the unsteady turbulent flow past a circular cylinder of diameter D at $Re = 10^4$. This example is used to assess the applicability of the proposed method to turbulent transient cases. The setup of the problem is depicted in figure 5.24, including the relevant dimensions and the boundary conditions. For the RANS equations, the Dirichlet boundary consists of the inflow boundary, where a constant horizontal velocity of magnitude one is imposed, and the cylinder wall, where a no-slip condition is enforced. For the Spalart-Allmaras equations, the Dirichlet condition is an inflow where far-field $\tilde{\nu}$ is set to 0.05, following Nithiarasu and Liu (2006), and the cylinder wall, where $\tilde{\nu}$ is set to zero, as recommended by Spalart and Allmaras (1992, 1994); Allmaras et al. (2012). At the outlet boundary, outflow conditions are imposed, while symmetry conditions are applied at the remaining two boundaries for both RANS and SA equations.

Two unstructured triangular meshes are considered. The grids are designed with a prescribed inflation layer around the cylinder and a wake refinement with an increasing grid size towards the outflow boundary. In Table 5.2, the detail of the unstructured grids used is shown. Here, $\mathtt{n_{el}}$, h_{wake}/D, h_{far}, h_0, r, h_{wall}/h_0, and y^+_{max} denote the grid number of cells, the variable wake grid size, the far-field grid size, the first layer height at the wall, the inflation layer growth ratio, the cell aspect ratio at the wall, and the maximum grid y^+ at the wall, respectively. The grid h_0 is chosen such that the estimated grid y^+ is in the range of 1-2, as recommended. The actual grid y^+_{max}, corresponding to the values shown in Table 5.2, is then obtained by postprocessing the flow field solution. In Fig. 5.25, the second grid with 98,193 cells is shown.

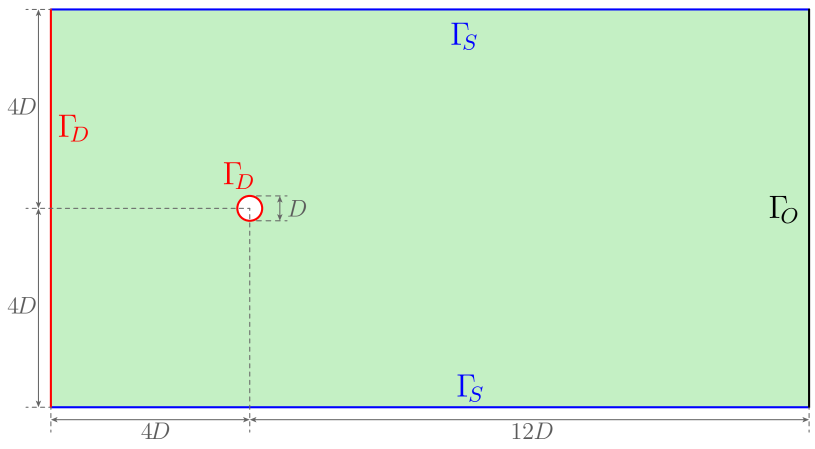


Figure 5.24: Unsteady turbulent flow past a circular cylinder: Problem setup.

Grid	`nel`	h_{wake}	h_{far}	h_0	r	h_{wall}/h_0	y^+_{max}
1	52,094	0.015 - 0.075	0.75	2.0×10^{-3}	1.15	7.5	1.6
2	98,319	0.010 - 0.050	0.75	1.0×10^{-3}	1.10	7.5	0.8

Table 5.2: Unstructed grids for the Turbulent Flow Past Cylinder problem.

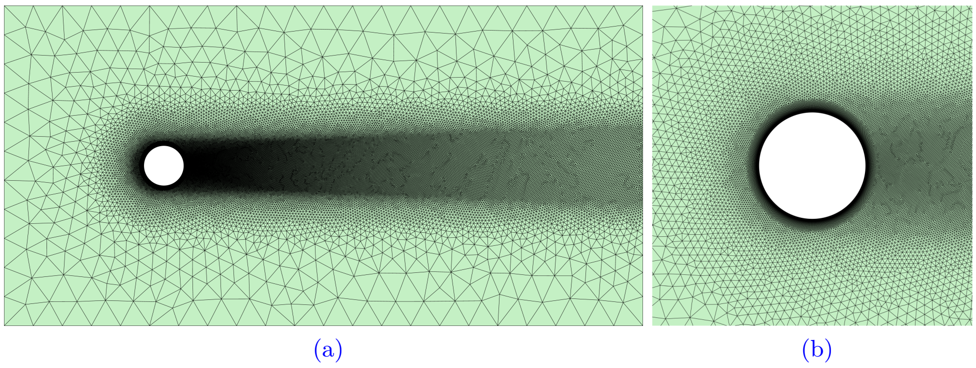
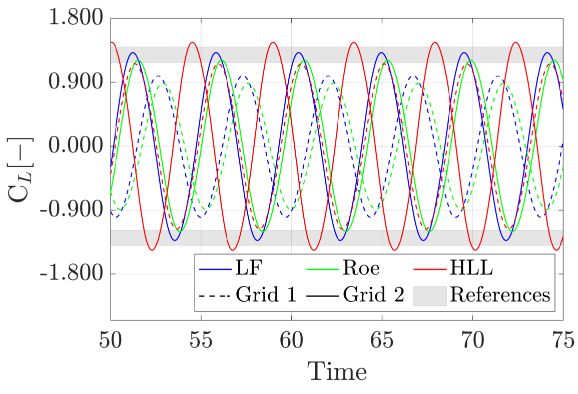
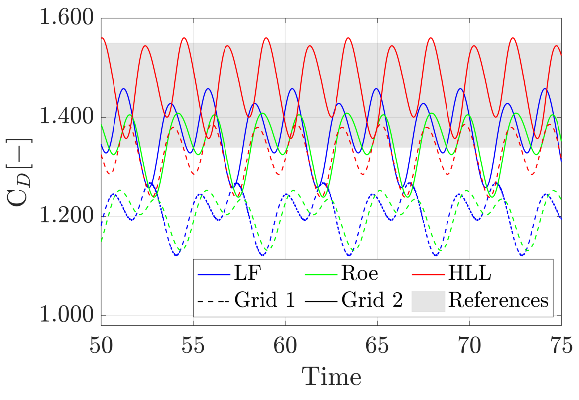


Figure 5.25: Unsteady turbulent flow past a circular cylinder: (a) Second mesh with 98,193 cells and (b) detail of the inflation layer.

For the time integration, the BDF2 scheme is employed with a time-step $\Delta t = 0.01$. The initial condition for the RANS equations is taken as the steady-state solution with $Re = 10$, and for the turbulent variable, a uniform $\tilde{\nu}$ field is set corresponding to the far-field value of 0.05. The monolithic solution strategy, described in Section 2.4.1, is used to handle the RANS-SA coupling. A tolerance of 10^{-6} is used for the nonlinear problems, and the average number of Newton-Raphson iterations across all time steps is three in all cases. This is slightly higher than the number of iterations observed for the laminar flow, illustrating the extra difficulty in solving the coupled RANS-SA system.



(a)



(b)

Figure 5.26: Unsteady turbulent flow past a circular cylinder: (a) Lift and (b) drag coefficients as a function of time for the two meshes and the three convective stabilisations.

FCFV results						
	C_L amplitude		Mean C_D		S_t	
Stabilisation	Mesh 1	Mesh 2	Mesh 1	Mesh 2	Mesh 1	Mesh 2
LF	1.00	1.32	1.21	1.37	0.213	0.219
Roe	0.90	1.20	1.20	1.34	0.213	0.218
HLL	1.18	1.46	1.32	1.46	0.218	0.224

Literature results			
Reference	C_L amplitude	Mean C_D	S_t
Nithiarasu and Liu (2006)	1.40	1.35	0.167
Panneer Selvam (1997)	1.18	1.34	[0.16, 0.19]
Saghafian et al. (2003)	–	1.55	[0.217,0.244]

Table 5.3: Unsteady turbulent flow past a circular cylinder: Amplitude of the lift coefficient, mean value of the drag and Strouhal number using both meshes and the three different stabilisations and reference results.

The accuracy of the FCFV simulations is assessed by comparing the computed lift (C_L) and drag (C_D) coefficients and the Strouhal number (S_t) against values found in the literature. Figure 5.26 shows the lift and drag coefficients as a function of time for the two meshes and the three convective stabilisations, where the shadowed area represents the range of values found in the works of Nithiarasu and Liu (2006); Panneer Selvam (1997); Saghafian et al. (2003).

A more detailed comparison is provided in table 5.3, reporting the amplitude of the lift coefficient, the mean value of the drag, and the Strouhal number using both meshes and the three different stabilisation. Results reported in the literature for the three quantities of interest are also included.

For the references considered, the number of cells varies between 2,000 and 47,000, but it

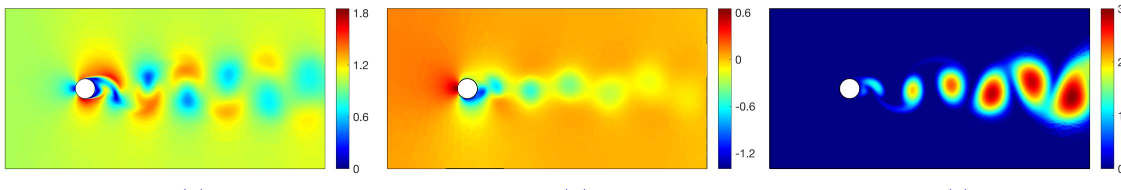


Figure 5.27: Unsteady turbulent flow past a circular cylinder: Snapshots of the (a) magnitude of the velocity, (b) pressure, and (c) turbulent viscosity at $t = 61$.

is important to note that in these references higher-order approximations are considered for the simulations. The results reported in the literature are obtained for a variety of numerical schemes, including finite volumes and stabilised finite elements.

To conclude, the simulation using the first mesh and the HLL flux is repeated using the laminar FCFV solver, without the SA turbulence model. For this case, the C_L amplitude obtained is 1.74, the mean C_D is 1.71, and the Strouhal number is 0.230. These results substantially differ from the results obtained with the SA model, shown in Table 5.3, and are outside of the range of results reported by Nithiarasu and Liu (2006); Panneer Selvam (1997); Saghafian et al. (2003). This shows, again, the need to incorporate the SA turbulence model to accurately simulate the problems considered here, even if the FCFV method proposed is first-order and therefore introduces higher dissipation when compared to other, second-order FV schemes.

A snapshot of the magnitude of the velocity, the pressure, and the turbulent viscosity at time $t = 61$ is presented in figure 5.27. The simulation corresponds to the FCFV solution in the second mesh using the HLL stabilisation. The results show a more complicated pattern of the vortices compared to the laminar flow of section 4.4, and, given the moderate Reynolds number employed, it can be observed that the turbulent effects are confined to the wake.

5.5 3D manufactured source problem

This 3D example involves a manufactured source problem, inspired on Roy et al. (2002, 2007); Lodares et al. (2022), solved in a computational domain such that $\Omega = (0, 1)^3$ and

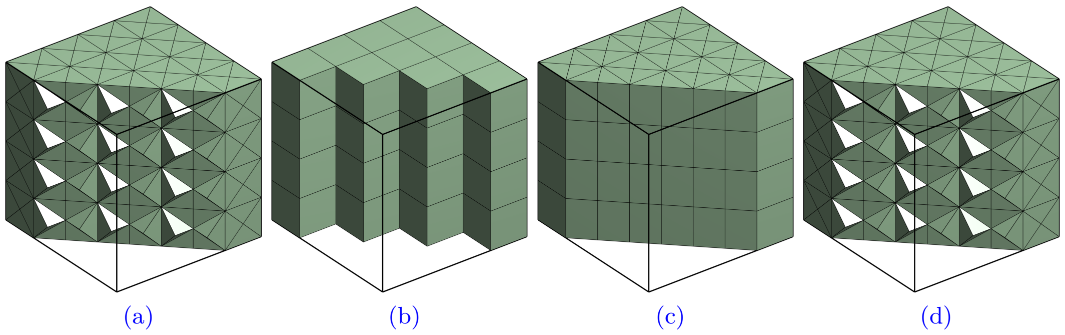


Figure 5.28: 3D manufactured source problem: Cut through of the second grid refinement level (a) Tetrahedrons, (b) Hexahedrons, (c) Prisms, and (d) Pyramids meshes.

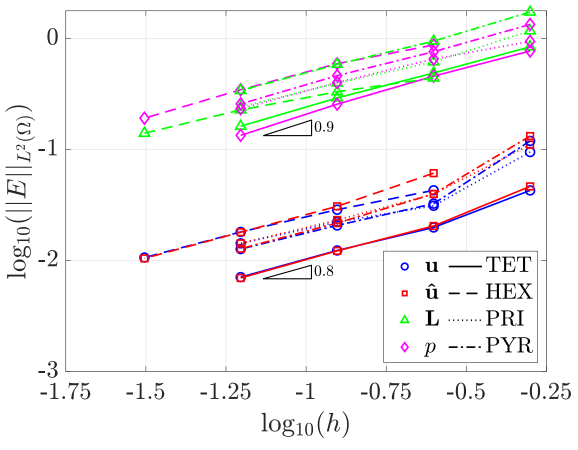
the source term is selected such that the analytical solution is

$$\begin{cases} u_1 = b + (x_3 - y)\sin(x_1 - b), \\ u_2 = a - x_2(x_3 - 0.5x_2)\cos(x_1 - b) - x_2(x_1 - 0.5x_2)\cos(x_3 - b), \\ u_3 = b + (x_1 - x_2)\sin(x_3 - b), \\ p = x_1(1 - x_1) + x_2(1 - x_2) + x_3(1 - x_3) \\ \tilde{\nu} = \tilde{\nu}_{\max}\sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3) \end{cases} \tag{5.3}$$

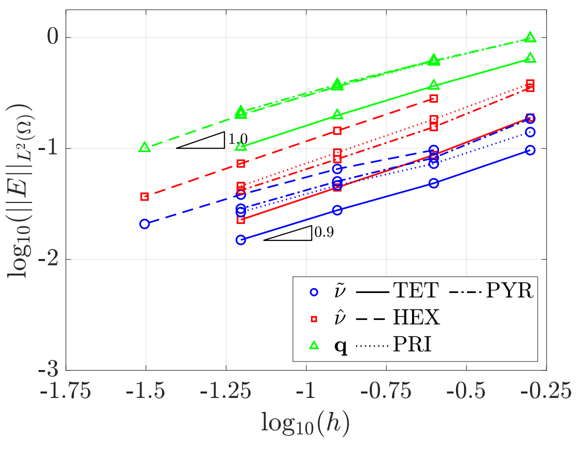
where $a = 1.0$, $b = 0.5$, and $\tilde{\nu}_{\max} = 10$. As it is a synthetic problem with no physical meaning and the exact solution does not depend upon the viscosity, the Reynolds number is selected as $Re = 1$. The analytical traction $\boldsymbol{g}$ is imposed at the Neumann boundary, namely $\Gamma_N = \{(x_1, x_2, x_3) \in \mathbb{R}^3 | x_3 = 0\}$, and analytical Dirichlet conditions $\boldsymbol{u}_D$ and $\tilde{\nu}_D$ are imposed on the remaining boundaries, namely $\Gamma_D = \partial\Omega \setminus \Gamma_N$.

In this example, grid convergence properties of the RANS-SA FCFV formulation will be studied for the four 3D grid element types, namely tetrahedrons (TET), hexahedrons (HEX), prisms (PRI), and pyramids (PYR). Following the remarks made on Section 4.5, this problem is solved using the HLL Rieman solver. The domain is discretised with five regular uniform grids for each element type: tetrahedrons, hexahedrons, prisms, and pyramids. A cut through the second grid refinement level for all cell types considered is shown in Fig. 5.28. The i-th hexahedrons grid has $2^i \times 2^i \times 2^i$ regular cells, while the tetrahedrons, prisms, and pyramids grids are obtained by dividing each cell of the Hexadrons grid into 24, 4, and 6 cells, respectively. The steady-state computation is performed without pseudo-time marching with analytical initial conditions and a tolerance of 10^{-12} on the scaled residuals.

The mesh convergence study for the RANS flow variables, namely cell velocity, face velocity, the gradient of the velocity, and pressure, for the four element types considered, is shown in Fig. 5.29(a). In Fig. 5.29(b), the variables related to the Sparlat-Allamaras equation



(a) RANS



(b) SA

Figure 5.29: 3D manufactured source problem: Mesh convergence of the error of (a) the cell velocity, face velocity, gradient of the velocity, and pressure, and (b) cell turbulent variable, face turbulent variable, and gradient of the turbulent variable for tetrahedrons, hexahedrons, prisms, and pyramids.

discretisation, namely the cell turbulent variable, face turbulent variable, and gradient of the turbulent variable, are shown.

The results reported show the optimal order of convergence for all flow variables and grid types. Additionally, slightly accurate results are observed for tetrahedral grids due to the extra degrees of freedom for a given level of grid refinement when compared to the other grid types.

Chapter 6

The `Fortran 90` face-centred finite volume code

This chapter provides a comprehensive overview of the `Fortran 90` FCFV implementation using a top-down approach. Firstly, Section 6.1 covers the code data structures by examining the derived-type data variables used and their components in detail. Next, Section 6.2 details the modular structure and dependencies within the code. Following this, Section 6.3 outlines the solver flow, building on the previously discussed modular and data structures. This section breaks down each main solver step, including data reading, data preprocessing, time-marching/Newton-Raphson loops, the global solver, the local solver, and post-processing.

6.1 Data structure and derived data types

In this `Fortran 90` FCFV implementation, extensive use of user-defined data types, known as derived data types, Chapman (2018), is made to store key data structures used in the code. This is a very convenient and seamless way to group variables related to the same family, for example, parameter, reference element, mesh, geometry, and so on. For the sake of code readability and efficiency, all derived data type variables are defined once in a centralised manner through a module called `typeDefinitions`. This way all code modules and their functions/routines have access to the derived type definitions a priory, without the need for derived data type creation during runtime.

In the following, we will look inside the `typeDefinitions` module and its defined derived data types.

6.1.1 Mesh

To store the mesh information, the derived data type `meshDef` is used. The `meshDef` has the following components/fields that are filled from the input mesh data:

- `nsd`: Number of spatial dimensions $\mathtt{n_{sd}}$.
- `nOfNodes`: Number of nodes $\mathtt{n_{nd}}$.
- `nOfElements`: Number of elements $\mathtt{n_{el}}$.
- `nOfIntFaces`: Number of internal faces $\mathtt{n_{fa,I}}$.
- `nOfExtFaces`: Number of external faces $\mathtt{n_{fa,E}}$.
- `nOfTotalFaces`: Number total of faces $\mathtt{n_{fa}}$.
- `nOfDirichletFaces`: Number of faces at Dirichlet boundary $\mathtt{n_{fa,D}}$.
- `typeElem`: Array of size `nOfElements` storing the element type flag.
- `X`: Array of size `nOfNodes`×`nsd` storing the nodal coordinates of the mesh.
- `indexT`: Array of size `nOfElements`×2 storing the elemental connectivity for a repeated nodal grid type, as depicted in 6.1(b). The first column contains the first node of the element and the second column contains the last node of the element. For single nodal input grid type, as depicted in 6.1(a), an array of size `nOfElements`×$\mathtt{n_{nd}}$ is used. Where $\mathtt{n_{nd}}$ is the maximum number of nodes in a single cell of the grid. In this case, each column contains the index of each node of the element.
- `intFaces`: Array of size `nOfIntFaces`×4. The first and second columns contain the first element sharing the face and its local face index, respectively. The third and fourth columns contain the second element sharing the face and its local face index, respectively.
- `extFaces`: Array of size `nOfExtFaces`×3. The first and second columns contain the element owning the face and its local face index, respectively. The third column contains the face boundary condition flag, defined in the derived data type `bcDef`.
- `indexTf`: Array of size `nOfTotalFaces`×2 storing the faces/skeleton global degrees of freedom indexes. The first column contains the first global degree of freedom index and the second column contains the last global degree of freedom index.

In Fig. 6.1, an example of a 2D triangular grid is shown, highlighting the indexing of elements, nodes, and faces. The code supports grids with both single nodal indexing, Fig. 6.1(a),

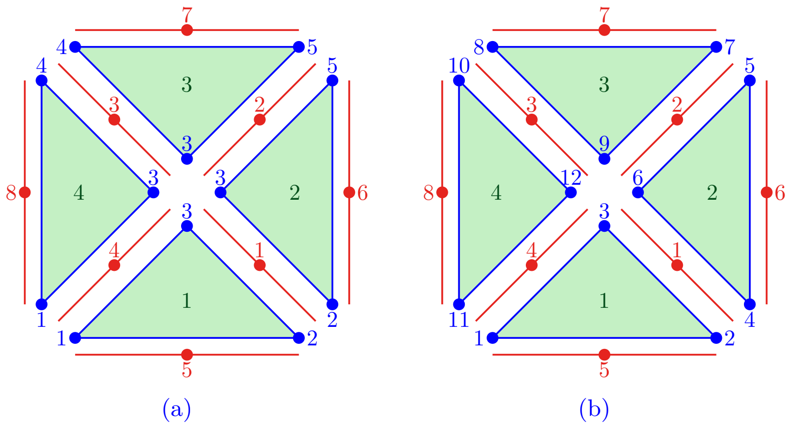


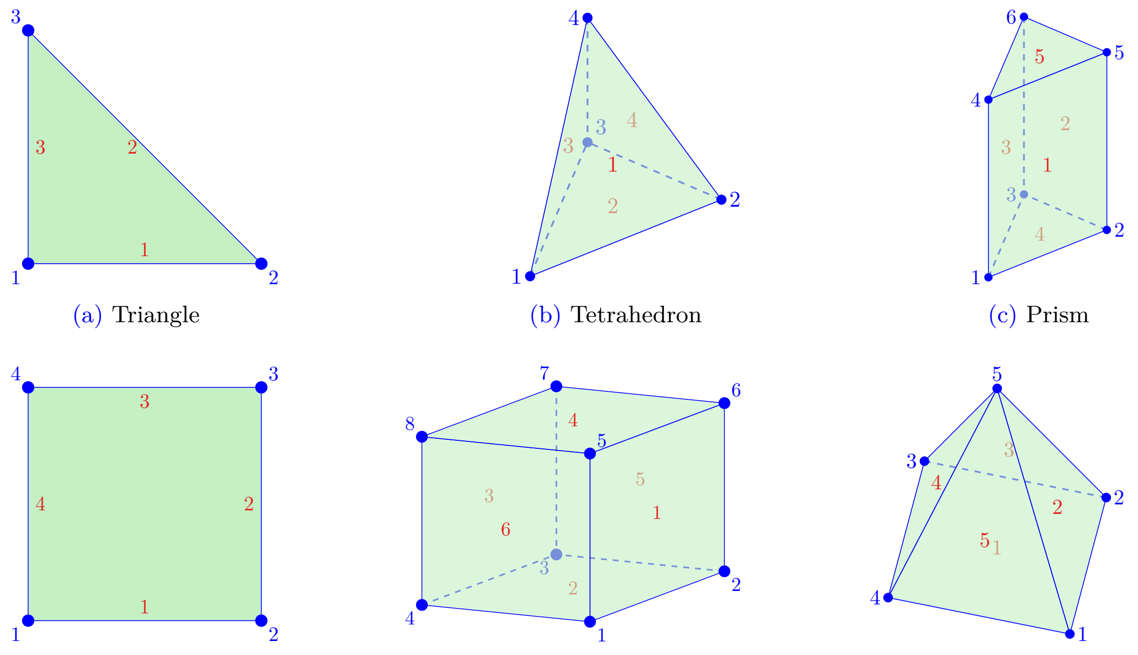
Figure 6.1: 2D example mesh of $\Omega = [0, 1]^2$, composed of four regular triangular elements, showing the global indexing of elements (green), nodes (blue), and face-centroids (red). (a) single nodal indexing, and (b) repeated nodal indexing.

and repeated nodal indexing, Fig. 6.1(b). In the first approach, nodes have the same global index for all cells sharing them, while in the latter, each node is assigned a unique global index for each cell. For grids with repeated nodes, Fig. 6.1(b), sequential indexing is used, allowing storage of only the first and last nodal index of each element, given the total number of nodes per element, $\mathtt{n_{nd}}$. Handling a mesh with repeated nodes simplifies parallelism in future implementations.

6.1.2 Reference element

A reference element data structure is stored to clearly establish the convention about local nodes and face numberings. The `refElemDef` is a derived type array, having a dimension of 2×1 for 2D, storing triangular and quadrilateral reference data, and 4×1 for 3D, storing tetrahedral, hexahedral, prismatic, and pyramidal reference data. This allows the seamless use of hybrid grids in 2D and 3D computations. Each component of the `refElemDef` array, representing one specific element type, has the following fields:

- `nOfFaces`: Number of faces in the reference element $\mathtt{n}_{\mathtt{fa}}^{ref}$.
- `nOfNodes`: Number of nodes in the reference element $\mathtt{n}_{\mathtt{nd}}^{ref}$.
- `faceType`: Array of dimension `nOfFaces`×1 containing the face type flag, with (1) for triangular face and (2) for quadrilateral face, in 3D. For 2D, `faceType` is always (1).
- `nOfFaceNodes`: Array of size `nOfFaces`×1 with the number of nodes for each face.
- `faceNodes`: Array of size `nOfFaces`×`nOfFaceNodes` with each row containing the local index of the nodes of a given face in the reference element. Here `nOfFaceNodes` is



(a) Triangle (b) Tetrahedron (c) Prism

(d) Quadrilateral (e) Hexahedron (f) Pyramid

Figure 6.2: Adopted convection for the reference nodal (blue) and face (red) indexing. 2D (a,d), and 3D (b,c,e,f) reference elements definition.

the maximum between all reference element faces. Faces with fewer nodes receive a dummy value of 0 in the corresponding column.

In Fig. 6.2, the adopted 2D and 3D reference elements are shown, along with their respective nodal and face indexing.

6.1.3 Geometrical temporary data

We adopt an on-the-fly computation for the geometrical quantities used in constructing global and local matrices and vectors to minimise data storage. The derived data type variable responsible for storing temporary geometrical data is `geoDef`. It is a derived type array with dimensions 2×1 for 2D, to temporarily store triangular and quadrilateral data, and 4×1 for 3D, to temporarily store tetrahedral, hexahedral, prismatic, and pyramidal data. Each component of the `geoDef` array contains the following fields:

- `gammaJ`: Array of dimension $\mathtt{n}^e_{\mathtt{fa}}\times 1$ stores each face j length $|\Gamma_{e,j}|$, in 2D, and area, in 3D, of the current element e.

- `nJ`: Array of dimension $\mathtt{n}_{\mathtt{sd}}\times\mathtt{n}^e_{\mathtt{fa}}$ stores each face j normal $\boldsymbol{n}_j$ vector of the current element e.

- Xfm: Array of dimension $\mathtt{n_{sd}}\times\mathtt{n^{e}_{fa}}$ stores each face j barycentre coordinate $\bar{\boldsymbol{x}}_{e,j}$ vector of the current element e.

- Xm: Array of size $\mathtt{n_{sd}}\times 1$ stores the current element e barycentre coordinate $\bar{\boldsymbol{x}}_{e}$.

- Omega: Scalar storing the element area, in 2D, and volume, in 3D, namely $|\Omega_e|$.

6.1.4 FCFV elemental face data

During the preprocessing phase, additional information about the mesh elements faces is extracted. This data is stored in a variable of type faceElemDef. This variable is an array of size $\mathtt{n_{el}}\times 1$, where each entry has a single field storing data of the respective element, as follows:

- faceInfo: Array of size $\mathtt{n^{e}_{fa}}\times 3$. The first column stores the global index of each face of the current element. The second column stores the updated global index of each face of the current element after the elimination of the Dirichlet faces. The third column stores the boundary condition flag of each face of the current element, following the notation introduced in Section 6.3.2.

6.1.5 FCFV solver data and auxiliary variables

For the FCFV formulation solver, it is convenient to introduce parameters and auxiliary variables. The derived data type fcfvDef is used for that. It is composed of the following fields:

- vDOFtoSolve: Array containing the indexes of the global degrees of freedom to be solved for. That is, all degrees of freedom except those associated with the Dirichlet faces.

- pureDirichlet: Pure Dirichlet problem flag. With (1) for pure Dirichlet and (0) for not pure Dirichlet.

- nDofUhat: Number of degrees of freedom associated with the hybrid velocity $\hat{\boldsymbol{u}}$.

- nDofNuHat: Number of degrees of freedom associated with the hybrid viscosity $\hat{\nu}$, for the RANS-SA case.

- nDofGlobal: Number total of degrees of freedom.

6.1.6 Problem parameters and constant valued data

Some of the solver data are constant during runtime, for example, boundary condition flags, problem setup parameters, and model constants. Entailed to store the boundary conditions flag we have `bcDef`, with the following fields

- `iBC_Interior`: Interior face flag.
- `iBC_Dirichlet`: Dirichlet boundary face flag.
- `iBC_Neumann`: Neumann boundary flag.
- `iBC_Outlet`: Outlet boundary face flag.
- `iBC_Symmetry`: Symmetry boundary face flag.

For turbulent RANS-SA cases, an additional derived data type, named `turbConstantDef`, is used to store the turbulence model constants.

- `Sigma, Cb1, Cb2, k, Cw1, Cw2, Cw3, Cv1, Cv2, Cv3, rLim, Ct3, Ct4, Cn1`:

  Sparlat-Allmaras turbulence model constants, defined in Section 2.1, σ, c_{b1}, c_{b2}, κ, c_{w1}, c_{w2}, c_{w3}, c_{v1}, c_{v2}, c_{v3}, r_{lim}, c_{t3}, c_{t4}, and c_{n1}, respectively.

6.1.7 Newton-Raphson data

To manage the data and internal control parameters for the Newton-Raphson iterations the derived data type variable `nrDef` is introduced. This variable has the following components:

- `DeltaUHat, DeltaRho, DeltaNuHat`: Arrays storing the global solution increments at time t^n and Newton-Raphson iteration m, namely $\Delta\hat{\mathbf{u}}^{n,m}$, $\Delta\boldsymbol{\rho}^{n,m}$, and $\Delta\hat{\boldsymbol{\nu}}^{n,m}$, respectively. Those vectors are the components of $\Delta\boldsymbol{\Lambda}^{n,m}$, appearing in Eq. (2.37). The array `DeltaNuHat` is only present in RANS-SA cases.

- `uHat, rho, NuHat`: Arrays storing the global solution at time t^n and Newton-Raphson iteration m, namely $\hat{\mathbf{u}}^{n,m}$, $\boldsymbol{\rho}^{n,m}$, and $\hat{\boldsymbol{\nu}}^{n,m}$, respectively. Those vectors are the components of $\boldsymbol{\Lambda}^{n,m}$, appearing in Eq. (2.38). The array `NuHat` is only present in RANS-SA cases.

- `uHat0, NuHat0`: Arrays storing the hybrid variables of the global solution at time t^{n-1}, namely $\hat{\mathbf{u}}^{n-1}$ and $\hat{\boldsymbol{\nu}}^{n-1}$. Those are stored to allow the evaluation of the numerical stabilization $\boldsymbol{\tau}^{n-1}$, and $\tilde{\tau}^{n-1}$ at time t^{n-1}. The array `NuHat0` is only present in RANS-SA cases.

- `L, u, q, Nu`: Arrays storing the local solution at time t^n and Newton-Raphson iteration m, namely $\mathbf{L}^{n,m}$, $\mathbf{u}^{n,m}$, $\tilde{\mathbf{q}}^{n,m}$, and $\tilde{\boldsymbol{\nu}}^{n,m}$. Those vectors are the components of $\mathbf{U}^{n,m}$, appearing in Eq. (2.38). The arrays `Nu` and `q` are only present in RANS-SA cases.

- `uP, uPP, NuP, NuPP`: Arrays storing the primal solution at time t^{n-1} and at time t^{n-2}, namely $\mathbf{u}^{n-1}$, $\mathbf{u}^{n-2}$, $\tilde{\boldsymbol{\nu}}^{n-1}$ and $\tilde{\boldsymbol{\nu}}^{n-2}$. Those are used to evaluate the numerical time derivative. The arrays `NuP`, and `NuPP` are only present in RANS-SA cases.

- `nOfTotalNrIter, nOfTotalNrIterNS, nOfTotalNrIterSA`: Integer counters storing the total number of Newton-Raphson iterations. Note that, the last two variables are only present in RANS staggered computations where NS and SA equations have independent Newton-Raphson loops (see Section 6.3.7).

- `nOfTotalOutIter`: Integer counter storing the total number of staggered loops. Only present in RANS-SA staggered computations (see Section 6.3.7).

- `Res`: Array of size `(nOfNrIterMax`$\times$`nOfSteps)`$\times 7$ storing scaled infinity norm residuals of the solution at each time t^n and Newton-Rapshon iteration m, namely $\|\mathbf{R}_{\hat{u}}^{n,m}\|_\infty$, $\|\mathbf{R}_{\rho}^{n,m}\|_\infty$, $\|\mathbf{R}_{u,s}^{n,m}\|_\infty$, $\|\mathbf{R}_{u}^{n,m}\|_\infty$, $\|\mathbf{R}_{\hat{\nu}}^{n,m}\|_\infty$, $\|\mathbf{R}_{\tilde{\nu},s}^{n,m}\|_\infty$, $\|\mathbf{R}_{\tilde{\nu}}^{n,m}\|_\infty$, with proper scaling defined in equations (B.6) and (B.8). The variables `nOfSteps`, and `nOfNrIterMax` are components of a `inputDataDef` variable, discussed in Section 6.1.11.

6.1.8 Time-marching scheme data

The derived data type variable `timeMarchingDef` is introduced to store the time-marching data and internal control parameters, and its components are organized as follows:

- `t,dT,iStep`: Variables storing the total time t, the current time-step Δt and the time step idex n.

- `intCoef`: Array of size 3x1 containing the time-scheme numerical derivatives coefficients a_2, present in Eq. (2.32).

- `CFL`: Array of size $\mathtt{n_{el}}\times 1$ storing the the time t^n Courant–Friedrichs–Lewy(CFL) number on each cell.

- `maxCFL, avgCFL`: Variables storing the time t^n maximum and average CFL number over the whole domain respectively.

- `ResTime`: Array of size `nOfSteps`$\times 3$ storing in the first two columns the scaled infinity norm of steady-state residuals, $\|\mathbf{R}_{u,s}^{n}\|_\infty$, $\|\mathbf{R}_{\tilde{\nu},s}^{n}\|$ of (B.8), at last Newton-Rapshon iteration of each time t^n. This is used to measure steady-state convergence when pseudo-time stepping is used. The last column stores the number of Newton-Raphson iterations required for convergence at each time t^n. The variable `nOfSteps` is a component of a `inputDataDef` variable, see Section 6.1.11.

- `FD`: Vector of size `nOfSteps`$\times$$\mathtt{n_{sd}}$ to store the drag force at each time step.
- `E`: Vector of size `nOfSteps`$\times 1$ to store the energy at each time step.
- `vDT`: Vector of size `nOfSteps`$\times 3$ to store the maximum CFL, minimum CFL and Δt on each time step.

6.1.9 Global and local temporary variables

Temporary variables store global auxiliary arrays and local arrays/matrices during runtime. As mentioned in the Section 6.1.3, on-the-fly computation of vectors and matrices is employed to avoid memory overhead.

The `tmpGlobalDef` derived data type is used for the global auxiliary arrays storage. While for local arrays and matrices storage, `tmpLocalDef` is used. Those are derived type arrays, with a dimension of 2×1 for 2D to temporarily store triangular and quadrilateral data and 4×1 for 3D to temporarily store tetrahedral, hexahedral, prismatic, and pyramidal data. Each component of the `tmpGlobalDef` array has the following fields:

- `uHatE, NuHatE`: Auxiliary array variables of size $\mathtt{n^e_{fa}}\times\mathtt{n_{sd}}$ and $\mathtt{n^e_{fa}}$, to store the hybrid velocity $\hat{\mathbf{u}}^n$ and hybrid SA viscosity $\hat{\boldsymbol{\nu}}^n$ at all faces of the current element.
- `uHatE0, NuHatE0`: Auxiliary array variables of size $\mathtt{n^e_{fa}}\times\mathtt{n_{sd}}$ and $\mathtt{n^e_{fa}}$, to store the hybrid velocity $\hat{\mathbf{u}}^{n-1}$ and hybrid SA viscosity $\hat{\boldsymbol{\nu}}^{n-1}$ at all faces of the current element.
- `idGlobalFace, idGlobalFaceRd`: Auxiliary array variables of size $\mathtt{n^e_{fa}}\times\mathtt{n_{sd}}$ to store the global indexes of all faces if the current element. The first stores the indexes before Dirichlet rows and columns elimination while the second stores the updated indexes after Dirichlet rows and columns elimination
- `idGlobalAssembly, idGlobalAssemblyRd`: Auxiliary array variables of size $\mathtt{n^e_{fa}}\times\mathtt{n_{sd}}$ to store the global degree of freedom indexes related to all faces of the current element. The first stores the indexes before Dirichlet rows and columns elimination while the second stores the updated indexes after Dirichlet rows and columns elimination.

Each component of the `tmpLocalDef` array will preallocate, for each element type, space for the storage of the local matrices/vectors related to the local residuals of equations (2.33) and Jacobian blocks, introduced in (2.38), arising from the local residuals linearisation. Each component of `tmpLocalDef` has the following fields related to the Navier-Stokes equations:

- `Tuu,TuuInv`: Matrices of size $\mathtt{n_{sd}}\times\mathtt{n_{sd}}$ storing the $\mathbf{T}^n_{uu}$ and $(\mathbf{T}^n_{uu})^{-1}$ blocks for the current element respectively.

- `Tuhu`: Matrix of size $\mathtt{n_{sd}}\times(\mathtt{n_{sd}}\mathtt{n^e_{fa}})$ storing $\mathbf{T}^n_{u\hat{u}}$ block for the current element.

- `TuL,Tuq`: Vector of size $\mathtt{n_{sd}}\times 1$ and matrix of size $\mathtt{n_{sd}}\times\mathtt{n_{sd}}$, storing $\mathbf{T}^n_{uL}$ and $\mathbf{T}^n_{u\tilde{q}}$ blocks for the current element respectively.

- `Tun,Ru,Rus,Rut`: Vectors of size $\mathtt{n_{sd}}\times 1$ storing $\mathbf{T}^n_{u\tilde{\nu}}$, $\mathbf{R}^n_{e,u}$, $\mathbf{R}^n_{e,u,s}$ for the current element. The subindex s stands for the steady state part of the residuals (see Section B.3). The `Rut` is an auxiliary vector residual storing the time derivative contributions for the current element.

- `TLhu`: Matrix of size $\mathtt{n^2_{sd}}\times\mathtt{n^e_{fa}}$ storing $\mathbf{T}^n_{L\hat{u}}$ block for the current element.

- `RL`: Vector of size $\mathtt{n^2_{sd}}\times 1$ storing $\mathbf{R}^n_L$ for the current element.

- `gI`: Matrix of size $\mathtt{n_{sd}}\times\mathtt{n^e_{fa}}$ storing the traction $\mathbf{g}_i$ imposed at each face of the current element

- `fRho`: Scalar storing the Dirichlet part of the global compatibility condition for the current element, seen in the second equation of (2.34).

The following components are related to the Spalart-Allmaras equation:

- `Tnunu, fnSA, Rn, Rns, Rnt`: Scalars storing $\mathbf{T}^n_{\tilde{\nu}\tilde{\nu}}$, s^n_e, $\mathrm{R}^n_{e,\tilde{\nu}}$, and $\mathrm{R}^n_{e,\tilde{\nu},s}$ respectively. The `Rnt` is an auxiliary residual storing the time derivative contributions for the current element.

- `Tnhu`: Vector of size $(\mathtt{n^e_{fa}}\times\mathtt{n_{sd}})\times 1$ storing $\mathbf{T}_{\tilde{\nu}\hat{u}}$ block for the current element.

- `Tnhn`: Vector of size $\mathtt{n^e_{fa}}\times 1$ storing $\mathbf{T}^n_{\tilde{\nu}\hat{\nu}}$ block for the current element.

- `TnL, Tnq`: Vectors of size $\mathtt{n^2_{sd}}\times 1$ and $\mathtt{n_{sd}}\times 1$, storing the blocks $\mathbf{T}^n_{\tilde{\nu}L}$ and $\mathbf{T}^n_{\tilde{\nu}\tilde{q}}$ for the current element respectively.

- `Tqhn`: Matrix of size $\mathtt{n_{sd}}\times\mathtt{n^e_{fa}}$ to store the $\mathbf{T}^n_{\tilde{q}\hat{\nu}}$ block for the current element.

- `Rq`: Vector of size $\mathtt{n_{sd}}\times 1$ storing R^n_q for the current element.

6.1.10 Postprocess data

The data arising from the solution post-process is stored in `errDef`, and `qoiDef`. The first is intended to store the solution error having the fields:

- `L2,Linfty`: Scalar storing the errors in $\mathcal{L}^2(\Omega)$,or $\mathcal{L}^2(\Gamma)$, and the $\mathcal{L}^\infty(\Omega)$, or $\mathcal{L}^\infty(\Gamma)$, for the concerned flow variable lying on the domain Ω or skeleton Γ.
- `L2elem`: Array of size $\texttt{n}_{\texttt{el}} \times 1$ storing the error in $\mathcal{L}^2$ norm on each element, namely $\mathcal{L}^2(\Omega_e)$, for the concerned flow variable lying on the domain Ω.
- `L2face`: Array of size $\texttt{n}_{\texttt{fa}} \times 1$ storing the error in $\mathcal{L}^2$ norm on the face, namely $\mathcal{L}^2(\Gamma_i)$, for the concerned flow variable lying on the skeleton Γ.

The second stores the computed quantity of interest in the following fields:

- `FD`: Array of size $\texttt{n}_{\texttt{sd}} \times 1$, for steady problems and size $\texttt{n}_{\texttt{sd}} \times$ `nOfsteps`, for unsteady problems. It stores the drag force in all cartesian directions on the specified boundary.
- `E`: Scalar storing the velocity field kinetic energy.

6.1.11 Input data and solver options

This derived type `inputDataDef` stores and distributes the data read from the user input. The components and the associated purpose are listed below: The user input problem setup data is stored in the respective components:

- `ex`: Example problem flag.
- `Rey`: Problem Reynolds number Re.
- `Lref`: Reference length L_{ref} for nondimesionalisation.
- `Vref`: Reference velocity U_{ref} for nondimesionalisation.
- `tripTermFt2`: Sparlat-Allmaras trip term f_{t2} flag, (1) activated and (0) deactivated.

The FCFV-related parameters, arising from the user input, are stored in the respective components:

- `tauType`: Numerical convective stabilization type flag, with (1) for LF, (2) for Roe, and (3) for HLL (see Appendix A).
- `beta`: Array of size 2x1 in 2D or 4x1 in 3D storing the numerical parameter β for each possible element type. This parameter composes the diffusive stabilization parameters $\boldsymbol{\tau}^d$ and $\tilde{\tau}^d$, of Eq. (2.22), from the NS and SA equations respectively.
- `epsilon, epsilonSA`: Array of size 2x1 in 2D or 4x1 in 3D storing the numerical parameter ϵ and $\tilde{\epsilon}$ for each possible element type. This parameter composes the convective stabilization parameters $\boldsymbol{\tau}^a$ and $\tilde{\tau}^a$, from the NS and SA equations respectively, defined in Appendix A.

The user input grid data is stored in the following components:

- `Folder`: String storing the folder name where the grid is located.
- `mesh`: String storing the name of the grid file.
- `fileExtension`: String storing the extension of the file name, it can be `.txt` or `.gmsh`.

The Newton-Raphson user input control parameters are stored at:

- `nOfNrIterMax`: Integer parameter setting the maximum number of Newton-Raphson iterations.
- `nOfStgIterMax`: Integer parameter setting the maximum number of staggered iterations. Only present in RANS-SA staggered computations.
- `nrTol`: Real parameter setting the Newton-Raphson relative tolerance.
- `stgTol`: Real parameter setting the staggered loop relative tolerance. Only present in RANS-SA staggered computations.
- `startType`: Initial condition type flag, with (0) for uniform flow conditions, (1) for input file initial condition, or (2) for analytical initial condition.
- `startFileGlobal`: String storing the global input file path when `startType` $= 1$.
- `startFileLocal`: String storing the local input file path when `startType` $= 1$.

The time-marching control user input parameters are stored as:

- `transient`: Computation regime flag, with (1) for transient/pseudo-transient and (2) for steady-state.
- `continuation`: Continuation of a transient computation from time $t \neq 0$ flag, with (1) for continuation and (0) to start from time $t = 0$. This is used to continue a transient computation at a given time t with an input file solution, given by `startFileGlobal` and `startFileLocal`.
- `scheme`: Time-scheme flag, with (1) for BE and (2) for BDF2.
- `nOfSteps`: Integer containing the total number of time-steps to be performed.
- `dT`: Real parameter setting the time step Δt. Only used when `dTtype = 1`.
- `dTtype`: Time-stepping type flag, with (1) for constant Δt and (2) for adapted Δt. When option (2) is set, `relaxStrategy` $= 1$ will be used by default.

- `timeTol`: Parameter setting the pseudo-time tolerance. Used when `dTtype` $= 2$.
- `iStep`: Integer parameter for the initial step n. Nonzero when `continuation = 1`.
- `t`: Real parameter storing the initial time. Nonzero when `continuation = 1`.
- `relaxStrategy`: Relaxation strategy flag, with (1) for SER, of Appendix B.1, and (2) for Newton-Raphson based, of Appendix B.2.
- `CFLmax, CFLmin`: Maximum and minimum allowed CFL number, namely $\mathrm{CFL}_{\max}$ and $\mathrm{CFL}_{\min}$, respectively.
- `gMax, gMin`: SER relaxation strategy parameters $\gamma_{\max}$, and $\gamma_{\min}$.
- `Nlow, Nhigh, rMax, rMin`: Newton-Raphson based relaxation strategy parameters, $\mathrm{N}_{\mathrm{low}}$, $\mathrm{N}_{\mathrm{high}}$, $\mathrm{r}_{\max}$, and $\mathrm{r}_{\min}$.
- `nStepsSnap`: Integer storing the saving of snapshot frequency.

Finally, the data output parameters are stored as:

- `computeQoI`: Quantity of interest computation flag, with (0) for no computation, (1) for computation of Drag, (2) for computation of energy, and (3) for computation of Drag and Energy.
- `computeErr`: Computation of error flag, with (0) for no computation and (1) for against analytical solution.
- `saveFolder`: String Array storing the path of the folder where files will be saved.

6.2 Solver modular structure

After describing the data structure used in the `Fortran 90` FCFV implementation, we proceed to outline how the solver routines are organised using modules. This section provides a comprehensive overview of the modules and routines within the code, preparing us for the explanation of the solver flow in Section 6.3.

6.2.1 Type definitions, reference element and preprocessing modules

The module `typeDefinitions` is responsible for the definition of all derived data types described in Section 6.1. It also sets the constant values for the variables of type `bcDef` and `turbConstantsDef` using the routines `setBCconstants` and `setTurbConstants` defined within this module.

The module `refElement` is responsible for initializing reference element data, with a single routine named `fcfvRefElem`. For data preprocessing, the module `preProcess` is used, which contains a single routine called `fcfvPreProcess`.

6.2.2 Input and output modules

Routines for input and output of data are present in the modules `input` and `output` respectively. The routines found in the `input` module are listed as follows,

- `readInputData`: Reads the user input data from a `.inp` file.
- `readMesh`: Reads the input mesh from a `.txt` or `.msh` file.
- `readInitialGuess`: Reads the solver initial guess global and local solution files with full path present in a `startFile`, field from a `inputDataDef` variable type, as described in Section 6.1.11.

The full mesh path is defined as `'Folder\mesh.fileExtension'`, from a `inputDataDef` variable type, as described in Section 6.1.11. When calling either `readMesh`, a variable of type `meshDef`, described in Section 6.1.1, is initialized and filled with the input values. The routines making up the `output` module are

- `saveSolution`: Saves the local and global solution on two separate `.dat` files in the `saveFolder`, field from a `inputDataDef` variable type. If `computeQoI` and/or `computeErr`, fields from a `inputDataDef` variable type, are flagged as (1), additional output files with QoI data and/or error data are generated.
- `exportToVTK`: Exports the local solution to a `.vtk` Paraview native format file in the `saveFolder` for external postprocessing.

6.2.3 Computation modules

Here we categorise as a computation module all modules containing routines for geometrical computations on cells and faces, algebraic computations, non-linear function evaluation or linearisation, and numerical stabilisation computation. For geometrical computation on elements, the `elements` module is present, having the following routines:

- `computeVolume`: Computes the area, in 2D, or the volume, in 3D of, a element. It has as inputs the current element nodes, $\mathtt{n_{sd}}$, and `typeElem`, from the variable of `meshDef` type of Section 6.1.1.
- `computeElementalGeometry`: Computes all geometrical data of the element found in a variable of type `geoDef` of Section 6.1.3.

For geometrical computation on faces, the `faces` module is present, having the following routines:

- `computeFaceAreaNormal`: Compute, for a given element e and face j, the length/area $|\Gamma_{e,j}|$ and normal $\boldsymbol{n}_j$.
- `computeTangFace`: Compute the $k = 1 \ldots \texttt{n}_{\texttt{sd}} - 1$ tangent vectors $\boldsymbol{t}^k$ of a given face j.
- `getFaceXf`: Recover the nodes $\boldsymbol{x}_{e,j}$ of a given face j. The output is a vector of size `nOfaceNodes`$\times\texttt{n}_{\texttt{sd}}$. The `nOfaceNodes` is obtained from a variable of type `refElemDef` of Section 6.1.2.

Algebraic computations with matrices and vectors are defined in the `algebraicOperations` module, composed of the following routines.

- `meaX`: Compute the mean value/barycenter of a set of points.
- `matInv`: Compute the analytical inverse of a matrix of size 2×2 or 3×3.
- `cross3D`: Compute the cross product of two vectors of size 3×1.

The module `functionsSAlinearisation` is used to evaluate the SA non-linear functions and their derivatives. The module has the following routines:

- `computeSourceSA`: Compute the discrete SA source term s of Eq. (2.12).
- `computefw, computeSt, computeS, computeft2`: Compute the function f_w, vorticity S, modified vorticity $\tilde{S}$, and function f_{t2}, defined in Eq. (2.4).
- `computeDft2, computeDfw, computeDSt, computeDfwDSt, computeDStDS`: Compute the derivatives $\partial f_{t2}/\partial \tilde{\nu}$, $\partial f_w/\partial \tilde{\nu}$, $\partial \tilde{S}/\partial \tilde{\nu}$, $\partial f_w/\partial \tilde{S}$, and $\partial \tilde{S}/\partial S$. The derivatives arise from local jacobian blocks computation of Eq. (2.38), described in Appendix C.1.1.
- `computeElementD`: Compute the element barycenter $\bar{\boldsymbol{x}}_e$ distance to the wall d.

In module `stabilization` the evaluation of the numerical stabilization parameters is done by means of the following routines:

- `computeTauFace, computeTauFaceGlobal`: Compute the NS stabilization parameter $\boldsymbol{\tau}$, seen from the local face j and global face i, respectively. The i and j contributions appears in equations (2.34) and (2.33) respectively.
- `computeTauFaceSA, computeTauFaceSAGlobal`: Compute the SA stabilization parameter $\tilde{\tau}$, , seen from the local face j and global face i, respectively. The i and j contributions appears in equations (2.34) and (2.33) respectively

6.2.4 Analytical solution modules

These modules are used for the computation of the analytical flow field values, used in problems involving manufactured solutions. For the evaluation of the analytical solution of the RANS/NS equations, the module `analyticalNS` is used. The routines have as inputs a given coordinate (x, y) and an example flag `ex`, present in a variable of type `inputDataDef`. The routines are the following:

- `anlVelocity`: Compute the analytical velocity $\boldsymbol{u}$.
- `anlL`: Compute the analytical NS mixed variable $\boldsymbol{L}$.
- `anlPressure`: Compute the analytical pressure p.
- `anlManSource`: Compute the analytical NS manufactured source $\boldsymbol{s}$.
- `anlNeumann`: Compute the analytical NS Neumann traction $\boldsymbol{g}$.

The module `analyticalSA` is used for the evaluation of the analytical solution of the SA equations, having the following routines:

- `anlSAnu`: Compute the analytical turbulence variable $\tilde{\nu}$.
- `anlSAq`: Compute the analytical SA mixed variable $\boldsymbol{q}$.
- `anlSAManSource`: Compute the analytical SA manufactured source s_m.
- `anlSAneumann`: Compute the analytical SA Neumann traction g.

6.2.5 Newton-Raphson module

The routines to control the Newton-Raphson iterations are organised inside the module `newtonRaphson` as follows:

- `mainNewtonRaphson`: Main routine for the Newton-Raphson solver.
- `initialGuess`: Routine to set initial guess.
- `zeroState`: Set a zero/uniform initial guess when `inputData%startType = 0`.
- `anlInitialGuess`: Set a analytical initial guess when `inputData%startType = 2`.
- `nRconvergence`: Check for the Newton-Raphson convergence.
- `nRconvergenceRANS, nRconvergenceSA`: Check for the Newton-Raphson convergence of RANS and SA equations separately. Used in the staggered RANS-SA solver.

6.2.6 Time stepping module

The routines related to the time-stepping control are within the module `timeMarching` as follows:

- `mainTimeMarching`: Main routine for the time-marching scheme.
- `initialCondition`: Set the initial condition.
- `timeIntegration`: Perform the time integration calling `mainNewtonRaphson`.
- `computeCFL`: Compute the elemental CFL.
- `updateDT`: Compute the new time step Δt^n based on the `inputData%relaxStrategy`.
- `timeConvergence`: Check for time convergence in pseudo-time marching problems when `inputData%dTtype = 2`.

6.2.7 Memory preallocation module

The routines preallocation of memory are located at the module `preAllocation` as follows:

- `preAllocationGeometry`: Preallocates temporary geometrical features in a variable of type `geoDef` of Section 6.1.3.
- `preAllocationGlobal`: Preallocates tempoarary global auxiliary vectors in a variable of type `tmpGlobalDef` of Section 6.1.9.
- `preAllocationLocal`: Preallocates temporary local matrices and vectors in a variable of type `tmpLocalDef`of Section 6.1.9.
- `setPreAllocationToZero`: Set to zero the fields of the preallocated variables of type `geoDef`, `tmpGlobalDef`, and `tmpLocalDef`.

6.2.8 PETSC interface module

The routines related to the interface of the solver to the PETSC library are located in the module called `interfacePETSC`

- `solveLinearSystem`: Solve linear system of equation formed by a PETSc MPI matrix `K` and vector `F`.
- `preAllocatePETSCKandF`: Creates and preallocates the matrix `K` and vector `F`.
- `preAllocatePETSCRes`: Preallocates PETSc MPI vectors `resU`, `resRho` and `resNu`, used to store $(\mathbf{R}_{\hat{u}})_i^e$, $(\mathbf{R}_{\hat{\nu}})_i^e$, and $(\mathbf{R}_{\rho})_i^e$, appearing in Eq. (2.34).

- `storeFhuInF, storeFhnInF, storeFrhoInF`: Stores the contributions of $(\mathbf{F}_{\hat{u}})_i^e$, $(\mathbf{F}_{\rho})_i^e$, and $(\mathbf{F}_{\hat{\nu}})_i^e$ into the PETSc MPI vector `F`, appearing the in right-hand side of Eq. (2.39).
- `storeRu, storeRn, storeRrho`: Store the contributions of $(\mathbf{R}_{\hat{u}})_i^e$, $(\mathbf{R}_{\hat{\nu}})_i^e$, and $(\mathbf{F}_{\hat{\nu}})_i^e$ into the respective PETSc MPI vector `resU`, `resRho` and `resNu`.
- `storeKhuhuInK, storeKhurInK, storeKhuhnInK, storeKhnhnInK`: Store the contributions of $(\mathbf{K}_{\hat{u}\hat{u}})_{i,j}^e$, $(\mathbf{K}_{\hat{u}\rho})_i^e$, $(\mathbf{K}_{\rho\hat{u}})_i^e$, $(\mathbf{K}_{\hat{u}\hat{\nu}})_{i,j}^e$, $(\mathbf{K}_{\hat{\nu}\hat{u}})_{i,j}^e$, and $(\mathbf{K}_{\hat{\nu}\hat{\nu}})_{i,j}^e$ into the MPI matrix `K`, appearing in the left-had side of Eq. (2.39).

6.2.9 Local and global solvers modules

The routines related to the global solver are within the module `globalSolver` and are listed below:

- `mainGlobal`: The main routine driving the global solver.
- `updateMatrices`: Compute the local matrices and vectors arising from the volume/-faces elemental integration appearing in (2.38).
- `computeResiduals`: Compute the global residuals appearing in (2.34).
- `computeLocalResiduals`: Compute the local residuals appearing in (2.33).
- `mainGlobalRANS, mainGlobalSA`: The main routine driving the global solver of the RANS and SA equations respectively. Used in the staggered RANS-SA solver only.
- `computeResidualsRANS, computeResidualsSA`: Compute the global residuals of the RANS and SA respectively. Used in the staggered RANS-SA solver only.
- `computeLocalResidualsRANS, computeLocalResidualsSA`: Compute the local residuals of the RANS and SA equations respectively. Used in the staggered RANS-SA solver only.

The local solver routines are placed inside the module `locaSolver`, as listed:

- `mainLocal`: The main routine driving the local problem.
- `updateMatricesLocal`: Compute the local matrices and vectors arising from the volume/faces elemental integration appearing in (2.38).
- `mainLocalRANS, mainLocalSA`: Main local problem driver of RANS and SA respectively. Used in the staggered RANS-SA solver only.
- `updateMatricesLocalRANS, updateMatricesLocalSA`: Compute the local matrices and vectors arising from the volume/faces elemental integration of RANS and SA respectively. Used in the staggered RANS-SA solver only.

6.2.10 Postprocessing modules

The computation of the quantities of interest is handled by the routines in the module `computeQoI`, and are listed below:

- `computeDragForce`: Compute the drag force at the wall.

- `computeEnergy`: Compute solution energy.

The error computation is done using the routines in the module `errComputation` as follows:

- `computeErrorL2normK0`: Compute the error of the solution against the analytical in $\mathcal{L}^2(\Omega)$ with one gauss integration point.

- `computeErrorFaceL2normK0`: Compute the error of the solution against the analytical in $\mathcal{L}^2(\Gamma)$ with one gauss integration point.

6.3 Solver workflow

After the presentation of the solver data structures, in Section 6.1, and modular structure, in Section 6.2, we move on to explaining the solver workflow, that is, the routines calling sequence and solver structuring.

The main program routine `fcfvRansSA` is shown in Listing 6.1, where the variables and the respective derived data type adopted are defined. Also, the code flow, that is, the calling sequence of the main routines is shown.

```fortran
program fcfvRansSA
use typeDefinitions
use input
use output
use fcfvPreProcess
use newtonRaphson
use timeMarching

implicit none
    type(meshDef):: mesh
    type(faceElemDef),dimension(:), allocatable :: faceElem
    type(refElemDef), dimension(:), allocatable :: refElem
    type(nrDef):: NR
    type(fcfvDef):: fcfv
    type(inputDataDef) :: inputData
    type(timeMarchingDef):: timeMarching
    .
    ! internal variables are omitted
    integer :: ierror , processRank, nOfProcessors

    call MPI_INIT(ierror)
    call MPI_COMM_SIZE(MPI_COMM_WORLD, nOfProcessors, ierror)
    call MPI_COMM_RANK(MPI_COMM_WORLD, processRank, error)

    call setBCconstants(BC)
    call setTurbConstants(turbCST)
    call readInputData(inputData)
    call readMesh(mesh,inputData)
    call fcfvRefElem(mesh%nsd,refElem)
    call fcfvPreProcess(mesh,fcfv,faceElem,refElem)

    if      (timeMarching%transient == 0) then     !STEADY-STATE
        call mainNewtonRaphson(NR,timeMarching,mesh,fcfv,
            & faceElem,refElem,inputData)
    else if (timeMarching%transient == 1) then     !TRANSIENT
        call mainTimeMarching(NR,timeMarching,mesh,fcfv,
            & faceElem,refElem,inputData)
    end if
    .
    ! Calls to QoI, error computation, and runtime print are omitted
    call saveSolution(NR,timeMarching,mesh,fcfv,inputData)
    call MPI_FINALIZE(error)
end program fcfvRansSA
```

Listing 6.1: Main program `fcfvRansSA` showing the routines calling sequence

The listed variables are used in subsequent routine calls with either input or output intent. Additionally, we include the MPI initialization, which, while not required for the proper functioning of the PETSc solver, allows us to obtain MPI data such as the number of processors and the current process rank.

To provide an overview and facilitate the understanding of the subsequent sections we introduce in Fig. 6.3 the schematic workflow of the `Fortran 90` FCFV solver, complementing the code overview of Listing 6.1. The workflow presented in Fig. 6.3 represents the general workflow of all implementations within this work, namely Navier-Stokes or RANS-SA implementations. However, for the RANS-SA staggered case, some differences are found in the non-linear solver, which will be addressed in Section 6.3.7. The next sections will cover each of the routines called in the sequence.

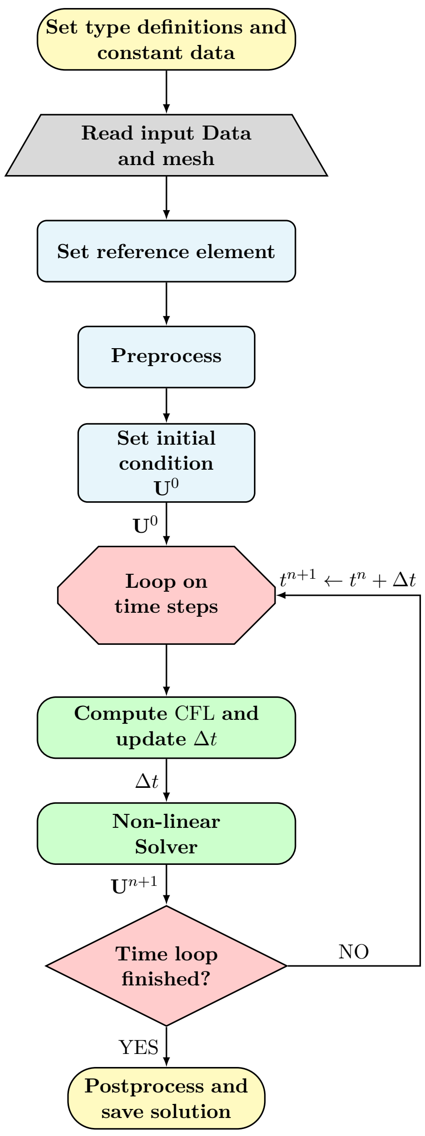

Set type definitions and constant data
Read input Data and mesh
Set reference element
Preprocess
Set initial condition $\mathbf{U}^0$
$\mathbf{U}^0$
Loop on time steps
$t^{n+1} \leftarrow t^n + \Delta t$
Compute CFL and update Δt
Δt
Non-linear Solver
$\mathbf{U}^{n+1}$
Time loop finished?
NO
YES
Postprocess and save solution

Figure 6.3: General `Fortran 90` FCFV workflow.

6.3.1 Definitions set up

As mentioned in Section 6.2.1, all derived data types used in the solver are defined in the module `typeDefinitions`. The definitions are made accessible by any solver module through the preamble command `use typeDefinitions`. Thus, no call to any routine is required to define the derived data types.

6.3.2 Set constants

After the definitions of the derived data type through `use typeDefinitions`, a call to the routines `setBCconstants` and `setTurbConstants`, of Section 6.2.1, will set the constant values of the variables `BC` and `turbCst`, respectively. With the first having a type of `bcDef`, of Section 6.1.6, and the assigned values for its components shown in Listing 6.2.

```
  subroutine setBCconstants(BC)
    type(bcDef),intent(out):: BC
        BC%iBC_Interior  = 0
        BC%iBC_Dirichlet = 1
        BC%iBC_Neumann   = 2
        BC%iBC_Outlet    = 3
        BC%iBC_Symmetry  = 4
  end subroutine setBCconstants
```

Listing 6.2: `setBCconstants` Routine

The second is a type of `turbConstantsDef`, Section 6.1.6, and will have the standard SA model constant values assigned to each of its components, as in Section 2.1. Both `BC` and `turbCst` are declared in the `typeDefinitions` module and thus are made available globally through the command `use typeDefinitions`.

6.3.3 Reading input data and mesh

The user input data are defined in an input file named after `data.inp`. All solver options, described in Section 6.1.11 are set in this file. In Listing 6.3 we see a input file snippet showing the user input data related to the problem setup, FCFV parameters, and input grid. The full `input.inp` file also contains data related to Newton-Raphson, time-marching, and output, which are not shown here.

```
&inputDataFile
!--------------------
! PROBLEM SETUP
!--------------------
inputData%ex    = 1
inputData%Rey   = 1
inputData%Lref = 1.0
inputData%Vref = 1.0
inputData%tripTermFt2 = 0
!---------------------
! FCFV PARAMETERS
!---------------------
inputData%tauType   = 2
```

```
inputData%beta      = 10.0, 10.0, 10.0, 10.0
inputData%epsilon   = 5e-2
inputData%epsilonSA = 1e-2
!----------------------------
! INPUT GRID
!----------------------------
inputData%Folder = 'dat/'
inputData%mesh   ='cube_DBC_NBC_TET_H2_P1'
inputData%FileExtension = '.txt'
.
! Similarly for Newton-Raphson, time-marching and output parameters
/
```

Listing 6.3: Portion of a input data file

Once all types are defined, a call to the routine **readInputData**, of Section 6.2.2, will read user input data and po into the variable named **inputData**, of type **inputDataDef** of Section 6.1.11, defined in the main **fcfvRansSA** show in Listing 6.1. Next, a call to the routine **readMesh**, present in the module **input** of Section 6.2.2, is performed. This call will initialize and populate the variable **mesh**, Section 6.1.1, also defined in **fcfvRansSA**.

6.3.4 Reference element definition

The next step is a call to the routine **fcfvRefElem**, of Section 6.2.1.

```
subroutine fcfvRefElem(nsd,refElem)
    implicit none
    integer, intent(in):: nsd
    type(refElemdef),dimension(:), allocatable, intent(out):: refElem

    if (nsd == 2) then
      allocate(refElem(2)) ! 1 TRI and 2 QUA
      ! 1 TRI
      refElem(1)%nOfFaces = 3
      refElem(1)%nOfNodes = 3
      refElem(1)%faceNodes = reshape([1, 2, 3, 2, 3, 1], &
        [refElem(1)%nOfFaces,2])
      refElem(1)%faceType = [1, 1, 1]
      refElem(1)%nOfFaceNodes = [2, 2, 2]
      ! Similarly for QUA, TET HEX, PRI, and PYR Reference elements
    endif
end subroutine
```

Listing 6.4: Portion of **fcfvRefElem** Routine.

In Listing 6.4, a section of the `fcfvRefElem` routine showing the initialization of the 2D reference element of the type triangle is observed.

This routine will take as input the number of spatial dimensions $\mathtt{n_{sd}}$, read from `mesh`, and `refElem` of type `refElemDef`, defined in the main `fcfvRansSA`. It will then fill the empty `refElem` with reference element data and output it. Each component of this array variable will carry reference information of each of the possible element types for a given $\mathtt{n_{sd}}$. Note that the following convection is adopted for the `typeElem`: In 2D, with (1) for triangles and (2) for quadrilaterals. In 3D, with (1) for tetrahedra, (2) for hexahedra, (3) for prisms and (4) for pyramids. For the `faceType`, the following is used: (1) for triangular face and (2) for quadrilateral face, in 3D. For 2D, `faceType` will always be (1).

6.3.5 Pre-process of input data

The next step is the preprocessing of the data read and generated in the previous steps. The routine `fcfvPreProcess`, defined in Section 6.2.1, is used for this goal.

The variable of `mesh`, already populated with the input mesh data after the read mesh routine call, has its remaining uninitialised field `indexTf` populated. Apart from that, the values of `nOfTotalFaces` and `nOfDirichletFaces` are set.

The fields `pureDirichlet`, `vDoftoSolve`, `nDofUhat`, and `nDofGlobal`, present in the variable `fcfv` of type `fcfvDef`, defined in Section 6.1.5, are also populated. Further, with a loop on all internal and external faces, the `faceInfo` field of each component of the array variable `faceElem` is set. As mentioned in Section 6.1.4, each component e of `faceElem` stores important information about the faces of the element e. These steps finalise the solver preprocessing phase.

6.3.6 Time-marching scheme

For a run with `inputData%transient = 1`, the `mainTimeMarching` routine, from the module of Section 6.2.6, is called after the preprocessing phase. For `inputData%transient = 0`, the routine `mainNewtonRaphson`, further described in Section 6.3.7, is directly called.

The routine `mainTimeMarching` is the main driver of the time stepping scheme. It starts with the allocation of the vectors of `timeMarching`, storing time-marching-related data, and is described in Section 6.1.8. The variable `NR%Res` is also allocated at this stage. The time-stepping loop is then initialised with a call to `initalCondition`, where the initial condition is decided by `inputData%startType`. The routine `timeIntegration` is then called for the first time just to set the first time-step parameters. For example, when `inputData%scheme = 2`, BDF2 is used, and the first step is forced to be a BE since only the initial guess vector state $\mathbf{U}^0$, of Eq. (2.38), is known. Otherwise, if `inputData%continuation = 1`, $\mathbf{U}^{n-1}$ and $\mathbf{U}^{n-2}$ are known and the first step is given with BDF2.

After this initial setup, the `computeCFL` routine is called, followed by a call to the routine `updateDT`. This routine will perform the time-step adaptation based on the computed CFL. If `inputData%dtType = 2`, the flag `inputData%relaxStrategy` will determine the relaxation strategy to be used. If `inputData%dtType = 1`, a constant time-step Δt is adopted, and `updatedDT` is skipped. Next, `timeIntegration` is called to perform the first actual time integration step. This routine calls `mainNewtonRaphson` to perform the Newton solver, described in Section 6.3.7, on the given time t^n. Once leaving the Newton solver, the routine `saveSolution` is called to save the time t^n solution snapshot, in case n fulfils snapshot saving frequency of `inputData%nStepsSnap`. Also, if `inputData%computeQoI` $\neq 0$, the corresponding routine to compute the quantity of interest is called, see Section 6.2.10.

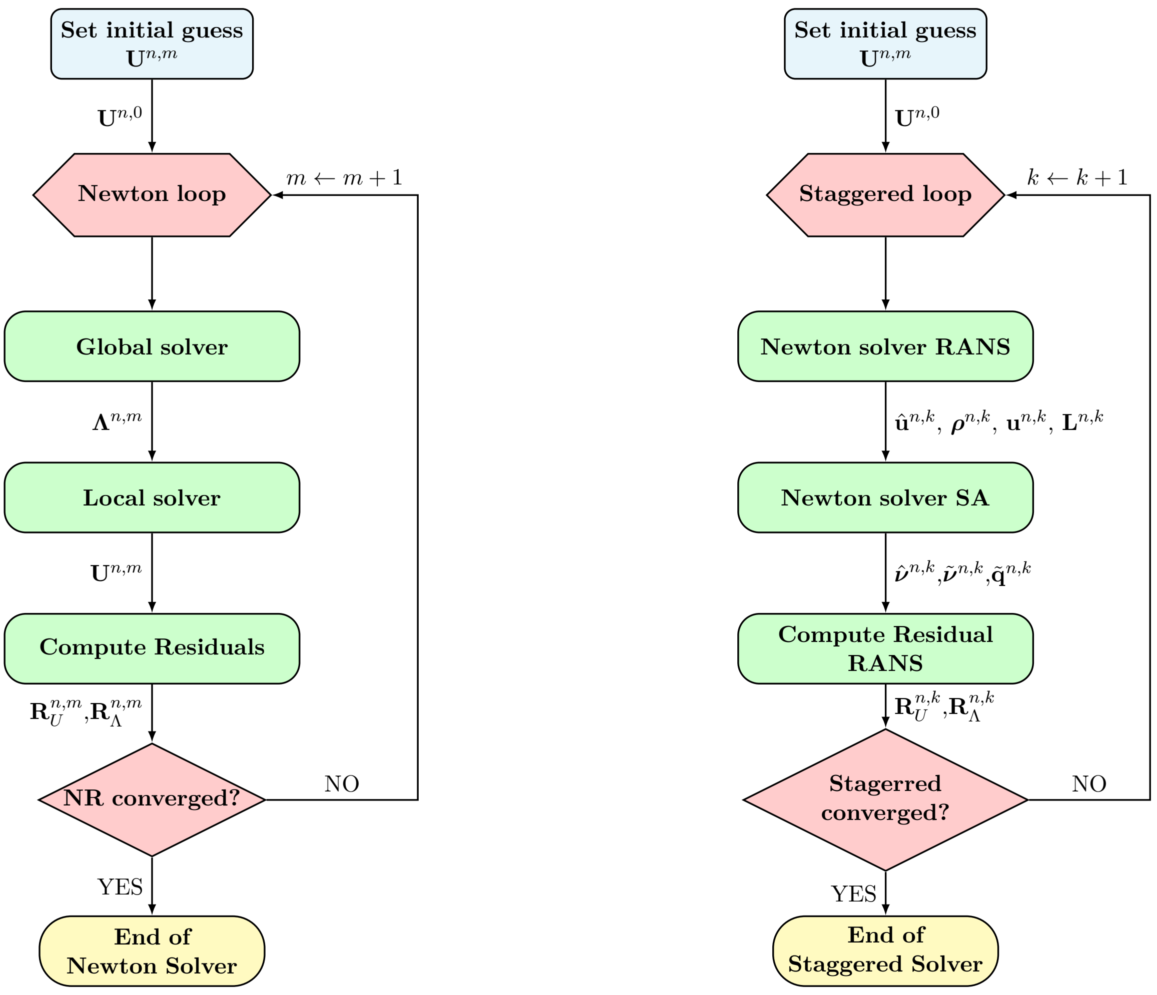
Next, back to the `mainTimeMarching` scope, the routine `timeConvergence` is called to check the steady state residuals, `timeMarching%ResTime`, against the time tolerance `inputData%timeTol`. If not converged, or `inputData%nOfSteps` is not reached, we return to the `computeCFL` routine, starting a new time step.

6.3.7 Non-linear Newton-Raphson solver

The routine responsible for the non-linear solution is `mainNewtonRaphson`, of Section 6.2.5, for both monolithic and staggered solvers. The routine starts with a call to the routine `initialGuess`, which in the case of `inputData%transient = 0` assumes the initial guess option set in `inputData%startType`. As mentioned in the previous section, the `mainNewtonRaphson` is called within each time step. In this case, `inputData%transient = 1` and two options are possible for the initial guess. In case it is the very first Newton-Raphson solve at time t^1, the initial guess will be the same as the one set in `inputData%startType`, otherwise, it is taken as the previous time t^{n-1} solution.

After the initial guess setup, the Newton-Raphson loop starts. Here we will make a distinction between the monolithic and staggered approaches. For the monolithic solver, the steps are shown in Fig. 6.4(a). A call to the `mainGlobal` is performed, this outputs the global increment solution at Newton-Raphson step m, namely $\Delta\mathbf{\Lambda}^{n,m}$ of Eq. (2.37), which is stored on the `NR` components `DeltaUHat`, `DeltaRho`, and `DeltaNuHat`, see Section 6.1.7. It also outputs the values of the solution residuals, `NR%Res`, before the solve. Following this, a call to the `mainLocal` will output the local solution $\mathbf{U}^{n,m}$. After that, with a call to `computeResiduals`, the recomputation of the residuals is performed. This is followed by a convergence check with a call to `nRconvergence`. This loop continues until `NR%Res` fulfils the `inputData%nrTol` or `inputData%nOfNrIterMax` is achieved. Note that `NR%Res(3)` and `NR%Res(6)` are not considered for this convergence check since they are the steady state residuals used to check convergence in time, see Section 6.1.7.

For the staggered solver, we add an extra loop outside the Newton-Raphson loop, referred to here as the staggered loop, as shown in Fig. 6.4(b). Note that in Fig. 6.4(b), for the


Set initial guess $\mathbf{U}^{n,m}$
$\mathbf{U}^{n,0}$
Newton loop
$m \leftarrow m+1$
Global solver
$\mathbf{\Lambda}^{n,m}$
Local solver
$\mathbf{U}^{n,m}$
Compute Residuals
$\mathbf{R}_U^{n,m}$,$\mathbf{R}_\Lambda^{n,m}$
NR converged?
NO
YES
End of Newton Solver
Set initial guess $\mathbf{U}^{n,m}$
$\mathbf{U}^{n,0}$
Staggered loop
$k \leftarrow k+1$
Newton solver RANS
$\hat{\mathbf{u}}^{n,k}$, $\boldsymbol{\rho}^{n,k}$, $\mathbf{u}^{n,k}$, $\mathbf{L}^{n,k}$
Newton solver SA
$\hat{\boldsymbol{\nu}}^{n,k}$,$\tilde{\boldsymbol{\nu}}^{n,k}$,$\tilde{\mathbf{q}}^{n,k}$
Compute Residual RANS
$\mathbf{R}_U^{n,k}$,$\mathbf{R}_\Lambda^{n,k}$
Stagerred converged?
NO
YES
End of Staggered Solver

(a) Monolithic (b) Staggered

Figure 6.4: `Fortran 90` FCFV (a) Newton-Raphson workflow for the Monolithic solver and (b) non-linear solver for the Staggered case.

staggered solver, we omitted all the steps inside each RANS and SA Newton-Raphson loop. The same steps done in the monolithic Newton-Raphon loop are performed separately for the RANS and SA equations.

The staggered loop starts with a Newton-Raphson loop on the RANS equations. A call to the `mainGlobalRANS` is performed, outputting the RANS global increment solution at Newton-Raphson step m, namely $\Delta\hat{\mathbf{u}}^{n,m,k}$ and $\Delta\boldsymbol{\rho}^{n,m,k}$, which is stored on the `NR` components `DeltaUHat` and `DeltaRho`. Following this, a call to the `mainLocalRANS` will output the local solution $\mathbf{L}^{n,m,k}$ and $\mathbf{u}^{n,m,k}$. With k denoting the staggered iteration number. After that, with a call to `computeResidualsRANS`, the recomputation of the residuals is performed. This is followed by a convergence check with a call to `nRconvergenceRANS`. This loop continues until the NS components of NR%Res fulfil the inputData%nrTol or inputData%nOfNrIterMax is achieved.

After convergence, this outputs updated local and global RANS solutions. With this RANS solution update, a Newton-Raphson solver is performed on the SA equations. A call to the mainGlobalSA is performed, this outputs the SA global increment solution at Newton-Raphson step m, namely $\Delta\hat{\boldsymbol{\nu}}^{n,m,k}$, which is stored on the NR components DeltaNuHat. Following this, a call to the mainLocalSA will output the local solution $\tilde{\mathbf{q}}^{n,m,k}$ and $\tilde{\boldsymbol{\nu}}^{n,m,k}$. After that, with a call to computeResidualsSA, the recomputation of the residuals is performed. This is followed by a convergence check with a call to nRconvergenceSA. This loop continues until the SA components of NR%Res fulfil the inputData%nrTol or inputData%nOfNrIterMax is achieved. With the updated local and global solutions of RANS and SA, a new RANS residual is computed with computeResidualsRANS. The updated residual NR%Res is checked against inputData%stgTol. The staggered loop continues until the residuals, NR%Res, fulfil inputData%stgTol or inputData%nOfStgIterMax is achieved.

6.3.8 Global Solver and PETSc interfacing

During the Newton-Raphson solver described in Section 6.3.7, the global solver routine mainGlobal is called. In this section, we will break down the routine that is the core of the Fortan 90 FCFV solver. Due to the interfacing with PETSc library, Balay et al. (2023), include directives and use petsc command needs to be added to the module globalSolver that carries the routine mainGlobal, as shown in Listing 6.5.

```
module globalSolver
#include <petsc/finclude/petsc.h>
#include <petscversion.h>

use petsc
use typeDefinitions
use algebraicOperations
use localSolver
use anlyticalNS
use anlyticalSA
use faces
use elements
use preAllocation
use stabilization
use functionsSAlinearisation
use interfacePETSC
```

Listing 6.5: Preamble of globalSolver module

Algorithm 6.1 `Fortran 90` FCFV global solver.

```
1: call PETSc initialize
2: call preallocation of temporary Global auxiliary vectors
3: call preallocation of temporary Local matrices/vectors
4: call preallocation of temporary geometrical data
5: call creation and preallocation of PETSc objects
6: for $e = 1$ until $\mathtt{n_{el}}$ do
7:     call update Local matrices
8:     for $i = 1$ until $\mathtt{n}^e_{\mathtt{fa}}$, with $i \in \mathcal{B}_e$ do
9:         $(\mathbf{K}_{\hat{u}\rho})^e_i \leftarrow (\mathbf{K}_{\hat{u}\rho})^e_i +$ ith contribution
10:        $(\mathbf{F}_{\hat{u}})^e_i \leftarrow (\mathbf{F}_{\hat{u}})^e_i +$ ith contribution
11:        $(\mathbf{F}_{\rho})^e_i \leftarrow (\mathbf{F}_{\rho})^e_i +$ ith contribution
12:        $(\mathbf{F}_{\hat{\nu}})^e_i \leftarrow (\mathbf{F}_{\hat{\nu}})^e_i +$ ith contribution
13:        $(\mathbf{R}_{\hat{u}})^e_i \leftarrow (\mathbf{R}_{\hat{u}})^e_i +$ ith contribution
14:        $(\mathbf{R}_{\rho})^e_i \leftarrow (\mathbf{R}_{\rho})^e_i +$ ith contribution
15:        $(\mathbf{R}_{\hat{\nu}})^e_i \leftarrow (\mathbf{R}_{\hat{\nu}})^e_i +$ ith contribution
16:        for $j = 1$ until $\mathtt{n}^e_{\mathtt{fa}}$, with $j \in \mathcal{B}_e$ do
17:            $(\mathbf{K}_{\hat{u}\hat{u}})^e_{i,j} \leftarrow (\mathbf{K}_{\hat{u}\hat{u}})^e_{i,j} +$ ith jth contribution
18:            $(\mathbf{K}_{\hat{u}\hat{\nu}})^e_{i,j} \leftarrow (\mathbf{K}_{\hat{u}\hat{\nu}})^e_{i,j} +$ ith jth contribution
19:            $(\mathbf{K}_{\hat{\nu}\hat{u}})^e_{i,j} \leftarrow (\mathbf{K}_{\hat{\nu}\hat{u}})^e_{i,j} +$ ith jth contribution
20:            $(\mathbf{K}_{\hat{\nu}\hat{\nu}})^e_{i,j} \leftarrow (\mathbf{K}_{\hat{\nu}\hat{\nu}})^e_{i,j} +$ ith jth contribution
21:            call storage of $(\mathbf{K}_{\hat{u}\hat{u}})^e_{i,j}$ into $\mathbb{K}$
22:            call storage of $(\mathbf{K}_{\hat{u}\hat{\nu}})^e_{i,j}$, $(\mathbf{K}_{\hat{\nu}\hat{u}})^e_{i,j}$ into $\mathbb{K}$
23:            call storage of $(\mathbf{K}_{\hat{\nu}\hat{\nu}})^e_{i,j}$ into $\mathbb{K}$
24:        end for
25:        call storage of $(\mathbf{K}_{\hat{u}\rho})^e_i$ into $\mathbb{K}$
26:        call storage of $(\mathbf{F}_{\hat{u}})^e_i$ into $\mathbb{F}$
27:        call storage of $(\mathbf{F}_{\rho})^e_i$ into $\mathbb{F}$
28:        call storage of $(\mathbf{F}_{\hat{\nu}})^e_i$ into $\mathbb{F}$
29:        call storage of $(\mathbf{R}_{\hat{u}})^e_i$ into $\mathbf{R}_{\hat{u}}$
30:        call storage of $(\mathbf{R}_{\rho})^e_i$ into $\mathbf{R}_{\hat{\nu}}$
31:        call storage of $(\mathbf{R}_{\hat{\nu}})^e_i$ into $\mathbf{R}_{\rho}$
32:    end for
33:    call set temporary variables to zero
34: end for
35: call PETSc objects assembly for $\mathbb{K}$, $\mathbb{F}$, $\mathbf{R}_{\hat{u}}$, $\mathbf{R}_{\hat{\nu}}$, $\mathbf{R}_{\rho}$
36: call linear system solver for $\mathbb{K}\Delta\mathbf{\Lambda} = \mathbb{F}$
37: call deallocate temporary variables
38: call Destroy PETSc objects
39: call PETSc finalize
```

Before diving into the routine workflow and calling sequence, we introduce a pseudo-code of the global solver, shown in Algorithim 6.1.

The routine `mainGlobal` starts with a PETSc initialise call with `PetscInitialize`. It is the PETSc native routine that initiates the PETSc database and MPI. Next, the preallocation routines are called, as seen in Listing 6.6. We see the definitions of the temporary auxiliary variables `tmpG`, `tmpL`, and `geo`, as well as the PETSc variables. The variable definition is followed by the preallocation routines, discussed in the following

```fortran
subroutine mainGlobal(NR,timeMarching,mesh,fcfv,faceElem, &
  refElem,inputData)
    ! Part of the preamble is omitted
    type(tmpGlobalDef), dimension(:), allocatable :: tmpG
    type(tmpLocalDef), dimension(:), allocatable :: tmpL
    type(geoDef), dimension(:), allocatable :: geo
    integer :: nDofU, nDofNu, nDofRho

    ! PETSc variables
    Mat  A
    Vec rhs, resU, resNu, resRho
    MPI_Comm comm
    PetscInt :: process_Rank, size_Of_Cluster
    PetscErrorCode :: ierr
    PetscScalar :: matValues, cpuTSolvePETSC
    ! variables initialisation are ommitted
    ! Intitialise PETSC
    call PetscInitialize(PETSC_NULL_CHARACTER,ierr)
    !PreAllocation of global system vector/matrices per element type
    call preAllocationGlobal(tmpG,mesh%nsd,refElem)
    !PreAllocation of local  vector and matrices per element type
    call preAllocationLocal(tmpL,mesh%nsd,refElem)
    !PreAllocation of geometrical  data per element type
    call preAllocationGeometry(geo,mesh%nsd,refElem)
    ! Creat Petsc matrix and vector
    call preAllocatePETSCAndRHS(A,rhs,size(fcfv%vDOFtoSolve))
    call preAllocatePETSCRes(resU,resNu,resRho,nDofU,nDofNu,nDofRho)
    ! Routine body is omitted
end subroutine
```

Listing 6.6: Portion of `mainGlobal` routine showing the initial calls.

The routine `preAllocationGlobal` will preallocate the derived data type array `tmpG` of type `tmpGlobalDef` in 6.1.9 for all possible element types given by `mesh%nsd`. For example, in a 2D case where $\mathtt{n_{sd}} = 2$, temporary triangular and quadrilateral variables will be preallocated. In Listing 6.7, the preallocation of the `tmpG` for a triangular element is shown. For the other element types, the allocation structure is the same, which means that for quadrilaterals, the `tmpG(2)` component is allocated instead of `tmpG(1)`. The same follows for 3D elements, where `tmpG(1)`, `tmpG(2)`, `tmpG(3)`, and `tmpG(4)` are allocated for tetrahedral, hexahedral, prims, and pyramids element types, respectively.

```
  subroutine preAllocationGlobal(tmpG,nsd,refElem)
    implicit none
    type(tmpGlobalDef),dimension(:),allocatable,intent(inout) :: tmpG
    integer, intent(in) :: nsd
    type(refElemDef), dimension(:), intent(in) :: refElem
    integer :: nOfFaces

    if (nsd == 2) then
      allocate(tmpG(2))
      nOfFaces = refElem(1)%nOfFaces
      allocate(tmpG(1)%uHatE(nOfFaces*nsd))
      allocate(tmpG(1)%uHatE0(nOfFaces*nsd))
      allocate(tmpG(1)%NuHatE(nOfFaces))
      allocate(tmpG(1)%NuHatE0(nOfFaces))
      allocate(tmpG(1)%idGlobalAssembly(nOfFaces*nsd))
      allocate(tmpG(1)%idGlobalAssemblyRd(nOfFaces*nsd))
      allocate(tmpG(1)%idGlobalFace(nOfFaces))
      allocate(tmpG(1)%idGlobalFaceRd(nOfFaces))
      tmpG(1)%uHatE    = 0
      tmpG(1)%uHatE0   = 0
      tmpG(1)%NuHatE   = 0
      tmpG(1)%NuHatE0  = 0
      tmpG(1)%idGlobalAssembly   = 0
      tmpG(1)%idGlobalAssemblyRd = 0
      tmpG(1)%idGlobalFace     = 0
      tmpG(1)%idGlobalFaceRd   = 0
      ! Remaining is omitted
end subroutine
```

Listing 6.7: Portion of `preAllocationGlobal` routine showing the preallocation of global auxiliary variables for triangles.

Following the same rationale, the preallocation of the local matrices in `tmpLocal`, introduced in 6.1.9, is done (see Appendix C.1 for their full definition). The preallocation for triangular elements is seen in Listing 6.8.

Only the preallocation of the matrices $\mathbf{T}_{uu}$,$\mathbf{T}_{uu}^{-1}$, and $\mathbf{T}_{u\hat{u}}$, appearing in Eq. (2.38) is shown for the sake of brevity. However, the preallocation of all other local matrices arising from the FCFV linearisation for all element types in 2D and 3D follows the same rationale.

The preallocation of the geometrical data also follows the same idea, as shown in Listing 6.9, where the memory preallocation for the geometrical features of triangular elements is shown.

```
  subroutine preAllocationLocal(tmpL,nsd,refElem)
    implicit none
    type(tmpLocalDef),dimension(:),allocatable,intent(inout) :: tmpL
    type(refElemDef),dimension(:), intent(in) :: refElem
    integer, intent(in) :: nsd
    integer :: nOfFaces, nDofElemU, nDofElemL
    nOfacesUnknowns = nsd
    nDofElemU       = nsd
    nDofElemL       = nsd*nsd
    if (nsd == 2) then
      allocate(tmpL(2))
      nOfFaces = refElem(1)%nOfFaces
      allocate(tmpL(1)%Tuu(nsd,nsd))
      allocate(tmpL(1)%TuuInv(nsd,nsd))
      allocate(tmpL(1)%Tuhu(nDofElemU,nOfFaces*nsd))
      ! Remaining is omitted
end subroutine
```

Listing 6.8: Portion of `preAllocationLocal` routine showing the allocation of matrices/arrays, $\mathbf{T}_{uu}$,$\mathbf{T}_{uu}^{-1}$, and $\mathbf{T}_{u\hat{u}}$, for triangular element.

```
 subroutine preAllocationGeometry(geo,nsd,refElem)
    implicit none
    type(geoDef), dimension(:), allocatable, intent(inout) :: geo
    type(refElemDef),dimension(:), intent(in) :: refElem
    integer, intent(in) :: nsd
    integer :: nOfFaces
    if (nsd == 2) then
      allocate(geo(2))
      nOfFaces = refElem(1)%nOfFaces
      allocate(geo(1)%gammaJ(nOfFaces))
      allocate(geo(1)%nJ(nsd,nOfFaces))
      allocate(geo(1)%Xfm(nsd,nOfFaces))
      allocate(geo(1)%Xm(nsd))
      geo(1)%Omega  = 0
      geo(1)%gammaJ = 0
      geo(1)%nJ     = 0
      geo(1)%Xfm    = 0
      geo(1)%Xm     = 0
      ! Remaining is omitted
end subroutine
```

Listing 6.9: Portion of `preAllocationGeometry` routine showing the preallocation of geometrical data for triangles.

After the preallocation of `tmpG`, `tmpL`, and `geo`, these variables are ready to be used within the global loop over elements. For each element, only `tmpG(typeElem)`, `tmpL(typeElem)`, and `geo(typeElem)` will be populated. Before moving to the next element, the fields are reset to zero by calling `setPreAllocationToZero`. This strategy eliminates the need to deallocate and reallocate variables when `typeElem` is not unique in the mesh. Additionally, it enables seamless on-the-fly computation of local matrices and geometrical features for single-type cell or hybrid grids.

Next, in the `preAllocatePETSCKandF` routine, MPI PETSc variables are created and preallocated. This routine creates the global matrix $\mathbb{K}$ and the right-hand side vector $\mathbb{F}$, of Eq. (2.39). Both $\mathbb{K}$ and $\mathbb{F}$ set up the linear system solver, being the only parallel section of the `Fortran 90` FCFV implementation.

```fortran
  subroutine preAllocatePETSCKandF(K,F,n)
    implicit none
    integer, intent(in) :: n
    ! PETSc variables
    PetscErrorCode :: ierr
    PetscInt :: localSize
    Mat, intent(out) :: K
    Vec, intent(out) :: F
    MPI_Comm comm
    comm = MPI_COMM_WORLD
    ! Matrix Creation
    call MatCreate(comm,K,ierr)
    CHKERRQ(ierr)
    call MatSetType(K,MATMPIAIJ,ierr)
    CHKERRQ(ierr)
    call MatSetSizes(K,PETSC_DECIDE,PETSC_DECIDE,n,n,ierr)
    CHKERRQ(ierr)
    ! PREALLOCATE MATRIX
    call MatMPIAIJSetPreallocation(K,30,PETSC_NULL_INTEGER, &
     30,PETSC_NULL_INTEGER,ierr)
    CHKERRQ(ierr)
    call MatGetLocalSize(K,localSize,PETSC_NULL_INTEGER,ierr)
    CHKERRQ(ierr)
    call VecCreateMPI(comm,localSize,PETSC_DECIDE,F,ierr)
    CHKERRQ(ierr)
end subroutine
```

Listing 6.10: `preAllocationAandF` routine

The routine, shown in Listing 6.10, begins with a call to the PETSc `MatCreate` function to create the matrix `K`. The matrix type is set to `MATMPIAIJ` for an MPI matrix, using PETSc

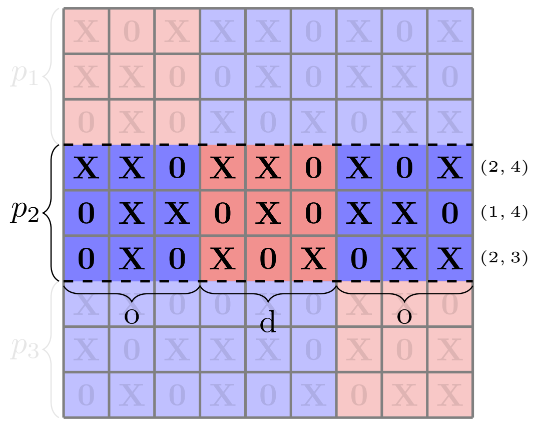


Figure 6.5: Partitioning of a sparse MPI matrix between three processors, p_n, in PETSc. In red is the diagonal (d) block and in blue are the off-diagonal (o) blocks, local to each processor. The non-zero entries, **X**, on each row owned by processor p_2 are highlighted. The total number of non-zero entries per row within the diagonal and off-diagonal blocks is shown as (`nnz_d`, `nnz_o`).

`MatSetType`. The global size of the matrix, given by `n`, is set using PETSc `MatSetSizes`. The matrix size on each processor, or local sizes, is determined by PETSc using the `PETSC_DECIDE` option.

In PETSc, the matrix is partitioned between the processes in a row-wise manner, as shown in Fig. 6.5. This means that each process owns a number of rows corresponding to the local size of the matrix. The preallocation of memory to store the matrix nonzeros is based on the number of nonzero entries on each process-owned row and is handled by the `MatMPIAIJSetPreallocation` routine. Proper preallocation is a crucial aspect of an efficient PETSc solver as it prevents runtime memory mallocs, which would otherwise significantly slow down the algorithm. The nonzeros per row are divided into diagonal (d) and off-diagonal (o) nonzeros, as illustrated in Fig. 6.5. There are two options for specifying the number of nonzero to preallocate.

The first option is to provide two vectors, `nnz_d` and `nnz_o`, that specifies the number of nonzero for each row in the diagonal and off-diagonal submatrices. This option allows the specification of the exact number of nonzeros on each matrix row. The second option is to provide two scalars, `nz_d` and `nz_o` that will apply to all rows, to indicate the number of nonzeros per row in the diagonal and off-diagonal blocks. In our implementation, we use the second option, setting `n_z = n_o = 60`, it is 60 nonzero for the rows of both the diagonal and off-diagonal blocks, as shown in Listing 6.10. This slight overestimation, based on the stencil of all possible element types, ensures that no memory allocations will be needed during runtime.

Afterwards, PETSc `MatGetLocalSize` is called to obtain the local size of `K` on each process, `localSize`, which is then used for constructing the MPI `F` vector. Note that the local size of F must have the same value as the local size of the matrix chunk owned by the respective processor. That is, the F local size should match the number of rows owned by the processor.

Back to the scope of mainGlobal, Listing 6.6, the call to the preAllocatePETSCRes routine preallocates the vectors resU, resNu, and resRho, that stores global residuals $\mathbf{R}_{\hat{u}}$, $\mathbf{R}_{\hat{\nu}}$, and $\mathbf{R}_{\rho}$, respectively. In this routine, Listing 6.11, the global size of each vector is set by nDofU, nDofNu, and nDofRho, being the respective number of degree of freedoms to solve for each global variable. Note that the PETSC_DECIDE directive is used for the local size. Since those vectors are not used to solve the linear system, there is no requirement for the local size to match the local size of the matrix K.

```
subroutine preAllocatePETSCRes(resU,resNu,resRho,nDofU,nDofNu,nDofRho)
  implicit none
  integer, intent(in) :: nDofU, nDofRho, nDofNu
  ! PETSc variables
  PetscErrorCode :: ierr
  Vec, intent(out) :: resU, resNu, resRho
  MPI_Comm comm
  comm = MPI_COMM_WORLD
  ! Creation of Residual MPI vectors
  call VecCreateMPI(comm,PETSC_DECIDE,nDofU,resU,ierr)
  call VecCreateMPI(comm,PETSC_DECIDE,nDofNu,resNu,ierr)
  call VecCreateMPI(comm,PETSC_DECIDE,nDofRho,resRho,ierr)
end subroutine
```

Listing 6.11: preAllocatePETSCRes routine.

After the preallocation phase, we finally enter the loop on elements to perform the global matrix $\mathbb{K}$ and the right-hand side vector $\mathbb{F}$ build and assemble. We remind you that the assembly process is made serially, but MPI is initialised to allow the construction of MPI PETSc objects that will be used in the parallel solution of the linear system of equations.

Before entering the loop on element e faces interactions, namely i, j loops, a call to the routine updateMatrices is performed. The matrices and vectors built on these routines are some of the local ones stored in tmpL(typeElem). Due to the static condensation of Eq. (2.39), some local matrices and vectors need to be computed to allow the construction of both $\mathbb{K}$ and $\mathbb{F}$. See Appendix C for details on those vectors/matrices computation. Note that the construction of the matrices/vectors described in Appendix C is based on exact zeroth order volume/face integral computations.

To provide the reader with an overview of how the updateMatrices routine works, we will break it down in the following. It starts with a call to computeElementalGeometry, performing the computation of the element e geometrical features. The computed data is stored in geo(typeElem), to be used on the subsequent matrices and vector construction, as seen in Listing 6.12.

```fortran
subroutine updateMatrices(NR,timeMarching,mesh,fcfv,faceElem,&
    refElem, inputData,iElem,tmpG,matL,geo)
    implicit none
    ! Internal and input variables declaring omitted
    ! omitted lines
    ! call to compute elemental geometry
    call computeElementalGeometry(iElem,geo,mesh,faceElem,refElem)
    ! Element geometrical data
    Omega = geo(typeElem)%Omega
    Xm    = geo(typeElem)%Xm
    d     = computeElementD(Xm,nsd)
    ! omitted lines
  end subroutine
```

Listing 6.12: Portion of `updateMatrices` routine showing the geometrical computation.

To demonstrate the integral computations in the matrix/vector building, we start looking into the blocks $\mathbf{T}^{n,e}_{uu}$, $\mathbf{T}^{n,e}_{\tilde{\nu}\tilde{\nu}}$, and $\mathbf{T}^{n,e}_{\tilde{\nu}\hat{u}}$, arising from the linearisation of Eq. (2.33) and appearing in Eq. (2.38). The full expression of those blocks can be found in Eq. (C.3). As an example, the computation of the transient contribution of $\mathbf{T}^{n,e}_{uu}$ and $\mathbf{T}^{n,e}_{\tilde{\nu}\tilde{\nu}}$ (`Tuu` and `Tnn` in the code) terms at the element barycenter is shown in Listing 6.13.

```fortran
subroutine updateMatrices(NR,timeMarching,mesh,fcfv,faceElem,&
    refElem,inputData,iElem,tmpG,matL,geo)
    ! Internal and input variables declaring omitted
    ! omitted lines
    ! Element Barycenter contributions
    if(timeMarching%transient == 1) then ! Transient contributions
        lambda = Omega/timeMarching%dT(iElem)
        mat(typeElem)%Tuu = mat(typeElem)%Tuu + &
            timeMarching%intCoef(1)*lambda*eyeNsd
        mat(typeElem)%Tnn = mat(typeElem)%Tnn + &
            timeMarching%intCoef(1)*lambda
        ! omitted lines
    end if
    ! omitted lines
  end subroutine
```

Listing 6.13: Portion of `updateMatrices` routine showing the computation of the transient contribution to $\mathbf{T}^{n,e}_{uu}$, and $\mathbf{T}^{n,e}_{\tilde{\nu}\tilde{\nu}}$.

In another snippet example, the computation of the contribution of some of the SA functions and derivatives to build $\mathbf{T}^{n,e}_{\tilde{\nu}\tilde{\nu}}$(`Tnn`) is shown in Listing 6.14.

Next, a loop on the element faces will account for face contributions on the local Jacobian blocks. In the snippet example of Listing 6.15 the partial construction of $\mathbf{T}^{n,e}_{\tilde{\nu}\hat{u}}$ (`Tnuhu`) on Neumann faces is shown. The face type of a given face `iFace` of the current element `iElem` is accessed by `iBC = faceElem(iElem)%faceInfo(3,iFace)`, as observed in Listing 6.15.

```fortran
subroutine updateMatrices(NR,timeMarching,mesh,fcfv,faceElem,&
    refElem,inputData,iElem,tmpG,matL,geo)
    ! omitted lines
    ! SA functions and derivatives
    St   = computeSt(mesh,NR,timeMarching,iElem,d)
    DSt  = computeDSt(mesh,NR,iElem,d)
    ft2  = computeft2(mesh,NR,timeMarching,iElem)
    Dft2 = computeDft2(mesh,NR,timeMarching,iElem)
    ! omitted lines
    mat(typeElem)%Tnn = mat(typeElem)%Tnn -
        & Omega*turbCst%Cb1*(1.0_dp - ft2)*St
    mat(typeElem)%Tnn = mat(typeElem)%Tnn -
        & Omega*turbCst%Cb1*(1.0_dp - ft2)*DSt*Nue
    mat(typeElem)%Tnn = mat(typeElem)%Tnn +
        & Omega*turbCst%Cb1*Dft2*St*Nue
    ! omitted lines
  end subroutine
```

Listing 6.14: Portion of `updateMatrices` routine showing the computation of some of the SA functions and derivatives building $\mathbf{T}^{n,e}_{\tilde{\nu}\tilde{\nu}}$.

```fortran
subroutine updateMatrices(NR,timeMarching,mesh,fcfv,faceElem,&
    refElem,inputData,iElem,tmpG,matL,geo)
    ! omitted lines
    do iFace = 1, nOfFaces ! Element faces contributions
      iBC = faceElem(iElem)%faceInfo(3,iFace)
      ! Face geometrical data
      gammaJ = geo(typeElem)%gammaJ(iFace)
      nJ     = geo(typeElem)%nJ(:,iFace)
      Xfm    = geo(typeElem)%Xfm(:,iFace)
      if (iBC == ctt%iBC_Neumann) then
        mat(typeElem)%Tnhu(faceComp) = mat(typeElem)%Tnhu(faceComp)+&
            gammaJ*nJ*NuHatFace
      endif
    enddo
        ! omitted lines
  end subroutine
```

Listing 6.15: Portions of `updateMatrices` routine showing the partial construction of $\mathbf{T}^{n,e}_{\tilde{\nu}\hat{u}}$ on Neumann faces.

After the computation of the vectors and matrix of `tmpL(typeElem)`, we come back to the scope of `mainGlobal` where the loop on the element i, j faces interaction starts. Here is where the build and assembly of $\mathbb{K}$ and $\mathbb{F}$, from Eq. (2.39), happens. For a given $i, j \in \mathcal{B}_e$ pair, the following blocks of $\mathbb{K}$ are computed: $(\mathbf{K}_{\hat{u}\hat{u}})^{e,n}_{i,j}$ of size $\mathtt{n_{sd}} \times \mathtt{n_{sd}}$, $(\mathbf{K}_{\hat{u}\rho})^{e,n}_{i}$ of size $\mathtt{n_{sd}} \times 1$, $(\mathbf{K}_{\hat{u}\hat{\nu}})^{e,n}_{i,j}$ of size $\mathtt{n_{sd}} \times 1$, $(\mathbf{K}_{\rho\hat{u}})^{e,n}_{i}$ of size $1 \times \mathtt{n_{sd}}$, $(\mathbf{K}_{\hat{\nu}\hat{u}})^{e,n}_{i,j}$ of size $1 \times \mathtt{n_{sd}}$, and $(\mathbf{K}_{\hat{\nu}\hat{\nu}})^{e,n}_{i,j}$ of size 1×1.

In Listing 6.16 we can see the exact construction of some of the contributions to the $(\mathbf{K}_{\hat{u}\hat{u}})^{n,e}_{i,j}$, $(\mathbf{K}_{\hat{u}\hat{\nu}})^{n,e}_{i,j}$, $(\mathbf{K}_{\hat{\nu}\hat{u}})^{n,e}_{i,j}$, and $(\mathbf{K}_{\hat{\nu}\hat{\nu}})^{n,e}_{i,j}$ blocks, being `Khuhu`, `Khuhn`, `Khnhu` and `Khnhn`, respectively. Where `tauITuuInv` is an auxiliary variable for $\boldsymbol{\tau}^{n-1}[\mathbf{T}^{e,n}_{uu}]^{-1}$ product. The variables `LeM` and `LeMT` are the elemental mixed variable in matrix form and its transpose, respectively, while `fv` is the SA function f_v. The full expression for each of the blocks of $\mathbb{K}$ and $\mathbb{F}$ is found in Eq. (C.11) and Eq. (C.12).

```
Khuhu = Khuhu + gamaJ*matmul(tauITuuInv,tauJ)
Khuhu = Khuhu - tauITuuInv*gamaJ*sum(uHatFaceJ*nJ)
! omitted lines
Khuhn = Khuhn - gamaJ*matmul(tauITuuInv,matmul(LeM + LeMT,nJ))/ &
        (Rey)*(4.0_dp-3.0_dp*fv)*fv
! omitted lines
Khnhu = Khnhu - tauTI*mat(typeElem)%Tnhu(fCompJ)/mat(typeElem)%Tnn
! omitted lines
Khnhn = Khnhn + gamaJ*tauTI*tauTJ/mat(typeElem)%Tnn
Khnhn = Khnhn - gamaJ*tauTI*sum(uHatFaceJ*nJ)/mat(typeElem)%Tnn
```

Listing 6.16: Portions of `mainGlobal` routine showing the partial construction of $(\mathbf{K}_{\hat{u}\hat{u}})^{n,e}_{i,j}$, $(\mathbf{K}_{\hat{u}\hat{\nu}})^{n,e}_{i,j}$, $(\mathbf{K}_{\hat{\nu}\hat{u}})^{n,e}_{i,j}$ and $(\mathbf{K}_{\hat{\nu}\hat{\nu}})^{n,e}_{i,j}$.

For the right-hand side vector $\mathbb{F}$, from Eq. (2.39), the following blocks are computed: $(\mathbf{F}_{\hat{u}})^{e}_{i}$ of size $\mathtt{n_{sd}}$, $(\mathbf{F}_{\rho})^{e}_{i}$ of size 1, and $(\mathbf{F}_{\hat{\nu}})^{e}_{i}$ of size 1. The blocks of the respective residual vectors, $(\mathbf{R}_{\hat{u}})^{e}_{i}$ of size $\mathtt{n_{sd}}$, $(\mathbf{R}_{\hat{\nu}})^{e}_{i}$, of size 1, and $(\mathbf{R}_{\rho})^{e}_{i}$ of size 1, are also computed at this stage.

The computed blocks, for a given i, j pair, namely $(\mathbf{K}_{\hat{u}\hat{u}})^{e}_{i,j}$, $(\mathbf{K}_{\hat{u}\rho})^{e,n}_{i}$, $(\mathbf{K}_{\hat{u}\hat{\nu}})^{e,n}_{i,j}$, $(\mathbf{K}_{\rho\hat{u}})^{e,n}_{i}$, $(\mathbf{K}_{\hat{\nu}\hat{u}})^{e,n}_{i,j}$ and $(\mathbf{K}_{\hat{\nu}\hat{\nu}})^{e,n}_{i,j}$ are stored into `K` through storage routines within the `interfacePETSC` module. The respective routines are `storeKhuhuInK`, `storeKhurInK`, `storeKhuhnInK`, and `storeKhnhnInK`, described in Section 6.2.8.

In Listing 6.17, the routine `storeKhuhuInK`, responsible for storing the input `Khuhu` block into `K` is shown. The input `indGlobalAsmbly` stores the global degrees of freedom of the current element, while `fCompI` and `fCompJ` are the corresponding components of the pair i, j. The routine starts with a call to the PETSc `MaGetOwnershipRange` that outputs two integer values, named here as `first` and `last`, with the values of the starting and end indexes of the `K` rows owned by the current processor.

Next, the input `Khuhu` block is written in as a vector of size $\mathtt{n}_{\mathtt{sd}}^2 \times 1$ and stored in `KhuhuBlock`. The respective i, j global indexes of the `KhuhuBlock` entries are stored in `iIndex` and `jIndex`. The blocks owned by the current processor are stored in a `do` loop.

```
subroutine storeKhuhuInK(K,Khuhu,nsd,fCompI,fCompJ,indGlobalAsmbly)
  implicit none
  real(kind = dp), intent(in):: Khuhu(nsd,nsd)
  real(kind = dp) :: KhuhuBlock(nsd*nsd)
  integer,intent(in):: nsd,fCompI(nsd),fCompJ(nsd),indGlobalAsmbly(:)
  integer :: iIndex(nsd*nsd), jIndex(nsd*nsd), compI(nsd), compJ(nsd)
  integer ::  i, first, last, matIndexi, matIndexJ

    Mat, intent(inout) :: K
    PetscScalar :: matValues
    PetscErrorCode :: ierr
    ! Get process matrix K ownership range
    call MatGetOwnershipRange(K,first,last,ierr)
    CHKERRQ(ierr)
    ! set block entries and indexes
    KhuhuBlock = reshape(transpose(Khuhu), [nsd*nsd])
    compI = indGlobalAsmbly(fCompI)
    compJ = indGlobalAsmbly(fCompJ)
    iIndex = reshape(spread(compI,1,nsd),[nsd*nsd])
    jIndex = reshape(spread(compJ,2,nsd),[nsd*nsd])
    ! storage loop
  do i = 1,nsd*nsd
      matValues = KhuhuBlock(i) ! Entries
      matIndexi = iIndex(i) - 1 ! PETSc K indexes
      matIndexj = jIndex(i) - 1 ! PETSc K indexes
      ! check entries process ownership
   if ( matIndexi >= first .and. matIndexi < last) then
    call MatSetValue(K,matIndexi,matIndexj,matValues,ADD_VALUES,ierr)
   end if
  end do
end subroutine
```

Listing 6.17: `storeKhuhuinK` routine.

Note that the indexes in `iIndex` and `jIndex` are shifted by -1 to generate `matIndexi` and `matIndexj`, as PETSc object indexing starts from 0. Following this, a `if` clause uses the integers `first` and `last` to verify whether the current processor owns the block entry. This is crucial because storing entries on processors that do not own the entry significantly reduces PETSc's efficiency, requiring more data exchange between processors during assembly. Finally, a call to PETSc `MatSetValues` will store the values into `K` with a `ADD_VALUES` flag to allow the sum of all values entered to the same pair of indexes being added during the assembly of all processes contribution. The routines `storeKhurInK`, `storeKhuhnInK`, and `storeKhnhnInK` follow that same rationale and are not shown for the sake of brevity.

For a given i the assembly of $(\mathbf{F}_{\hat{u}})_i^{e,n}$, $(\mathbf{F}_{\rho})^{e,n}$, $(\mathbf{F}_{\hat{\nu}})_i^{e,n}$ into `F` are performed with the routines `storeFhuInF`, `storeFrhoInF`, and `storeFhnInF`. In Listing 6.18, the `storeFhuInF` routine is shown.

```fortran
subroutine storeFhuinF(F,Fhu,nsd,fCompI,indGlobalAsmbly)
  implicit none
  real(kind = dp), intent(in):: Fhu(nsd)
  integer, intent(in) :: nsd, fCompI(nsd), indGlobalAsmbly(:)
  integer ::  i, first, last, compI(nsd), vecIndexi
  Vec, intent(inout) :: F
  PetscScalar :: vecValues
  PetscErrorCode :: ierr
  ! Get process vector F ownership range
  call VecGetOwnershipRange(F,first,last,ierr)
  CHKERRQ(ierr)
  compI = indGlobalAsmbly(fCompI)
  ! storage loop
  do i = 1,nsd
    vecValues = Fhu(i)
    vecIndexi = compI(i) - 1
    ! check entries process ownership
    if (vecIndexi >= first .and. vecIndexi < last) then
      call VecSetValue(F, vecIndexi,vecValues, ADD_VALUES,ierr)
    endif
  end do
end subroutine
```

Listing 6.18: `storeFhuInF` routine.

As explained before, the ownership range of the current process is checked, and the values are set into `F` if the current processor owns the indexes of the entries. The routines `storeFrhoInF` and `storeFhnInF` will follow the same rationale. Similarly, for a given i the assembly of $(\mathbf{R}_{\hat{u}})_i^{e,n}$, $(\mathbf{R}_{\hat{\nu}})_i^{e,n}$, and $(\mathbf{R}_{\rho})_i^{e,n}$ into `resU`, `resRho`, `resRho` will be done in the routines `storeRu`, `storeRn`, and `storeRrho`.

Before moving to a new element e in the `do` loop, the temporary variables `tmpL`, `tmpG`, and `geo` will have their fields set to zero to avoid spurious data at the beginning of a new element computation. This is done with the routine `setPreAllocationToZero`.

After the loop on all elements is completed, the assembly of all PETSc objects is performed to account for the entries added by all processors. This is handled by the respective PETSc matrix and vector assembly calls, as seen in Listing 6.19.

```
    call MatAssemblyBegin(K,MAT_FINAL_ASSEMBLY,ierr)
    call MatAssemblyEnd(K,MAT_FINAL_ASSEMBLY,ierr)
    call VecAssemblyBegin(F,ierr)
    call VecAssemblyEnd(F,ierr)
    call VecAssemblyBegin(resU,ierr)
    call VecAssemblyEnd(resU,ierr)
    call VecAssemblyBegin(resNu,ierr)
    call VecAssemblyEnd(resNu,ierr)
    call VecAssemblyBegin(resRho,ierr)
    call VecAssemblyEnd(resRho,ierr)
```

Listing 6.19: Portion of `mainGlobal` routine showing the PETSc objects final assemble.

When the build and assembly of `K` and `F` are finalised, we are ready to move to the solution of the linear system of equations by calling the routine `solveLinearSystem`. The routine initialisation and solver setup are shown in Listing 6.20.

```
subroutine solveSparseMatrix(K,F,solGlobal, n,cpuTSolvePETSC)
    implicit none
    ! Internal and input variables omitted
    call VecDuplicate(F,Sol,ierr)!Create MPI vector to store solution
    call KSPCreate(comm,Solver,ierr).       !Create KSP solver
    call KSPSetOperators(Solver,K,K,ierr)   !Set matrix for KSP solver
    call KSPSetType(Solver,KSPPREONLY,ierr)!Set KSP type
    call KSPGetPC(Solver,Prec,ierr)         !Set precond. operator
    call PCSetType(Prec,PCLU,ierr)          !Preconditioner type LU
    call PCFactorSetMatSolverType(Prec,MATSOLVERMUMPS,ierr)!Set MUMPS
    call PetscTime(startTime, ierr)      ! Get the starting time
    call KSPSolve(Solver,F,Sol,ierr)     ! Solve Linear System
    call PetscTime(endTime, ierr)        ! Get the ending time
    call VecAssemblyBegin(Sol,ierr)      ! Assemble MPI Sol vector
    call VecAssemblyEnd(Sol,ierr)
```

Listing 6.20: Portion of `solveLinearSystem` routine showing the initial setup and solve.

It starts with the creation of the MPI vector through `Sol` PETSc `VecDuplicate` to store the solution. Next, the KSP PETSc solver context and KSP matrix operator are created through PETSc `KSPCreate` and `KSPSetOperators`. To allow the use of direct solvers, the PETSc `KSPSetType` is called with the option `KSPPREONLY`. This applies the preconditioner only once.

Next, the preconditioner context is defined and the preconditioner type is set to `PCLU` by the calls to PETSc `KSPGetPC` and PETSc `PCSetType`, respectively. This means that a complete LU will be used, characterising a direct solver. The MUMPS solver is chosen with a call to PETSc `PCFactorSetMatSolverType` with the option `MATSOLVERMUMPS`. Finally, the solve is performed through PETSc `KSPSolve`, and the result is assembled into `Sol`.

Since the `Fortran 90` FCFV implementation features a serial workflow, except for the linear system solver phase, the distributed solution stored at the MPI vector `Sol` needs to be made fully available in all n_p processors. This is accomplished by the creation of a PETSc scatter context that scatters the distributed solution in `Sol`, having a size of n/n_p, to a sequential vector `SolAll` with size n. Finally, the PETSc object data in `SolAll` is copied to a non-PETSc serial object `solGlobal` to be used outside the `solveLinearSytem` scope, as seen in Listing 6.21.

```
    ! Scatter Solution to sequential vector
    call VecScatterCreateToAll(Sol, scatter_sol, SolAll, ierr)
    call VecScatterBegin(scatter_sol, Sol, SolAll, &
         INSERT_VALUES, SCATTER_FORWARD, ierr)
    call VecScatterEnd(scatter_sol, Sol, SolAll, &
         INSERT_VALUES, SCATTER_FORWARD, ierr)
    ! Get solution to Sequential output array
    call VecGetArrayReadF90(SolAll,Sol_pointer,ierr)
    solGlobal = Sol_pointer
    call VecRestoreArrayReadF90(SolAll,Sol_pointer,ierr)
```

Listing 6.21: Portion of `solveLinearSystem` routine showing the solution scatter.

To avoid PETSc memory leaks, all PETSc objects created in `solveLinearSystem` are destroyed before exiting the routine scope, as seen in Listing 6.22.

```
    ! Destroy Solver Context
    call KSPDestroy(Solver,ierr)
    call MatDestroy(A,ierr)
    call VecDestroy(F,ierr)
    ! Destroy Solution and Scatter Context
    call VecDestroy(Sol,ierr)
    call VecDestroy(SolAll,ierr)
    call VecScatterDestroy(scatter_sol, ierr)
```

Listing 6.22: Portion of `solveLinearSystem` routine showing the destruction of PETSc objects.

Back to the context of the `mainGlobal`, the solution in `solGlobal` is passed on to the vectors responsible for storing the global variables increments, namely `NR%DeltaUhat`, `NR%DeltaRho`, and `NR%DeltaNuHat`, to continue the Newton-Raphson iteration.

Before exiting `mainGlobal`, PETSc objects created at the routine scope are destroyed, followed by a call to `PetsFinalize`. Additionally, the temporary variables `tmpL`, `tmpG`, and `geo` are deallocated, as shown in Listing 6.23.

```
    ! Destroy Solver Context
    call MatDestroy(A,ierr)
    call VecDestroy(F,ierr)
    call VecDestroy(resU,ierr)
    call VecDestroy(resNu,ierr)
    call VecDestroy(resRho,ierr)
    call PetscFinalize(ierr)
    deallocate(tmpL)
    deallocate(tmpG)
    deallocate(geo)
```

Listing 6.23: Portion of `mainGlobal` routine showing the final destruction of PETSc objects and deallocation of temporary variables.

6.3.9 Local Solver

Similarly to the global solver, the local solver routine `mainLocal` starts with a call to the preallocation routines `preAllocationLocal`, `preAlloationGlobal`, and preAllocationGeometry, generating three variables, namely, `tmpL`, `tmpG`, and `geo`. Again, on-the-fly computation of matrices, vectors, and geometrical features is employed. The loop on elements starts, and `updaMatricesLocal` is called. This routine has a similar structure as `updateMatrices`, differing in some additional matrices/vectors that are required for the local solution evaluation. Next local solutions for the current element are evaluated, namely `NR%L`, `NR%u`, `NR%q`, and `NR%Nu`. A call to the routine `setToZeroPreallocation` is performed before moving the next element in the `do` loop.

6.3.10 Post-process and output

After the solution stage is completed, a check to `inputData%computeQoI` is performed to check whether quantities of interest are to be computed or not. This is followed by a call to a check to `inputData%computeErr` to check if errors need to be computed. Next, a call to `saveSolution` is performed to save the solution field in `.dat` format, as well as QoI and error data, in case they were computed in the previous steps. Finally, a call to `exportToVTK` will save the solution field in `.vtk` format, allowing external postprocessing of the solution in ParaView software.

Chapter 7

Summary

This thesis presented the development of low-order face-based finite volume approaches, specifically the face-centred finite volume methods, for the computation of incompressible laminar and turbulent flow problems in a twofold manner. Firstly, the foundational aspects of the novel strategies, including all key components of the numerical formulation, were introduced. Secondly, the development of a CFD tool implemented in `Fortran 90` was presented as an initial framework to support the integration of face-centred finite volume methods within the CFD community.

The summary of the results obtained and contributions is outlined as follows:

1. **A face-centred finite volume formulation for the solution of incompressible laminar (NS) and turbulent (RANS) flows.**

   The face-centred finite volume formulation, first introduced in Sevilla et al. (2018) for second-order elliptic problems (i.e., Poisson and Stokes problems), was developed for the first time for the incompressible convection-dominated problems in the realm of laminar and turbulent flows. The face-centred finite volume formulation of the incompressible Reynolds-averaged Navier-Stokes (RANS) equations coupled with the negative Spalart-Allmaras (SA) turbulence model of Allmaras et al. (2012), was presented in this work and published in Vieira et al. (2024). A seamless particularisation of the formulation for the laminar cases, that is, Navier-Stokes equations, was also shown. For steady-state cases, two time-relaxation strategies were proposed to deal with the highly non-linearity of the discrete equations, especially for the RANS-SA case. For the RANS-SA cases. Also, *monolithic* and *staggered* approaches were devised and successfully implemented to solve the RANS-SA equations coupling. A comprehensive set of laminar and turbulent benchmark tests in both steady and unsteady regimes, involving different grid types, in 2-dimensional and 3-dimensional settings were performed. Attention was given to the capability of the FCFV method to preserve optimal rates of convergence, accuracy, and robustness of the non-linear solver convergence when highly stretched and distorted grids were employed. The *monolithic* and *staggered* approaches were compared under the time-relaxation scheme. Results showed that while the first is more stable, allowing a higher CFL evolution speed and fewer time steps for convergence, the second is less stable, requiring extra care in the CFL evolution law. This brings about a trade-off between the CFL evolution rate and the computational cost of each time step (i.e., size of the linear system of equations), which is lower for the *staggered* approach compared to the *monolithic* one. The performance and robustness of the proposed FCFV in handling general grid types, with no constraints in grid quality, without the requirement for pressure-velocity couple segregated algorithms, and without the need for gradient reconstruction provides ground for a strategy that can fill some of the gaps of the current CFD methodologies found in industry to deal with laminar and turbulent flows.

2. **A hybrid pressure face-centred finite volume formulation for the solution of incompressible Navier-Stokes equations.**

   The novel hybrid pressure face-centred finite volume (hybrid pressure FCFV) formulation for the solution of the incompressible Navier-Stokes equations was proposed and introduced. The proposed formulation had its performance evaluated and compared with the standard FCFV formulation, presented in this work and published in Vieira et al. (2024). To this end, a set of steady-state laminar benchmark tests involving different grid types in 2-dimensional and 3-dimensional settings were performed. The method consistency in terms of fulfilment of the compressibility condition in the limit of infinitely fine grids was verified. Numerical results at moderate Reynolds numbers have shown the capabilities of the hybrid pressure FCFV formulation in providing enhanced results compared to the standard FCFV formulation, especially in a 3-dimensional setting, due to the richer pressure space and artificial compressibility introduced. For a very low Reynolds number, close to the limit of a Stokes flow, both formulations showed similar performance. The computational cost of the hybrid pressure FCFV formulation was also compared with the standard FCFV, and a good trade-off between computational cost and accuracy was demonstrated for a 3-dimensional case.

3. **Riemann solvers for the convective stabilization.**

   Novel stabilisation tensors, following Vila-Pérez et al. (2021, 2022), for the convective part of the incompressible RANS, NS, and SA equations are proposed and successfully implemented in the FCFV formulations presented in this work. The three Riemann solvers were compared in terms of numerical diffusion, and the HLL Riemann solver proved to be consistently less diffusive than the Roe and LF Riemann solvers, as expected. The LF solver proved to be the most diffusive one but more stable in terms of non-linear solver convergence, especially when unsteady turbulent flows were concerned. This is because they provide extra numerical diffusion for the SA negative model and RANS equations.

4. **A `Fortran 90` FCFV implementation.**

   The standard FCFV and hybrid pressure FCFV formulations were implemented in a `Fortran 90` solver capable of dealing with incompressible laminar and turbulent flows in 2-dimensional and 3-dimensional settings. The solver handles hybrid grids with triangles and quadrilaterals in 2D and tetrahedrons, hexahedrons, prims, and pyramids in 3D. A comprehensive overview of the solver implementation and technicalities related to the non-linear solver, relaxation strategies, and linear solver algorithms was provided in this work. The presented `Fortran 90` FCFV solver provides a basis for the development of an engineering tool for fast, robust, and sufficiently accurate prediction of incompressible laminar and turbulent flows in a range of interests for most aerodynamics applications. With these features and flexibility concerning meshing, CFD aerodynamic problems in the aerospace and automotive fields can be tackled. Since the RANS model is still widely used in industry for the quantification of averaged quantities of interest, this CFD tool can be employed for fast decisions at the early design stage. Also, this computationally cheap CFD tool coupled with a reduced-order model technique can figure as a platform for real-time/overnight CFD solutions.

7.1 Future development and studies

Some gaps not addressed or explored in the current work offer opportunities for further investigation, either to improve the presented methodologies or to extend them to a broader range of problems within the CFD field. The proposed lines of investigation for future work are:

1. **Fully parallelization of the `Fortran 90` FCFV solver.**

   In future works, with the exploration of high-performance computing (HPC), such as MPI and Open-MP parallelisation paradigms, a competitive CFD tool in the face of current commercial CFD packages can be aimed at.

2. **Explore iterative linear solvers and Newton-Kyrlov methods.**

   The present work focused on aspects of the FCFV related to numerical stabilisation and non-linear solvers (e.g., linearisation, time relaxation strategies, and RANS-SA coupling splitting). The solution of the linear system of equations, arising from the linearisation of the discrete equations, was not explored, and primarily direct methods were employed given their robustness and fewer control parameters. For large-scale problems, however, direct linear solvers become prohibited due to the memory overhead related to the storage of the factorised matrices. Thus, the employment of iterative solvers would expand the limits of the FCFV solver. Further, the iterative linear solver can be combined with the Newton-Raphson solver in a Newton-Kyrlov method, as done in Crivellini and Bassi (2011); Crivellini et al. (2013), where matrix-free approaches are explored for enhanced solver efficiency.

3. **Extension of the FCFV solver to other RANS turbulence models.**

   Since the RANS-SA equations are mostly suited for external aerodynamics computations, the implementation of other RANS turbulence models, such as the two-equations ones, e.g., $\kappa-\epsilon$ and $\kappa-\omega$, can extend the range of application for the newly introduced FCFV-RANS CFD solver.

4. **Benchmark the FCFV solver against commercial FV codes.**

   While this work provided an extensive set of benchmarks comparing the FCFV solution against various numerical references using different numerical methods and commercial FV codes, it is also desirable to conduct a more comprehensive and complete benchmark of the FCFV solver for laminar and turbulent flows against the most prominent second-order FV solvers in the industry. Such a study has already been performed by Vila-Pérez et al. (2023), where the FCFV method for compressible flows was benchmarked against the Ansys Fluent commercial CFD FV-based solver, demonstrating the superior performance of the FCFV solver, particularly on highly distorted and stretched grids.

5. **Extend the FCFV solver to compressible RANS equations.**

   The FCFV method has already been introduced for the solution of the compressible flows in Vila-Pérez et al. (2022), where the Riemann solvers in the FCFV context are introduced and demonstrate non-oscillatory capturing of sharp oscillations. Also, the same methodology was benchmarked in Vila-Pérez et al. (2023) showing robust performance in the face of low-quality grids in contrast to CFD commercial code. The mentioned findings encourage the extension of the FCFV methodology for the solution of the compressible RANS equations coupled with a suitable turbulence model in future works.

6. **Extend the hybrid pressure FCFV for unsteady and turbulent flows.**

   In the present study, the hybrid pressure FCFV formulation was first introduced and benchmarked for the solution of the steady incompressible flows, modelled by the Navier-Stokes equations. The extension for unsteady flows is a straightforward one since no difference in the time-dependent terms in the local problem is found between the hybrid pressure and standard FCFV formulations. On the other hand, the extension of this formulation for the solution of the RANS equations (i.e., turbulent flows) is of interest given the encouraging enhanced results obtained at a moderate Reynolds number for laminar flows compared to the standard FCFV formulation.

7. **Extend the second-order FCFV to the incompressible laminar and turbulent flows.**

   The FCFV formulation developed in this work for incompressible laminar and turbulent flows builds on the ideas presented in Sevilla et al. (2018, 2019), where first-order convergence is achieved for all flow variables and their gradients. A second-order accurate FCFV formulation was proposed in Vieira et al. (2020) for second-order elliptic problems, such as Poisson and Stokes flows. In this formulation, second-order convergence was attained for the velocity field, while first-order convergence was achieved for other flow variables and their gradients.

   The second-order FCFV formulation for the incompressible Navier-Stokes equations was initially tested in this work, but no definitive conclusions were drawn regarding its ability to deliver second-order convergence for the velocity field in convection-dominated problems. Further studies are needed on convective numerical stabilisation, which was initially inspired by the works of Giacomini et al. (2020). The results obtained in this work suggest that using Riemann solvers for convective stabilisation could enhance the second-order FCFV formulation for convection-dominated flows.

Appendix A

Riemann solvers and convective stabilization

This appendix presents three new convective stabilization tensors for the incompressible RANS-SA and Navier-Stokes equations, used in the formulations presented in Chap. 2 and Chap. 3 of this work. The convective stabilizations $\boldsymbol{\tau}^a$ and $\tilde{\tau}^a$ are used to define the full stabilization tensor $\boldsymbol{\tau}$ and scalar $\tilde{\tau}$ in the local and global problems given by equations (2.33) and (2.34), respectively, for the standard FCFV formulation of RANS-SA equations. The convective stabilization $\boldsymbol{\tau}^a$ is also used to define the tensor $\boldsymbol{\tau}$ for laminar cases, as in equations (2.53) and (2.54), for the standard FCFV formulation and in equations (3.30) and (3.31), for the hybrid pressure FCFV formulation. They are employed and compared in several examples presented in Chap. 4 and Chap. 5.

The rationale is inspired by classical Lax-Friedrichs, Roe, and Harten-Lax-van Leer Riemann solvers for compressible flows Toro (2009) and it is derived following the unified approach presented in Vila-Pérez et al. (2021, 2022).

The convective part of the RANS momentum equation can be written as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \mathbf{f}_k(\boldsymbol{u})}{\partial x_k} = \mathbf{0}, \tag{A.1}$$

where $\mathbf{f}_k(\boldsymbol{u}) = u_k \boldsymbol{u}$, or in quasi-linear form as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \mathbf{A}_k \frac{\partial \boldsymbol{u}}{\partial x_k} = \mathbf{0}, \tag{A.2}$$

where $\mathbf{A}_k = \partial \mathbf{f}_k / \partial \boldsymbol{u}$.

To derive the stabilisation tensors $\boldsymbol{\tau}^a$, the system of equations is restricted to an arbitrary direction $\boldsymbol{n}$, which in the context of the FCFV will correspond to the normal direction to an edge/face of a cell. The resulting system is

$$\frac{\partial \boldsymbol{u}}{\partial t} + \mathbf{A}_{\boldsymbol{n}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} = \mathbf{0}, \tag{A.3}$$

with $\mathbf{A}_{\boldsymbol{n}} = (\boldsymbol{u} \cdot \boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}} + \boldsymbol{u} \otimes \boldsymbol{n}$.

The spectral decomposition of $\mathbf{A}_{\boldsymbol{n}}$ leads to $\mathbf{A}_{\boldsymbol{n}} = \mathbf{R}\,\boldsymbol{\Lambda}\,\mathbf{R}^{-1}$, where $\mathbf{R}$ is the matrix whose columns are the right eigenvectors of $\mathbf{A}_{\boldsymbol{n}}$ and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_{\mathtt{n_{sd}}})$ with $\lambda_1 = \cdots = \lambda_{\mathtt{n_{sd}}-1} = v_{\boldsymbol{n}}$, $\lambda_{\mathtt{n_{sd}}} = 2v_{\boldsymbol{n}}$, and $v_{\boldsymbol{n}} = \boldsymbol{u} \cdot \boldsymbol{n}$. To ensure a minimum value of the stabilisation, the regularised eigenvalues are considered, namely $\hat{\lambda}_1 = \cdots = \hat{\lambda}_{\mathtt{n_{sd}}-1} = \max\left(|\boldsymbol{u} \cdot \boldsymbol{n}|, \varepsilon\right)$, $\hat{\lambda}_{\mathtt{n_{sd}}} = \max\left(2|\boldsymbol{u} \cdot \boldsymbol{n}|, \varepsilon\right)$, where $\varepsilon > 0$ is a real, user-defined numerical parameter.

Using the maximum eigenvalue of $\hat{\boldsymbol{\Lambda}} = \operatorname{diag}\left(\hat{\lambda}_1, \dots, \hat{\lambda}_{\mathtt{n_{sd}}}\right)$, the stabilisation inspired by the Lax-Friedrichs (LF) Riemman solver is defined as

$$\boldsymbol{\tau}^a_{\mathtt{LF}} = \max\left\{2|\hat{\boldsymbol{u}} \cdot \boldsymbol{n}|, \varepsilon\right\} \mathbf{I}_{\mathtt{n_{sd}}}. \tag{A.4}$$

The stabilisation inspired by the Roe Riemman solver is defined as

$$\boldsymbol{\tau}^a_{\mathtt{Roe}} = \mathbf{R}\,\hat{\boldsymbol{\Lambda}}\,\mathbf{R}^{-1}. \tag{A.5}$$

From an implementation point of view, it is convenient to rewrite the Roe stabilisation as

$$\boldsymbol{\tau}^a_{\mathtt{Roe}} = \begin{cases} \varepsilon\,\mathbf{I}_{\mathtt{n_{sd}}} & \text{for } |\hat{\boldsymbol{u}} \cdot \boldsymbol{n}| \leq \varepsilon/2, \\ \varepsilon\,\mathbf{I}_{\mathtt{n_{sd}}} + \operatorname{sign}(\hat{\boldsymbol{u}}\cdot\boldsymbol{n})\left(2 - (\varepsilon/|\hat{\boldsymbol{u}}\cdot\boldsymbol{n}|)\right)(\hat{\boldsymbol{u}} \otimes \boldsymbol{n}) & \text{for } |\hat{\boldsymbol{u}} \cdot \boldsymbol{n}| \in \left(\varepsilon/2, \varepsilon\right), \\ \operatorname{sign}(\hat{\boldsymbol{u}}\cdot\boldsymbol{n})\left[(\hat{\boldsymbol{u}}\cdot\boldsymbol{n})\mathbf{I}_{\mathtt{n_{sd}}} + (\hat{\boldsymbol{u}} \otimes \boldsymbol{n})\right] & \text{for } |\hat{\boldsymbol{u}} \cdot \boldsymbol{n}| \geq \varepsilon, \end{cases} \tag{A.6}$$

The last stabilisation considered is inspired by the Harten-Lax-van Leer (HLL) Riemann solver. This stabilisation, defined as

$$\boldsymbol{\tau}^a_{\mathtt{HLL}} = \max\{2(\hat{\boldsymbol{u}} \cdot \boldsymbol{n}), \varepsilon\}\mathbf{I}_{\mathtt{n_{sd}}}, \tag{A.7}$$

employs an estimate of the largest wave speed of the Riemann problem.

For the SA turbulence model, the derivation of the convective stabilisation $\tilde{\tau}^a$ is simpler due to the scalar nature of the equation. The stabilisation inspired by both LF and Roe Riemann solvers is simply

$$\tilde{\tau}^a_{\mathtt{LF}} = \tilde{\tau}^a_{\mathtt{Roe}} = \max\left\{|\hat{\boldsymbol{u}} \cdot \boldsymbol{n}|, \tilde{\varepsilon}\right\}. \tag{A.8}$$

whereas the stabilisation inspired by the HLL Riemann solver is

$$\tilde{\tau}^a_{\mathtt{HLL}} = \max\left\{\hat{\boldsymbol{u}} \cdot \boldsymbol{n}, \tilde{\varepsilon}\right\}. \tag{A.9}$$

It can be observed that both the Lax-Friedrichs and the Roe stabilisations provide a unique value of the stabilisation as seen from the two cells sharing an edge/face. In contrast, the HLL stabilisation provides different values for the two cells sharing an edge/face.

The Roe Riemann solver of the RANS equations, given in equation (A.6), provides a continuous expression when $|\hat{\boldsymbol{u}} \cdot \boldsymbol{n}|$ tends to ε and $\varepsilon/2$, which numerical experiments have confirmed to be a key factor to guarantee convergence of the numerical scheme.

In all the examples, the numerical parameter ε is selected as 5×10^{-2} for the LF and HLL stabilisations of RANS equations, whereas a slightly higher value of 10^{-1} is used for the Roe stabilisation. For the SA equation, the value of $\tilde{\varepsilon}$ is taken as 10^{-2} for all stabilisations.

Appendix B

Relaxation strategies

When a steady-state simulation is of interest, the solution of a pseudo-transient problem, where t denotes the artificial or pseudo-time, is often preferred due to the difficulty of finding an initial guess that guarantees convergence of the Newton-Raphson algorithm. In this scenario, time accuracy is not a requirement and, therefore, it is common to utilize a first-order time marching scheme to reach the steady state. Furthermore, strategies to vary the time step as the solution evolves are attractive because, at the beginning of a simulation, smaller time steps are required to guarantee convergence of the nonlinear problem, whereas as the solution approaches a steady state it is possible to use extremely large time steps.

When employed for steady-state RANS-SA problems, pure Newton-Raphson solvers are often unstable. This is owned to the high non-linearity of the coupled RANS-SA system of equations and the difficulty in finding an initial guess that is close enough to the basin of attraction of the non-linear system solution. As a remedy, pseudo time-marching, or time relaxation, strategies are often employed for the solution of steady-state RANS-SA coupled problems, as done in Crivellini et al. (2013); Crivellini and Bassi (2011); Lodares et al. (2022). The idea is to accelerate the evolution to steady-state by increasing the time-step Δt as the solution evolves, this leads to a pure second-order Newton-Raphson solver as $\Delta t \to \infty$.

B.1 Switched Evolution Relaxation Strategy (SER)

When time marches to a steady state, this work employs the low-order BDF1 scheme. Several strategies have been proposed to automatically adjust the time step during steady state simulations, e.g. a simple exponential law Kelley and Keyes (1998); Bucker et al. (2009). More elaborated approaches such as the so-called switched evolution relaxation (SER) Mulder and Van Leer (1985); Bucker et al. (2009); Kelley and Keyes (1998) approach use the ratio between the nonlinear residuals at two consecutive time steps to adjust the time step. This work adapts the SER approach to the proposed FCFV scheme.

At the beginning of a simulation, the CFL is initialized to a low value, namely $\text{CFL}^0 = 0.1$. This is particularly important to avoid the potential negative impact of performing an impulsive start with an initial condition not satisfying the boundary conditions, e.g. using free-stream values for the initial condition. Following Crivellini et al. (2013); Crivellini and Bassi (2011); Kelley and Keyes (1998); Ceze and Fidkowski (2013); Economon (2019), after the first time step the CFL is updated according to the expression

$$\text{CFL}^{n+1} = \min\left\{\text{CFL}^n/f^\gamma, \text{CFL}_{\text{max}}\right\}, \tag{B.1}$$

where

$$f = \max\left\{ \frac{\left\|\mathbf{R}^n_{u,s}\right\|_\infty}{\left\|\mathbf{R}^{n-1}_{u,s}\right\|_\infty}, \frac{\left\|\mathbf{R}^n_{\tilde{\nu},s}\right\|_\infty}{\left\|\mathbf{R}^{n-1}_{\tilde{\nu},s}\right\|_\infty} \right\} \tag{B.2}$$

and the exponent γ is taken as $\gamma_{\text{max}} > 1$ if both residuals decrease with respect to the previous time step, i.e., if $f \leq 1$, or as $\gamma_{\text{min}} < 1$ if at least one of the residuals increases, i.e., if $f > 1$. In equation (B.2) the residual vector $\mathbf{R}_{u,s}$ and $\mathbf{R}_{\tilde{\nu},s}$ correspond to the residuals defined in (B.8). It is worth noting that only two residuals of the local problem (2.33) are employed to define f in equation (B.2) for the reason explained in Section B.3.

Following Crivellini et al. (2013); Crivellini and Bassi (2011), the maximum allowed CFL is taken as $\text{CFL}_{\text{max}} = 10^{20}$ in all the numerical examples. When employing the monolithic FCFV approach, the parameters to define the exponent γ are taken as $\gamma_{\text{max}} = 2$ and $\gamma_{\text{min}} = 0.1$, following Economon (2019). When using the staggered FCFV approach, numerical experiments show that it is necessary to employ a lower value for γ_{max}, which is taken as $\gamma_{\text{max}} = 1.2$.

B.2 Newton-Raphson-based strategy

In this work, we introduce a more conservative approach at which the time-step size Δt at time $n+1$ is adapted based on the number of Newton-Raphson iterations N^n required for the convergence at time n. Note that, even though we are not interested in the time-dependent solution of the problem, for robustness, we still make sure the solution converges within each pseudo-time step. The time-step Δt evolves with the CFL number, which follows the following rule

$$\text{CFL}^{n+1} = \begin{cases} \text{CFL}_{\text{max}} & \text{if } \text{N}^n \leq \text{N}_{\text{low}}, \\ \text{CFL}_{\text{min}} & \text{if } \text{N}^n \geq \text{N}_{\text{high}}, \\ f(\text{N}^n) & \text{otherwise}, \end{cases} \tag{B.3}$$

The input parameters CFL_{min}, CFL_{max} are the minimum initial CFL and the maximum allowed CFL. While N_{low} and N_{high} are estimations for the number of Newton Raphson solves required for convergence when the solution is close and far away from the basin of attraction respectively. The function $f(\mathrm{N}^n)$ is defined as

$$f(N^n) = \frac{\mathrm{CFL_{max}} - \mathrm{CFL_{min}}}{\mathrm{N_{low}} - \mathrm{N_{high}}} (\mathrm{N}^n - \mathrm{N_{low}}) + \mathrm{CFL_{max}} \tag{B.4}$$

In order to avoid abrupt CFL^{n+1} variations we introduce the following limiter

$$\mathrm{CFL}^{n+1} = \max\left(\min\left(\mathrm{CFL}^{n+1}, \mathrm{r_{max}}\mathrm{CFL}^n\right), \mathrm{r_{min}}\mathrm{CFL}^n\right) \tag{B.5}$$

Here, $\mathrm{r_{min}}$ and $\mathrm{r_{max}}$ are the maximum allowed decrement and increment ratios respectively. Note that this limiter will render a geometric progression increment of the CFL number between successive time steps, which is comparable to the exponential law, as in Kelley and Keyes (1998); Bucker et al. (2009). At the beginning of a simulation, the CFL is initialized to a low value, namely $\mathrm{CFL}^0 = 0.1$. The values adopted for the parameters are $\mathrm{CFL_{min}} = 10^0$, $\mathrm{CFL_{max}} = 10^{20}$,$\mathrm{N_{low}} = 2$, $\mathrm{N_{high}} = 9$, $\mathrm{r_{max}} = 2.0$ and $\mathrm{r_{min}} = 0.05$. The convergence of each Newton-Raphson solver is measured by (B.6), while the convergence to the steady-state solution is measured with (B.8). Note that within each time step, the Newton-Raphson stop criteria will be based on the user input values for the residual tolerance and the maximum allowed number of Newton solves.

B.3 Newton-Raphson and time convergence criteria

To check the convergence of the Newton-Raphson solver, the following residual is employed for the RANS-SA case

$$\|\mathcal{R}^n\| := \max\left\{\frac{\|\mathbf{R}^n_{\hat{u}}\|_\infty}{\|\mathbf{f}^n_u\|_\infty}, \frac{\|\mathbf{R}^n_{\tilde{\nu}}\|_\infty}{\|\mathbf{f}^n_{\tilde{\nu}}\|_\infty}\right\} \tag{B.6}$$

which corresponds to the result of assembling the elemental contributions of the second and fourth residual in equation (2.34), respectively. Where the normalizing factors are defined as

$$\begin{aligned} \mathbf{f}^n_u &:= |\Omega_e|\mathbf{s}^n_e + \sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|\Big(\boldsymbol{\tau}^{n-1}_j - (\boldsymbol{u}^n_{D,j}\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big)\boldsymbol{u}^n_{D,j} \\ \mathbf{f}^n_{\tilde{\nu}} &:= \sum_{j\in\mathcal{D}_e} |\Gamma_{e,j}|(\tilde{\tau}^{n-1}_j - \boldsymbol{u}^n_{D,j}\cdot\boldsymbol{n}_j)\tilde{\nu}^n_{D,j} \end{aligned} \tag{B.7}$$

In the cases where a pseudo-time step is employed, to measure the convergence to the steady-state solution we use the steady-state version of the local residuals, that is

$$\|\mathcal{R}^n_s\| := \max\left\{\frac{\left\|\mathbf{R}^n_{u,s}\right\|_\infty}{\|\mathbf{f}^n_u\|_\infty}, \frac{\left\|\mathbf{R}^n_{\tilde{\nu},s}\right\|_\infty}{\|\mathbf{f}^n_{\tilde{\nu}}\|_\infty}\right\} \tag{B.8}$$

with the normalising factors defined in (B.7). The residual vector $\mathbf{R}_{u,s}$ and $\mathbf{R}_{\tilde{\nu},s}$ correspond to the result of assembling the elemental contributions of the second and fourth residual in equation (2.33), respectively, but without including the term corresponding to the discrete-time derivative. It is worth noting that only two residuals of the local problem (2.33) are employed to measure convergence in time. This is because the other two local residuals are linear. Furthermore, those are the only residuals having time derivative terms, and for the same reason, the global residuals are not employed to measure convergence to steady-state. Note that, for laminar cases, only the residuals related to $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ are employed to measure Newton-Raphson and time convergence.

Appendix C

Linearisation of the FCFV discrete forms

This appendix showcases the derivation of the linearised forms of the residuals arising from the FCFV strategies presented in this work. Section Section C.1 shows the linearisation of the residuals arising from the local and global problems of the RANS-SA FCFV formulation, derived in Chap. 2. In Section C.2 the linearisation of the residuals of the global problem, arising from Navier-Stokes FCFV formulation described in Section 2.5, is presented. Finally, section C.3 describes the linearization of the global residuals arising from the Navier-Stokes hybrid pressure FCFV formulation discussed in Chap. 3.

C.1 FCFV for the incompressible RANS-SA equations

For the linearisation of the residuals related to the RANS-SA FCFV formulation, described in Chap. 2, given by equations (2.33) and (2.34), we recall the linear problem to be solved at each time step n and Newton-Rpahson solve m for the increments of the local and global variables,

$$\begin{bmatrix} \mathbf{T}_{UU} & \mathbf{T}_{U\Lambda} \\ \mathbf{T}_{\Lambda U} & \mathbf{T}_{\Lambda\Lambda} \end{bmatrix}^{n,m} \begin{Bmatrix} \Delta\mathbf{U} \\ \Delta\boldsymbol{\Lambda} \end{Bmatrix}^{n,m} = - \begin{Bmatrix} \mathbf{R}_U \\ \mathbf{R}_\Lambda \end{Bmatrix}^{n,m}, \tag{C.1}$$

where

$$\mathbf{T}_{UU} = \begin{bmatrix} \mathbf{T}_{LL} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{uL} & \mathbf{T}_{uu} & \mathbf{T}_{u\tilde{q}} & \mathbf{T}_{u\tilde{\nu}} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_{\tilde{q}\tilde{q}} & \mathbf{0} \\ \mathbf{T}_{\tilde{\nu}L} & \mathbf{0} & \mathbf{T}_{\tilde{\nu}\tilde{q}} & \mathbf{T}_{\tilde{\nu}\tilde{\nu}} \end{bmatrix}, \quad \mathbf{T}_{U\Lambda} = \begin{bmatrix} \mathbf{T}_{L\hat{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{u\hat{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_{\tilde{q}\hat{\nu}} \\ \mathbf{T}_{\tilde{\nu}\hat{u}} & \mathbf{0} & \mathbf{T}_{\tilde{\nu}\hat{\nu}} \end{bmatrix}, \tag{C.2a}$$

$$\mathbf{T}_{\Lambda U} = \begin{bmatrix} \mathbf{T}_{\hat{u}L} & \mathbf{T}_{\hat{u}u} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_{\hat{\nu}\tilde{q}} & \mathbf{T}_{\hat{\nu}\tilde{\nu}} \end{bmatrix}, \quad \mathbf{T}_{\Lambda\Lambda} = \begin{bmatrix} \mathbf{T}_{\hat{u}\hat{u}} & \mathbf{T}_{\hat{u}\rho} & \mathbf{T}_{\hat{u}\hat{\nu}} \\ \mathbf{T}_{\rho\hat{u}} & 0 & 0 \\ \mathbf{T}_{\hat{\nu}\hat{u}} & 0 & \mathbf{T}_{\hat{\nu}\hat{\nu}} \end{bmatrix},$$

$$\mathbf{U} = \begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \\ \tilde{\mathbf{q}} \\ \tilde{\boldsymbol{\nu}} \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\rho} \\ \hat{\boldsymbol{\nu}} \end{Bmatrix}, \quad \mathbf{R}_U = \begin{Bmatrix} \mathbf{R}_L \\ \mathbf{R}_u \\ \mathbf{R}_{\tilde{q}} \\ \mathbf{R}_{\tilde{\nu}} \end{Bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_\rho \\ \mathbf{R}_{\hat{\nu}} \end{Bmatrix}, \tag{C.2b}$$

with $\mathbf{T}_{ab}$ denoting the tangent matrix obtained by assembling the cell contributions $(\mathbf{T}_{ab})_e := \partial\mathbf{R}_{e,a}/\partial b$ and $\Delta\circledcirc^{n,m} = \circledcirc^{n,m+1} - \circledcirc^{n,m}$ the solution increment from the Newton-Raphson iteration m to $m+1$.

The jacobian block $\mathbf{T}_{UU}$, given by the sub-blocks in Eq. (C.2), is computed on each cell e as follows,

$$\begin{cases} (\mathbf{T}_{LL})^{e,n} := |\Omega_e|\mathbf{I}_{\mathtt{n_{sd}}} \\ (\mathbf{T}_{uL})^{e,n} := -\dfrac{|\Omega_e|}{Re}\mathbf{I}_{\mathtt{n_{sd}}} \otimes \mathbf{q}_{t,e}(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n) - \dfrac{|\Omega_e|}{Re}\mathbf{q}_{t,e}(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n) \otimes \mathbf{I}_{\mathtt{n_{sd}}} \\ (\mathbf{T}_{uu})^{e,n} := |\Omega_e| a_0 \mathbf{I}_{\mathtt{n_{sd}}} + \displaystyle\sum_{j\in\mathcal{A}_e} |\Gamma_{e,j}|\boldsymbol{\tau}_j^{n-1} \\ (\mathbf{T}_{u\tilde{q}})^{e,n} := -\dfrac{|\Omega_e|}{Re}(\mathbf{L}_e^n + [\mathbf{L}_e^n]^T)\dfrac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n)}{\partial\tilde{\mathbf{q}}_e^n} \\ (\mathbf{T}_{u\tilde{\nu}})^{e,n} := -\dfrac{|\Omega_e|}{Re}(\mathbf{L}_e^n + [\mathbf{L}_e^n]^T)\dfrac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n} \\ (\mathbf{T}_{\tilde{q}\tilde{q}})^{e,n} := |\Omega_e| \\ (\mathbf{T}_{\tilde{\nu}L})^{e,n} := -|\Omega_e|\dfrac{\partial \mathrm{s}_e(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n, \mathbf{L}_e^n)}{\partial\mathbf{L}_e^n} \\ (\mathbf{T}_{\tilde{\nu}\tilde{q}})^{e,n} := -2\dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}_e^n f_{n,e}(\tilde{\nu}_e^n) - 2\dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}_e^n \tilde{\nu}_e^n \dfrac{\partial f_{n,e}(\tilde{\nu}_e^n)}{\partial\tilde{\nu}_e^n} - |\Omega_e|\dfrac{\partial \mathrm{s}_e(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n, \mathbf{L}_e^n)}{\partial\tilde{\mathbf{q}}_e^n} \\ (\mathbf{T}_{\tilde{\nu}\tilde{\nu}})^{e,n} := |\Omega_e| a_0 + \displaystyle\sum_{j\in\mathcal{A}_e} |\Gamma_{e,j}|\tilde{\tau}_j^{n-1} - 2\dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}_e^n \cdot \tilde{\mathbf{q}}_e^n \dfrac{\partial f_{n,e}(\tilde{\nu}_e^n)}{\partial\tilde{\nu}_e^n} \\ \qquad - \dfrac{|\Omega_e|}{\sigma Re}\tilde{\mathbf{q}}_e^n \cdot \tilde{\mathbf{q}}_e^n \tilde{\nu}_e^n \dfrac{\partial^2 f_{n,e}(\tilde{\nu}_e^n)}{\partial(\tilde{\nu}_e^n)^2} - |\Omega_e|\dfrac{\partial \mathrm{s}_e(\tilde{\nu}_e^n, \tilde{\mathbf{q}}_e^n, \mathbf{L}_e^n)}{\partial\tilde{\nu}_e^n} \end{cases} \tag{C.3}$$

with the partial derivative terms of (C.3) being derived in Section C.1.1. Similarly, the block $\mathbf{T}_{U\Lambda}$ computation is given by the assembly of the cell e contribution on each $j \in \mathcal{B}_e$ face as

$$\begin{cases} (\mathbf{T}_{L\hat{u}})_j^{e,n} := |\Gamma_{e,j}|\Big(\mathbf{I}_{\mathtt{n_{sd}}} \otimes \boldsymbol{n}_j\Big)^T \\ (\mathbf{T}_{u\hat{u}})_j^{e,n} := -|\Gamma_{e,j}|\boldsymbol{\tau}_j^{n-1} + |\Gamma_{e,j}|(\hat{\mathbf{u}}_j^n \cdot \boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}} + |\Gamma_{e,j}|(\hat{\mathbf{u}}_j^n \otimes \boldsymbol{n}_j) \\ (\mathbf{T}_{\tilde{q}\hat{\nu}})_j^{e,n} := |\Gamma_{e,j}|\boldsymbol{n}_j \\ (\mathbf{T}_{\tilde{\nu}\hat{u}})_j^{e,n} := |\Gamma_{e,j}|\boldsymbol{n}_j\hat{\nu}_j^n \\ (\mathbf{T}_{\tilde{\nu}\hat{\nu}})_j^{e,n} := -|\Gamma_{e,j}|\tilde{\tau}_j^{n-1} + |\Gamma_{e,j}|(\hat{\mathbf{u}}_j^n \cdot \boldsymbol{n}_j) \end{cases} \tag{C.4}$$

The block $\mathbf{T}_{\Lambda U}$ computation is given by the assembly of the cell e contribution on each $i \in \mathcal{B}_e$ face as

$$\begin{cases} (\mathbf{T}_{\hat{u}L})_i^{e,n} := |\Gamma_{e,i}|\dfrac{1+\hat{\nu}_i^n f_{v1}(\hat{\nu}_i^n)}{Re}\Big(\mathbf{I}_{\mathtt{n_{sd}}} \otimes \boldsymbol{n}_i\Big) \\ (\mathbf{T}_{\hat{u}u})_i^{e,n} := |\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1} \\ (\mathbf{T}_{\hat{\nu}\tilde{q}})_i^{e,n} := |\Gamma_{e,i}|\dfrac{1+\hat{\nu}_i^n f_n(\hat{\nu}_i^n)}{\sigma Re}\boldsymbol{n}_i \\ (\mathbf{T}_{\hat{\nu}\tilde{\nu}})_i^{e,n} := |\Gamma_{e,i}|\tilde{\tau}_i^{n-1} \end{cases} \tag{C.5}$$

Finally, the block $\mathbf{T}_{\Lambda\Lambda}$ is given by the assembly of the cell e contribution on each $i, j \in \mathcal{B}_e$ faces as

$$\begin{cases} (\mathbf{T}_{\hat{u}\hat{u}})_{i,j}^{e,n} := -|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}\delta_{ij} + |\Gamma_{e,i}|(\hat{\mathbf{u}}_i^n \cdot \boldsymbol{n}_i)\mathbf{I}_{\mathtt{n_{sd}}}\chi_{\mathcal{N}_e}(i)\delta_{ij} + |\Gamma_{e,i}|(\hat{\mathbf{u}}_i^n \otimes \boldsymbol{n}_i)\,\chi_{\mathcal{N}_e}(i)\delta_{ij} \\ (\mathbf{T}_{\hat{u}\rho})_i^{e,n} := |\Gamma_{e,i}|\boldsymbol{n}_i \\ (\mathbf{T}_{\hat{u}\hat{\nu}})_{i,j}^{e,n} := |\Gamma_{e,i}|\dfrac{1}{Re}\mathbf{L}_e^n\boldsymbol{n}_i\dfrac{\partial(\hat{\nu}_i^n f_{v1}(\hat{\nu}_i^n))}{\partial\hat{\nu}_i^n}\delta_{ij} \\ (\mathbf{T}_{\rho\hat{u}})_j^{e,n} := |\Gamma_{e,j}|\boldsymbol{n}_j^T \\ (\mathbf{T}_{\hat{\nu}\hat{u}})_{i,j}^{e,n} := |\Gamma_{e,i}|\hat{\nu}_i^n\,\chi_{\mathcal{N}_e}(i)\ \delta_{ij} \\ (\mathbf{T}_{\hat{\nu}\hat{\nu}})_{i,j}^{e,n} := -|\Gamma_{e,i}|\tilde{\tau}_i^{n-1}\delta_{ij} +\ |\Gamma_{e,i}|(\hat{\mathbf{u}}_i^n \cdot \boldsymbol{n}_i)\,\chi_{\mathcal{N}_e}(i)\delta_{ij} + |\Gamma_{e,i}|\dfrac{1}{\sigma Re}\ \tilde{\mathbf{q}}_e^n \cdot \boldsymbol{n}_i\dfrac{\partial(\hat{\nu}_i^n f_n(\hat{\nu}_i^n))}{\partial\hat{\nu}_i^n}\delta_{ij} \end{cases} \tag{C.6}$$

with the partial derivative terms of (C.6) being derived in Section C.1.1. After the reduction of the linearised problem by eliminating the local degrees of freedom, we obtain the global system of equations of (C.7), being composed of the following blocks

$$\mathbb{K}^{n,m}\Delta\boldsymbol{\Lambda}^{n,m} = \mathbb{F}^{n,m}, \tag{C.7}$$

$$\mathbb{K} = \begin{bmatrix} \mathbf{K}_{\hat{u}\hat{u}} & \mathbf{K}_{\hat{u}\rho} & \mathbf{K}_{\hat{u}\hat{\nu}} \\ \mathbf{K}_{\rho\hat{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{\hat{\nu}\hat{u}} & \mathbf{0} & \mathbf{K}_{\hat{\nu}\hat{\nu}} \end{bmatrix}, \mathbb{F} = \begin{Bmatrix} \mathbf{F}_{\hat{u}} \\ \mathbf{F}_{\rho} \\ \mathbf{F}_{\hat{\nu}} \end{Bmatrix}, \tag{C.8}$$

The blocks composing the left-hand side of the linear system of equations of (C.8), arising from the elimination of the local degrees of freedom, are implicitly given by the assembly of the cell contribution of

$$
\begin{aligned}
\mathbf{K}^e_{\hat{u}\hat{u}} &:= \mathbf{T}^e_{\hat{u}\hat{u}} - \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\hat{u}} - \mathbf{T}^e_{\hat{u}L}[\mathbf{T}^e_{LL}]^{-1}\mathbf{T}^e_{L\hat{u}} + \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{uL}[\mathbf{T}^e_{LL}]^{-1}\mathbf{T}^e_{L\hat{u}} \\
&\quad + \mathbf{T}_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\hat{u}} - \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}L}[\mathbf{T}^e_{LL}]^{-1}\mathbf{T}^e_{L\hat{u}} \\
\mathbf{K}^e_{\hat{u}\rho} &:= \mathbf{T}^e_{\hat{u}\rho} \\
\mathbf{K}^e_{\hat{u}\hat{\nu}} &:= \mathbf{T}^e_{\hat{u}\hat{\nu}} + \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{T}^e_{\tilde{q}\hat{\nu}} + \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\hat{\nu}} \\
&\quad - \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{T}^e_{\tilde{q}\hat{\nu}} \\
\mathbf{K}^e_{\rho\hat{u}} &:= \mathbf{T}^e_{\rho\hat{u}} \\
\mathbf{K}^e_{\hat{\nu}\hat{u}} &:= \mathbf{T}^e_{\hat{\nu}\hat{u}} - \mathbf{T}^e_{\hat{\nu}\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\hat{u}} + \mathbf{T}^e_{\hat{\nu}\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}L}[\mathbf{T}^e_{LL}]^{-1}\mathbf{T}^e_{L\hat{u}} \\
\mathbf{K}^e_{\hat{\nu}\hat{\nu}} &:= \mathbf{T}^e_{\hat{\nu}\hat{\nu}} - \mathbf{T}^e_{\hat{\nu}\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\hat{\nu}} - \mathbf{T}^e_{\hat{\nu}\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{T}^e_{\tilde{q}\hat{\nu}} + \mathbf{T}^e_{\hat{\nu}\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{T}^e_{\tilde{q}\hat{\nu}}
\end{aligned}
\tag{C.9}
$$

While the right-hand side blocks are implicitly given by the assembly of the cell contribution of

$$
\begin{aligned}
\mathbf{F}^e_{\hat{u}} &:= -\mathbf{R}^e_{\hat{u}} + \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{R}^e_u + \mathbf{T}^e_{\hat{u}L}[\mathbf{T}^e_{LL}]^{-1}\mathbf{R}^e_L - \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{J}^e_{uL}[\mathbf{T}^e_{LL}]^{-1}\mathbf{R}^e_L \\
&\quad - \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{R}^e_{\tilde{q}} - \mathbf{T}_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{R}^e_{\tilde{\nu}} \\
&\quad + \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}L}[\mathbf{T}^e_{LL}]^{-1}\mathbf{R}^e_L \\
&\quad + \mathbf{T}^e_{\hat{u}u}[\mathbf{T}^e_{uu}]^{-1}\mathbf{T}^e_{u\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{R}^e_{\tilde{q}} \\
\mathbf{F}^e_{\rho} &:= -\mathbf{R}^e_{\rho} \\
\mathbf{F}^e_{\hat{\nu}} &:= -\mathbf{R}^e_{\hat{\nu}} + \mathbf{T}^e_{\hat{\nu}\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{R}^e_{\tilde{\nu}} - \mathbf{T}^e_{\hat{\nu}\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}L}[\mathbf{T}^e_{LL}]^{-1}\mathbf{R}^e_L + \mathbf{T}^e_{\hat{\nu}\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{R}^e_{\tilde{q}} \\
&\quad - \mathbf{T}^e_{\hat{\nu}\tilde{\nu}}[\mathbf{T}^e_{\tilde{\nu}\tilde{\nu}}]^{-1}\mathbf{T}^e_{\tilde{\nu}\tilde{q}}[\mathbf{T}^e_{\tilde{q}\tilde{q}}]^{-1}\mathbf{R}^e_{\tilde{q}}
\end{aligned}
\tag{C.10}
$$

with the left-hand side blocks being explicitly computed by the assembly of each cell e

contribution on each $i, j \in \mathcal{B}_e$ faces given by

$$
\left\{
\begin{aligned}
&(\mathbf{K}_{\hat{u}\hat{u}})_{i,j}^{e,n} := |\Gamma_{e,i}|\Big\{-\boldsymbol{\tau}_i^{n-1}\delta_{ij} + (\hat{\mathbf{u}}_i^n\cdot\boldsymbol{n}_i)\mathbf{I}_{\mathtt{n_{sd}}}\,\chi_{\mathcal{N}_e}(i)\delta_{ij} + (\hat{\mathbf{u}}_i^n\otimes\boldsymbol{n}_i)\,\chi_{\mathcal{N}_e}(i)\delta_{ij} \\
&\quad +|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}\boldsymbol{\tau}_j^{n-1} - |\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}\mathbf{I}_{\mathtt{n_{sd}}}(\hat{\mathbf{u}}_j^n\cdot\boldsymbol{n}_j) \\
&\quad -|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\hat{\mathbf{u}}_j^n\otimes\boldsymbol{n}_j) - |\Gamma_{e,j}|\frac{1+\hat{\nu}_i^n f_{v1}(\hat{\nu}_i^n)}{|\Omega_e|Re}(\boldsymbol{n}_i\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}} \\
&\quad -\frac{1}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}\big(\mathbf{I}_{\mathtt{n_{sd}}}\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)\cdot\boldsymbol{n}_j + \boldsymbol{n}_j\otimes\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)\big) \\
&\quad -\frac{|\Omega_e|}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\left(\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}\otimes\boldsymbol{n}_j\right)[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\hat{\nu}_j \\
&\quad -\frac{|\Omega_e|}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\left(\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}\otimes\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\mathbf{L}_e^n}\cdot\boldsymbol{n}_j\right)[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\Big\} \\
&(\mathbf{K}_{\hat{u}\rho})_i^{e,n} := |\Gamma_{e,i}|\boldsymbol{n}_i \\
&(\mathbf{K}_{\hat{u}\hat{\nu}})_{i,j}^{e,n} := |\Gamma_{e,i}|\Big\{\frac{1}{Re}\mathbf{L}_e^n\boldsymbol{n}_i\frac{\partial(\hat{\nu}_i^n f_{v1}(\hat{\nu}_i^n))}{\partial\hat{\nu}_i^n}\delta_{ij} \\
&\quad -\frac{1}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\mathbf{q}}_e^n}\boldsymbol{n}_j \\
&\quad +\frac{|\Omega_e|}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\big(\tilde{\tau}_j^{n-1}-(\hat{\mathbf{u}}_j^n\cdot\boldsymbol{n}_j)\big) \\
&\quad -\frac{|\Omega_e|}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n f_{n,e}(\tilde{\nu}_e^n)\cdot\boldsymbol{n}_j\right) \\
&\quad -\frac{|\Omega_e|}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n\tilde{\nu}_e^n\frac{\partial f_{n,e}(\tilde{\nu}_e^n)}{\partial\tilde{\nu}_e^n}\cdot\boldsymbol{n}_j\right) \\
&\quad -\frac{|\Omega_e|}{Re}|\Gamma_{e,j}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\tilde{\mathbf{q}}_e^n}\cdot\boldsymbol{n}_j\right)\Big\}
\end{aligned}
\right.
\tag{C.11a}
$$

$$
\left\{
\begin{aligned}
&(\mathbf{K}_{\rho\hat{u}})_j^{e,n} := |\Gamma_{e,j}|\boldsymbol{n}_j \\
&(\mathbf{K}_{\hat{\nu}\hat{u}})_{i,j}^{e} := |\Gamma_{e,i}|\Big\{\hat{\nu}_i^n\,\chi_{\mathcal{N}_e}(i)\delta_{ij}-|\Gamma_{e,j}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\Big(\hat{\nu}_j\boldsymbol{n}_j-\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\mathbf{L}_e^n}\cdot\boldsymbol{n}_j\Big)\Big\} \\
&(\mathbf{K}_{\hat{\nu}\hat{\nu}})_{i,j}^{e} := |\Gamma_{e,i}|\Big\{-\tilde{\tau}_i^{n-1}\delta_{ij} + (\hat{\mathbf{u}}_i^n\cdot\boldsymbol{n}_i)\,\chi_{\mathcal{N}_e}(i)\delta_{ij} + \frac{1}{\sigma Re}\;\tilde{\mathbf{q}}_e^n\cdot\boldsymbol{n}_i\frac{\partial(\hat{\nu}_i^n f_n(\hat{\nu}_i^n))}{\partial\hat{\nu}_i^n}\delta_{ij} \\
&\quad -|\Gamma_{e,j}|\frac{1+\hat{\nu}_i^n f_n(\hat{\nu}_i^n)}{|\Omega_e|\sigma Re}\,\boldsymbol{n}_i\cdot\boldsymbol{n}_j+|\Gamma_{e,j}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\Big(\tilde{\tau}_j^{n-1}-(\hat{\mathbf{u}}_j^n\cdot\boldsymbol{n}_j)\Big) \\
&\quad -|\Gamma_{e,j}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n f_{n,e}(\tilde{\nu}_e^n)\cdot\boldsymbol{n}_j-|\Gamma_{e,j}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n\tilde{\nu}_e^n\frac{\partial f_{n,e}(\tilde{\nu}_e^n)}{\partial\tilde{\nu}_e^n}\cdot\boldsymbol{n}_j \\
&\quad -|\Gamma_{e,j}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\tilde{\mathbf{q}}_e^n}\cdot\boldsymbol{n}_j\Big\}
\end{aligned}
\right.
\tag{C.11b}
$$

The right-hand side blocks are explicitly computed by the assembly of each cell e contribution on each $i \in \mathcal{B}_e$ face as follows

$$
\left\{
\begin{aligned}
&(\mathbf{F}_{\hat{u}})_i^{e,n} := -(\mathbf{R}_{\hat{u}})_i^{e,n} + |\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}\mathbf{R}_u^{e,n} + |\Gamma_{e,i}|\frac{1+\hat{\nu}_i^n f_{v1}(\hat{\nu}_i^n)}{|\Omega_e| Re}\mathbf{R}_L^{e,n}\cdot\boldsymbol{n}_i \\
&+\frac{1}{Re}|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}\Big(\mathbf{R}_L^{e,n}\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n) + [\mathbf{R}_L^{e,n}]^T\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)\Big) \\
&+\frac{1}{Re}|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\mathbf{q}}_e^n}\mathbf{R}_{\tilde{q}}^{e,n} \\
&+\frac{|\Omega_e|}{Re}|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\mathbf{R}_{\tilde{\nu}}^{e,n} \\
&-\frac{|\Omega_e|}{Re}|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\mathbf{L}_e^n}:\mathbf{R}_L^{e,n}\right) \\
&+\frac{|\Omega_e|}{Re}|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n f_{n,e}(\tilde{\nu}_e^n)\cdot\mathbf{R}_{\tilde{q}}^{e,n}\right) \\
&+\frac{|\Omega_e|}{Re}|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n\tilde{\nu}_e^n\frac{\partial f_{n,e}(\tilde{\nu}_e^n)}{\partial\tilde{\nu}_e^n}\cdot\mathbf{R}_{\tilde{q}}^{e,n}\right) \\
&+\frac{|\Omega_e|}{Re}|\Gamma_{e,i}|\boldsymbol{\tau}_i^{n-1}[\mathbf{T}_{uu}^{e,n}]^{-1}(\mathbf{L}_e^n+[\mathbf{L}_e^n]^T)\frac{\partial\mathbf{q}_{t,e}(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n)}{\partial\tilde{\nu}_e^n}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\tilde{\mathbf{q}}_e^n}\cdot\mathbf{R}_{\tilde{q}}^{e,n}\right) \\
&(\mathbf{F}_\rho)^{e,n} := -\mathbf{R}_\rho^{e,n} \\
&(\mathbf{F}_{\hat{\nu}})_i^{e,n} := -(\mathbf{R}_{\hat{\nu}})_i^{e,n} + |\Gamma_{e,i}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\mathbf{R}_{\tilde{\nu}}^{e,n} \\
&+|\Gamma_{e,i}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\mathbf{L}_e^n}:\mathbf{R}_L^{e,n}\right) \\
&+|\Gamma_{e,i}|\frac{1+\hat{\nu}_i^n f_n(\hat{\nu}_i^n)}{|\Omega_e|\sigma Re}\boldsymbol{n}_i\cdot\mathbf{R}_{\tilde{\nu}}^{e,n} + |\Gamma_{e,i}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n f_{n,e}(\tilde{\nu}_e^n)\cdot\mathbf{R}_{\tilde{q}}^{e,n}\right) \\
&+|\Gamma_{e,i}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{2}{\sigma Re}\tilde{\mathbf{q}}_e^n\tilde{\nu}_e^n\frac{\partial f_{n,e}(\tilde{\nu}_e^n)}{\partial\tilde{\nu}_e^n}\cdot\mathbf{R}_{\tilde{q}}^{e,n}\right) \\
&+|\Gamma_{e,i}|\tilde{\tau}_i^{n-1}[\mathbf{T}_{\tilde{\nu}\tilde{\nu}}^{e,n}]^{-1}\left(\frac{\partial\mathrm{s}_e(\tilde{\nu}_e^n,\tilde{\mathbf{q}}_e^n,\mathbf{L}_e^n)}{\partial\tilde{\mathbf{q}}_e^n}\cdot\mathbf{R}_{\tilde{q}}^{e,n}\right)
\end{aligned}
\right. \tag{C.12}
$$

The expressions for $\mathbf{R}_L^{e,n}$, $\mathbf{R}_u^{e,n}$, $\mathbf{R}_{\tilde{q}}^{e,n}$, $\mathbf{R}_{\tilde{\nu}}^{e,n}$, $(\mathbf{R}_{\hat{u}})_i^{e,n}$, $\mathbf{R}_\rho^{e,n}$, and $(\mathbf{R}_{\hat{\nu}})_i^{e,n}$ are already stated in Eq. (2.33) and Eq. (2.34).

C.1.1 Linearisation of the negative Sparlat-Allmaras functions

This section discusses the linearisation of the Sparlat-Allmaras functions, introduced in equations (C.3) and (C.6) and also appearing in equations (C.11) and (C.12). We recall the definition of the discrete form of the Sparlat-Allmaras negative model source term defined locally at cell e level.

$$
\mathrm{s}_e = \begin{cases} c_{b1}(1-f_{t2,e})\tilde{S}_e\tilde{\nu}_e + \dfrac{c_{b2}}{\sigma Re}\tilde{\mathbf{q}}_e\cdot\tilde{\mathbf{q}}_e - \dfrac{1}{Re}\left(c_{w1,e}f_{w,e} - \dfrac{c_{b1}}{\kappa^2}f_{t2,e}\right)\left(\dfrac{\tilde{\nu}_e}{d_e}\right)^2 & \text{if } \tilde{\nu}_e \geq 0, \\ c_{b1}(1-c_{t3})S_e\tilde{\nu}_e + \dfrac{c_{b2}}{\sigma Re}\tilde{\mathbf{q}}_e\cdot\tilde{\mathbf{q}}_e + \dfrac{c_{w1}}{Re}\left(\dfrac{\tilde{\nu}_e}{d_e}\right)^2 & \text{otherwise,} \end{cases} \tag{C.13}
$$

The subindex e denotes the evaluation of functions, defined in (2.4), at the cell e centre in terms of the cell centre variables. The implicit form of the discrete Sparlat-Allmaras source term, s_e, derivatives, appearing in the blocks $(\mathbf{T}_{\tilde{\nu}L})^{e,n}$, $(\mathbf{T}_{\tilde{\nu}\tilde{\nu}})^{e,n}$, and $(\mathbf{T}_{\tilde{\nu}\tilde{q}})^{e,n}$ of Eq. (C.3), are given by

$$
\frac{\partial \mathrm{s}_e}{\partial \mathbf{L}_e} := \begin{cases} c_{b1}(1-f_{t2,e})\dfrac{\partial \tilde{S}_e}{\partial \mathbf{L}_e}\tilde{\nu}_e - \dfrac{1}{Re}c_{w1}\dfrac{\partial f_{w,e}}{\partial \mathbf{L}_e}\left(\dfrac{\tilde{\nu}_e}{d_e}\right)^2 & \text{if } \tilde{\nu}_e \geq 0, \\ c_{b1}(1-c_{t3})\dfrac{\partial S_e}{\partial \mathbf{L}_e}\tilde{\nu}_e & \text{otherwise,} \end{cases} \tag{C.14a}
$$

$$
\frac{\partial \mathrm{s}_e}{\partial \tilde{\nu}_e} := \begin{cases} c_{b1}(1-f_{t2,e})\tilde{S}_e + c_{b1}(1-f_{t2})\dfrac{\partial \tilde{S}_e}{\partial \tilde{\nu}_e}\tilde{\nu}_e \\ \qquad - c_{b1}\dfrac{\partial f_{t2,e}}{\partial \tilde{\nu}_e}\tilde{S}_e\tilde{\nu}_e - \dfrac{2}{Re}\left(c_{w1}f_{w,e} - \dfrac{c_{b1}}{\kappa^2}f_{t2,e}\right)\dfrac{\tilde{\nu}_e}{d_e^2} \\ \qquad\qquad - \dfrac{1}{Re}c_{w1}\dfrac{\partial f_{w,e}}{\partial \tilde{\nu}_e}\left(\dfrac{\tilde{\nu}_e}{d_e}\right)^2 + \dfrac{1}{Re}\dfrac{c_{b1}}{\kappa^2}\dfrac{\partial f_{t2,e}}{\partial \tilde{\nu}_e}\left(\dfrac{\tilde{\nu}_e}{d_e}\right)^2 & \text{if } \tilde{\nu}_e \geq 0, \\ c_{b1}(1-c_{t3})S_e + 2\dfrac{c_{w1}}{Re}\dfrac{\tilde{\nu}_e}{d_e^2} & \text{otherwise,} \end{cases} \tag{C.14b}
$$

$$
\frac{\partial \mathrm{s}_e}{\partial \tilde{\mathbf{q}}_e} := \begin{cases} 2\dfrac{c_{b2}}{\sigma Re}\tilde{\boldsymbol{q}} & \text{if } \tilde{\nu}_e \geq 0, \\ 2\dfrac{c_{b2}}{\sigma Re}\tilde{\boldsymbol{q}} & \text{otherwise,} \end{cases} \tag{C.14c}
$$

The computation of $\frac{\partial \tilde{S}_e}{\partial \mathbf{L}_e}$, $\frac{\partial S_e}{\partial \mathbf{L}_e}$, and $\frac{\partial f_{w,e}}{\partial \mathbf{L}_e}$, appearing in (C.14a) is explained in Remark C.1

Remark C.1 (Computation of $\frac{\partial \tilde{S}_e}{\partial \mathbf{L}_e}$, $\frac{\partial f_{w,e}}{\partial \mathbf{L}_e}$, and $\frac{\partial S_e}{\partial \mathbf{L}_e}$). For the computation of $\frac{\partial \tilde{S}_e}{\partial \mathbf{L}_e}$ we evoke the chain rule

$$
\frac{\partial \tilde{S}_e}{\partial \mathbf{L}_e} := \frac{\partial \tilde{S}_e}{\partial S_e}\frac{\partial S_e}{\partial \mathbf{L}_e} \tag{C.15}
$$

Let us recall the discrete definition of the modified vorticity $\tilde{S}$, from Allmaras et al. (2012), particularised at the cell e

$$
\tilde{S}_e := \begin{cases} S_e + \bar{S}_e & \text{if } \bar{S}_e \geq -c_{v2}S_e, \\ S_e + \dfrac{S_e(S_e c_{v2}^2 + \bar{S}_e c_{v3})}{S_e(c_{v3} - 2c_{v2}) - \bar{S}_e} & \text{otherwise,} \end{cases}
$$

where $S_e = \sqrt{2\mathbf{S}_e : \mathbf{S}_e}$ is the magnitude of the vorticity, $\mathbf{S}_e = (\boldsymbol{\nabla}\mathbf{u}_e - \boldsymbol{\nabla}^T\mathbf{u}_e)/2$ is the vorticity tensor, $\bar{S}_e = \tilde{\nu} f_{v2,e}/(Re\,\kappa^2 d_e^2)$ and d_e is the minimum distance to the wall for the cell e. From the definition of $\tilde{S}_e$ we can find

$$\frac{\partial \tilde{S}_e}{\partial S_e} := \begin{cases} 1 & : \ \bar{S}_e \geq -c_{v2} S_e \\ 1 + \dfrac{(2c_{v2}^2 S_e + c_{v3}\bar{S}_e)}{(c_{v3} - 2c_{v2})S_e - \bar{S}_e} - \dfrac{S_e(c_{v2}^2 S_e + c_{v3}\bar{S}_e)(c_{v3} - 2c_{v2})}{((c_{v3} - 2c_{v2})S_e - \bar{S}_e)^2} & : \ \bar{S}_e < -c_{v2} S_e \end{cases} \tag{C.16}$$

With the help of the mixed variable $\mathbf{L}_e$, the vorticity tensor $\mathbf{S}$, and its magnitude S, at element e level, are written as

$$\mathbf{S}_e := \frac{\mathbf{L}_e^T - \mathbf{L}_e}{2}, \quad S_e := \sqrt{2\mathbf{S}_e{:}\mathbf{S}_e}$$

From algebraic manipulation, the derivative with respect to $\mathbf{L}_e$ reduces to

$$\frac{\partial S_e}{\partial \mathbf{L}_e} := \frac{1}{S_e}(\mathbf{L}_e^T - \mathbf{L}_e) \tag{C.17}$$

The chain derivative in (C.15) is computed with the help of (C.16) and (C.17). For the computation of $\frac{\partial f_{w,e}}{\partial \mathbf{L}_e}$ we also evoke the chain rule

$$\frac{\partial f_{w,e}}{\partial \mathbf{L}_e} = \frac{\partial f_{w,e}}{\partial g_e}\frac{\partial g_e}{\partial r_e}\frac{\partial r_e}{\tilde{S}_e}\frac{\partial \tilde{S}_e}{\partial S_e}\frac{\partial S_e}{\partial \mathbf{L}_e} \tag{C.18}$$

with the definitions introduced in (2.4) and particularised at the cell centre e we obtain

$$\frac{\partial f_{w,e}}{\partial g_e} := \left[\frac{1 + c_{w3}^6}{g_e^6 + c_{w3}^6}\right]^{1/6} - g_e^6\left[\frac{1 + c_{w3}^6}{(g_e^6 + c_{w3}^6)^7}\right]^{1/6} \tag{C.19a}$$

$$\frac{\partial g_e}{\partial r_e} := 1 + c_{w2}(6r_e^5 - 1), \tag{C.19b}$$

$$\frac{\partial r_e}{\partial \tilde{S}_e} := \begin{cases} -\dfrac{\tilde{\nu}_e}{Re\ \tilde{S}_e^2 \kappa^2 d_e^2} & : \ \dfrac{\tilde{\nu}_e}{Re\ \tilde{S}_e \kappa^2 d_e^2} < r_{lim} \\ 0 & : \ \dfrac{\tilde{\nu}_e}{Re\ \tilde{S}_e \kappa^2 d_e^2} \geq r_{lim} \end{cases} \tag{C.19c}$$

Finally, right-had side of the chain derivative in (C.18) is evaluated with equations (C.19), (C.16), and (C.17).

The computation of $\frac{\partial \tilde{S}_e}{\partial \tilde{\nu}_e}$, and $\frac{\partial f_{w,e}}{\partial \tilde{\nu}_e}$, appearing in (C.14b) is explained in Remark C.2.

Remark C.2 (Computation of $\frac{\partial \tilde{S}_e}{\partial \tilde{\nu}_e}$, and $\frac{\partial f_{w,e}}{\partial \tilde{\nu}_e}$). For the computation of $\frac{\partial \tilde{S}_e}{\partial \tilde{\nu}_e}$ we recall the definition of $\tilde{S}_e$

$$\tilde{S}_e := \begin{cases} S_e + \bar{S}_e & \text{if } \bar{S}_e \geq -c_{v2} S_e, \\ S_e + \dfrac{S_e(S_e c_{v2}^2 + \bar{S}_e c_{v3})}{S_e(c_{v3} - 2c_{v2}) - \bar{S}_e} & \text{otherwise,} \end{cases}$$

Taking the derivative of $\tilde{S}_e$ with respect to $\tilde{\nu}_e$ one can obtain

$$\frac{\partial \tilde{S}_e}{\partial \tilde{\nu}_e} := \begin{cases} \dfrac{\partial \bar{S}_e}{\partial \tilde{\nu}_e} & : \ \bar{S}_e \geq -C_{v2} S_e \\ \dfrac{\partial \bar{S}_e}{\partial \tilde{\nu}_e} \left(\frac{c_{v3} S_e ((c_{v3} - 2c_{v2}) S_e - \bar{S}_e) + S_e (c_{v2}^2 S_e + c_{v3} \bar{S}_e)}{((c_{v3} - 2C_{v2}) S_e - \bar{S}_e)^2} \right) & : \ \bar{S}_e < -c_{v2} S_e \end{cases} \tag{C.20}$$

with the definitions introduced in (2.4) and particularised at the cell centre e we obtain

$$\frac{\partial \bar{S}_e}{\partial \tilde{\nu}_e} := \frac{1}{Re\ \kappa^2 d_e^2} f_{v2,e} + \frac{\tilde{\nu}_e}{Re\ \kappa^2 d_e^2} \frac{\partial f_{v2,e}}{\partial \tilde{\nu}_e} \tag{C.21a}$$

$$\frac{\partial f_{v2,e}}{\partial \tilde{\nu}_e} := -\frac{1}{1 + \tilde{\nu}_e f_{v1,e}} + \frac{\tilde{\nu}_e}{(1 + \tilde{\nu}_e f_{v1,e})^2} \left(f_{v1} + \tilde{\nu}_e \frac{\partial f_{v1,e}}{\partial \tilde{\nu}_e} \right) \tag{C.21b}$$

$$\frac{\partial f_{v1,e}}{\partial \tilde{\nu}_e} := \frac{3\tilde{\nu}_e^2}{\tilde{\nu}_e^3 + c_{v1}^3} \left(1 - f_{v1,e} \right) \tag{C.21c}$$

The derivative in (C.20) is evaluated with the help of the equations in (C.21).

For the computation of $\frac{\partial f_{w,e}}{\partial \tilde{\nu}_e}$ we evoke the chain rule, written as

$$\frac{\partial f_{w,e}}{\partial \tilde{\nu}_e} = \frac{\partial f_{w,e}}{\partial g_e} \frac{\partial g_e}{\partial r_e} \frac{\partial r_e}{\partial \tilde{\nu}_e} \tag{C.22}$$

with the definitions introduced in (2.4) and particularised at the cell centre e we obtain

$$\frac{\partial f_{w,e}}{\partial g_e} := \left[\frac{1 + c_{w3}^6}{g_e^6 + c_{w3}^6} \right]^{1/6} - g_e^6 \left[\frac{1 + c_{w3}^6}{(g_e^6 + c_{w3}^6)^7} \right]^{1/6} \tag{C.23a}$$

$$\frac{\partial g_e}{\partial r_e} := 1 + c_{w2}(6r_e^5 - 1), \tag{C.23b}$$

$$\frac{\partial r_e}{\partial \tilde{\nu}_e} := \begin{cases} \dfrac{1}{Re\ \tilde{S}_e \kappa^2 d_e^2} \left(1 - \dfrac{\tilde{\nu}_e}{\tilde{S}_e} \dfrac{\partial \tilde{S}_e}{\partial \tilde{\nu}_e} \right) & : \ \dfrac{\tilde{\nu}_e}{Re\ \tilde{S}_e \kappa^2 d_e^2} < r_{lim} \\ 0 & : \ \dfrac{\tilde{\nu}_e}{Re\ \tilde{S}_e \kappa^2 d_e^2} \geq r_{lim} \end{cases} \tag{C.23c}$$

Finally, the right-hand side of the chain derivative in (C.22) is evaluated with equations in (C.23).

The derivatives of $\mathbf{q}_{t,e}$, appearing in the blocks $(\mathbf{T}_{u\tilde{q}})^{e,n}$ and $(\mathbf{T}_{u\tilde{\nu}})^{e,n}$ of Eq. (C.3) are computed with

$$\frac{\partial \mathbf{q}_{t,e}}{\partial \tilde{\mathbf{q}}_e} := (4-3f_{v1,e})f_{v1,e}, \tag{C.24a}$$

$$\frac{\partial \mathbf{q}_{t,e}}{\partial \tilde{\nu}_e} := (4-6f_{v1,e})\frac{\partial f_{v1,e}}{\partial \tilde{\nu}_e}\tilde{\mathbf{q}}_e, \tag{C.24b}$$

$$\frac{\partial f_{v1,e}}{\partial \tilde{\nu}_e} := \frac{3\tilde{\nu}_e^2}{\tilde{\nu}_e^3 + c_{v1}^3}\left(1 - f_{v1,e}\right) \tag{C.24c}$$

The derivatives of $f_{n,e}$, appearing in the blocks $(\mathbf{T}_{\tilde{\nu}\tilde{q}})^{e,n}$ and $(\mathbf{T}_{\tilde{\nu}\tilde{\nu}})^{e,n}$ of Eq. (C.3) are computed as

$$\frac{\partial f_{n,e}}{\partial \tilde{\nu}_e} := \frac{3\tilde{\nu}_e^2}{c_{n1}-\tilde{\nu}_e^3}(1+f_{n,e}), \tag{C.25a}$$

$$\frac{\partial^2 f_{n,e}}{\partial \tilde{\nu}_e^2} := \frac{6\tilde{\nu}_e}{c_{n1}-\tilde{\nu}_e^3}\left(1+\frac{3\tilde{\nu}_e^3}{c_{n1}-\tilde{\nu}_e^3}\right)(1+f_{n,e}) \tag{C.25b}$$

The derivatives of terms $\hat{\nu}_i f_{v1}(\hat{\nu}_i)$ and $\hat{\nu}_i f_n(\hat{\nu}_i)$ evaluated at grid skeleton face i, appearing in the blocks $(\mathbf{T}_{\hat{u}\hat{\nu}})^{e,n}$ and $(\mathbf{T}_{\hat{\nu}\hat{\nu}})^{e,n}$ of Eq. (C.6) are computed as

$$\frac{\partial(\hat{\nu}_i f_{v1,i})}{\partial \hat{\nu}_i} := (4 - 3f_{v1,i})f_{v1,i}\delta_{ij} \tag{C.26a}$$

$$\frac{\partial(\hat{\nu}_i f_{n,i})}{\partial \hat{\nu}_i} := f_{n,i}+\frac{3\hat{\nu}_i^2}{c_{n1}-\hat{\nu}_i^3}(1+f_{n,i})\delta_{ij} \tag{C.26b}$$

C.2 FCFV for the incompressible Navier-Stokes equations

Let us recall the linear problem to be solved for the increments of FCFV global variables, namely $\hat{\boldsymbol{u}}$ and ρ

$$\mathbb{K}^{n,m}\Delta\boldsymbol{\Lambda} = -\mathbf{R}_{\Lambda}^{n,m}, \tag{C.27}$$

where

$$\mathbb{K}^{n,m} = \begin{bmatrix} \mathbf{K}_{\hat{u}\hat{u}} & \mathbf{K}_{\hat{u}\rho} \\ \mathbf{K}_{\rho\hat{u}} & \mathbf{0} \end{bmatrix}, \quad \mathbf{R}_{\Lambda} = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_{\rho} \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\rho} \end{Bmatrix}, \quad \mathbf{U} = \begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \end{Bmatrix}, \tag{C.28}$$

with $\mathbf{K}_{ab}$ denoting the tangent matrix obtained by assembling the cell contributions of $(\mathbf{K}_{ab})_e := \partial\mathbf{R}_{e,a}/\partial b$ and $\Delta\circledcirc^{n,m} = \circledcirc^{n,m+1} - \circledcirc^{n,m}$ the solution increment from the Newton-Raphson iteration m to $m+1$. The right-had side of the global linear system of equations of (C.27) is explicitly computed by the assembly of each cell e contribution on each $i, j \in \mathcal{B}_e$

faces given by

$$\begin{cases} (\mathbf{K}_{\hat{u}\hat{u}})^{e,n}_{i,j} := |\Gamma_{e,i}|\Big\{-\boldsymbol{\tau}^{n-1}_i\delta_{ij} + (\hat{\mathbf{u}}^n_i\cdot\boldsymbol{n}_i)\mathbf{I}_{\mathtt{n_{sd}}}\,\chi_{\mathcal{N}_e}(i)\delta_{ij} + (\hat{\mathbf{u}}^n_i\otimes\boldsymbol{n}_i)\,\chi_{\mathcal{N}_e}(i)\delta_{ij} \\ \quad +|\Gamma_{e,j}|\boldsymbol{\tau}^{n-1}_i[\boldsymbol{\alpha}_e(\boldsymbol{\tau}^{n-1}_j)]^{-1}\boldsymbol{\tau}^{n-1}_j - |\Gamma_{e,j}|\boldsymbol{\tau}^{n-1}_i[\boldsymbol{\alpha}_e(\boldsymbol{\tau}^{n-1}_j)]^{-1}\mathbf{I}_{\mathtt{n_{sd}}}(\hat{\mathbf{u}}^n_j\cdot\boldsymbol{n}_j) \\ \quad -|\Gamma_{e,j}|\boldsymbol{\tau}^{n-1}_i[\boldsymbol{\alpha}_e(\boldsymbol{\tau}^{n-1}_j)]^{-1}(\hat{\mathbf{u}}^n_j\otimes\boldsymbol{n}_j) - |\Gamma_{e,j}|\dfrac{|\Omega_e|}{Re}^{-1}(\boldsymbol{n}_i\cdot\boldsymbol{n}_j)\mathbf{I}_{\mathtt{n_{sd}}}\Big\} \\ \quad (\mathbf{K}_{\hat{u}\rho})^e_i := |\Gamma_{e,i}|\boldsymbol{n}_i \\ \quad (\mathbf{K}_{\rho\hat{u}})^e_j := |\Gamma_{e,j}|\boldsymbol{n}^T_j \end{cases} \tag{C.29}$$

for all $i \in \mathcal{B}_e$. With $\boldsymbol{\alpha}_e$ being defined as

$$\boldsymbol{\alpha}_e = |\Omega_e|a_0 + \sum_{j\in\mathcal{A}_e}|\Gamma_{e,j}|\boldsymbol{\tau}^{n-1}_j$$

where the constant a_0 is the first coefficient of the corresponding time marching scheme, as described in Section 2.3.2. The right-hand side is explicitly computed by the assembly of each cell e contribution on each $i \in \mathcal{B}_e$ face on $(\mathbf{R}_{\hat{u}})^{e,n}_i$ and $(\mathbf{R}_\rho)^{e,n}$, with the expressions given in Eq. (2.55).

C.3 Hybrid pressure FCFV for the incompressible Navier-Stokes equations

Let us recall the linear problem to be solved for the increments of hybrid pressure FCFV global variables, namely $\hat{\boldsymbol{u}}$ and $\hat{p}$

$$\mathbb{K}^m\Delta\boldsymbol{\Lambda} = -\mathbf{R}^m_\Lambda, \tag{C.30}$$

where

$$\mathbb{K}^m = \begin{bmatrix} \mathbf{K}_{\hat{u}\hat{u}} & \mathbf{K}_{\hat{u}\hat{p}} \\ \mathbf{K}_{\hat{p}\hat{u}} & \mathbf{K}_{\hat{p}\hat{p}} \end{bmatrix}, \quad \mathbf{R}_\Lambda = \begin{Bmatrix} \mathbf{R}_{\hat{u}} \\ \mathbf{R}_{\hat{p}} \end{Bmatrix}, \quad \boldsymbol{\Lambda} = \begin{Bmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{p}} \end{Bmatrix}, \quad \mathbf{U} \;=\begin{Bmatrix} \mathbf{L} \\ \mathbf{u} \\ \mathbf{p} \end{Bmatrix}, \tag{C.31}$$

with $\mathbf{K}_{ab}$ denoting the tangent matrix obtained by assembling the cell contributions of $(\mathbf{K}_{ab})_e := \partial\mathbf{R}_{e,a}/\partial b$ and $\Delta\circledcirc^m = \circledcirc^{m+1} - \circledcirc^m$ the solution increment from the Newton-Raphson iteration m to $m+1$. The right-hand side of the global linear system of equations of (C.30) is explicitly computed by the assembly of each cell e contribution on each i, j faces given by

$$\begin{cases} (\mathbf{K}_{\hat{u}\hat{u}})^e_{i,j} := |\Gamma_{e,i}|\Big\{ -\boldsymbol{\tau}_i\delta_{ij} + (\hat{\mathbf{u}}_i \cdot \boldsymbol{n}_i)\mathbf{I}_{\mathtt{n_{sd}}}\,\chi_{\mathcal{N}_e}(i)\delta_{ij} + (\hat{\mathbf{u}}_i \otimes \boldsymbol{n}_i)\,\chi_{\mathcal{N}_e}(i)\delta_{ij} \\ \quad + |\Gamma_{e,j}|\boldsymbol{\tau}_i\boldsymbol{\alpha}_e^{-1}\boldsymbol{\tau}_j - |\Gamma_{e,j}|\boldsymbol{\tau}_i\boldsymbol{\alpha}_e^{-1}\mathbf{I}_{\mathtt{n_{sd}}}(\hat{\mathbf{u}}_j \cdot \boldsymbol{n}_j) \\ \quad - |\Gamma_{e,j}|\boldsymbol{\tau}_i\boldsymbol{\alpha}_e^{-1}(\hat{\mathbf{u}}_j^n \otimes \boldsymbol{n}_j) - |\Gamma_{e,j}||\Omega_e|^{-1}\mathbf{N}_i^T\mathbf{D}(\mathbf{N}_j - \mathbf{E}\dfrac{\mathbf{E}^T}{3}\mathbf{N}_j)\Big\} & \text{for}\, i,j \in \mathcal{B}_e, \\ (\mathbf{K}_{\hat{u}\hat{p}})^e_{i,j} := |\Gamma_{e,i}|\Big\{ -|\Gamma_{e,j}|\boldsymbol{\tau}_i\boldsymbol{\alpha}_e^{-1}\boldsymbol{n}_j + \chi_{\mathcal{N}_e}(i)\boldsymbol{n}_i\delta_{ij} \Big\} & \text{for}\, i \in \mathcal{B}_e \,\text{and}\, j \in \mathcal{A}_e, \\ (\mathbf{K}_{\hat{p}\hat{u}})^e_{i,j} := -|\Gamma_{e,i}||\Gamma_{e,j}|\gamma_e^{-1}\tau_i^p\boldsymbol{n}_j^T & \text{for}\, i \in \mathcal{A}_e \,\text{and}\, j \in \mathcal{B}_e, \\ (\mathbf{K}_{\hat{p}\hat{p}})^e_{i,j} := |\Gamma_{e,i}|\Big\{ |\Gamma_{e,j}|\gamma_e^{-1}\tau_i^p\tau_j^p - \tau_i^p\delta_{ij} \Big\} & \text{for}\, i,j \in \mathcal{A}_e, \end{cases} \tag{C.32}$$

with $\boldsymbol{\alpha}_e$, and γ_e being defined as

$$\boldsymbol{\alpha}_e := \sum_{j \in \mathcal{A}_e} |\Gamma_{e,j}|\boldsymbol{\tau}_j, \quad \gamma_e := \sum_{j \in \mathcal{A}_e} |\Gamma_{e,j}|\tau_j^p$$

The right-hand side is explicitly computed by the assembly of each cell e contribution on each i face on $(\mathbf{R}_{\hat{u}})^e_i$ and $(\mathbf{R}_{\hat{p}})^e_i$, with the expressions given in Eq. (3.35).

Bibliography

Allmaras, S. R., F. T. Johnson, and P. R. Spalart (2012). Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In *Seventh International Conference on Computational Fluid Dynamics*, pp. 1902.

Altinisik, A., O. Yemenici, and H. Umur (2015, 08). Aerodynamic Analysis of a Passenger Car at Yaw Angle and Two-Vehicle Platoon. *Journal of Fluids Engineering 137*(12), 121107.

Amestoy, P. R., A. Buttari, J.-Y. L'Excellent, and T. Mary (2019). Performance and scalability of the block low-rank multifrontal factorization on multicore architectures. *ACM Trans. Math. Softw. 45*(1), 1–23.

Amestoy, P. R., I. S. Duff, J.-Y. L'Excellent, and J. Koster (2001). A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications 23*(1), 15–41.

Arnold, D. N., F. Brezzi, B. Cockburn, and D. Marini (2000). Discontinuous galerkin methods for elliptic problems. In B. Cockburn, G. E. Karniadakis, and C.-W. Shu (Eds.), *Discontinuous Galerkin Methods*, Berlin, Heidelberg, pp. 89–101. Springer Berlin Heidelberg.

Ascher, U. M. and L. R. Petzold (1998). *Computer methods for ordinary differential equations and differential-algebraic equations*, Volume 61. SIAM.

Aupoix, B. and P. Spalart (2003). Extensions of the spalart–allmaras turbulence model to account for wall roughness. *International Journal of Heat and Fluid Flow 24*(4), 454–462. Selected Papers from the Fifth International Conference on Engineering Turbulence Modelling and Measurements.

Balay, S., W. D. Gropp, L. C. McInnes, B. F. Smith, et al. (2023). *PETSc Users Manual.* Argonne National Laboratory.

Barth, T., R. Herbin, and M. Ohlberger (2017). Finite volume methods: Foundation and analysis. In E. Stein, R. de Borst, and T. Hughes (Eds.), *Encyclopedia of Computational Mechanics Second Edition*, pp. 1–60. John Wiley & Sons, Ltd.

Biedron, R. T., J. M. Derlaga, P. A. Gnoffo, D. P. Hammond, W. T. Jones, B. Kleb, E. M. Lee-Rausch, E. J. Nielsen, M. A. Park, C. L. Rumsey, J. L. Thomas, and W. A. Wood (2014). *FUN3D Manual.* National Aeronautics and Space Administration, Langley Research Center.

Bucker, H. M., B. Pullul, and A. Rasch (2009). On CFL evolution strategies for implicit upwind methods in linearized Euler equations. *International Journal for Numerical Methods in Fluids 59*, 1–18.

Ceze, M. and K. Fidkowski (2013). Pseudo-transient continuation, solution update methods, and cfl strategies for dg discretizations of the rans-sa equations. In *21st AIAA Computational Fluid Dynamics Conference*, pp. 2686.

Chapman, S. J. (2018). *Fortran for scientists and engineers.* Mc Graw Hill Education.

Cheung, S. W., E. Chung, H. H. Kim, and Y. Qian (2015). Staggered discontinuous Galerkin methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics 302*, 251–266.

Childs, P. (2011). *Rotating flow*, Chapter 6, pp. 185–192. Butterworth-Heinemann.

Cockburn, B., B. Dong, and J. Guzmán (2008). A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems. *Mathematics of Computation 77*(264), 1887–1916.

Cockburn, B. and J. Gopalakrishnan (2004). A characterization of hybridized mixed methods for second order elliptic problems. *SIAM Journal on Numerical Analysis 42*(1), 283–301.

Cockburn, B. and J. Gopalakrishnan (2005a). Incompressible finite elements via hybridization. I. The Stokes system in two space dimensions. *SIAM Journal on Numerical Analysis 43*(4), 1627–1650.

Cockburn, B. and J. Gopalakrishnan (2005b). New hybridization techniques. *GAMM-Mitt. 28*(2), 154–182.

Cockburn, B. and J. Gopalakrishnan (2009). The derivation of hybridizable discontinuous Galerkin methods for Stokes flow. *SIAM Journal on Numerical Analysis 47*(2), 1092–1125.

Cockburn, B., J. Gopalakrishnan, and R. Lazarov (2009). Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis 47*(2), 1319–1365.

Cockburn, B., G. E. Karniadakis, and C.-W. Shu (2000). The development of discontinuous galerkin methods. In B. Cockburn, G. E. Karniadakis, and C.-W. Shu (Eds.), *Discontinuous Galerkin Methods*, Berlin, Heidelberg, pp. 3–50. Springer Berlin Heidelberg.

Cockburn, B., N. C. Nguyen, and J. Peraire (2010). A comparison of HDG methods for Stokes flow. *Journal of Scientific Computing 45*(1-3), 215–237.

Crivellini, A. and F. Bassi (2011). An implicit matrix-free discontinuous Galerkin solver for viscous and turbulent aerodynamic simulations. *Computers & Fluids 50*, 81–93.

Crivellini, A., V. D'Alessandro, and F. Bassi (2013). A Spalart-Allmaras turbulence model implementation in a discontinuos Galerkin solver for incompressible flows. *Journal of Computational Physics 241*, 388–415.

Crivellini, A., V. D'Alessandro, and F. Bassi (2013). Assessment of a high-order discontinuous Galerkin method for incompressible three-dimensional Navier–Stokes equations: Benchmark results for the flow past a sphere up to Re=500. *Computers & Fluids 86*, 442–458.

del Álamo, J. C. and J. Jiménez (2003). Spectra of the very large anisotropic scales in turbulent channels. *Physics of Fluids 15*(6), L41–L44.

Diskin, B. and J. L. Thomas (2011). Comparison of node-centered and cell-centered unstructured finite-volume discretizations: inviscid fluxes. *AIAA Journal 49*(4), 836–854.

Diskin, B., J. L. Thomas, E. J. Nielsen, H. Nishikawa, and J. A. White (2010). Comparison of node-centered and cell-centered unstructured finite-volume discretizations: viscous fluxes. *AIAA Journal 48*(7), 1326.

Donea, J. and A. Huerta (2003). *Finite Element Methods for Flow Problems.* John Wiley & Sons, Ltd.

Droniou, J. (2014). Finite volume schemes for diffusion equations: Introduction to and review of modern methods. *Mathematical Models and Methods in Applied Sciences 24*(08), 1575–1619.

Ebrahim, H. and R. Dominy (2020). Wake and surface pressure analysis of vehicles in platoon. *Journal of Wind Engineering and Industrial Aerodynamics 201*, 104144.

Economon, T. D. (2019). The SU2 tutorial collection. https://su2code.github.io/tutorials/home/.

Eça, L., M. Hoekstra, A. Hay, and D. Pelletier (2007). On the construction of manufactured solutions for one and two-equation eddy-viscosity models. *International Journal for Numerical Methods in Fluids 54*(2), 119–154.

Eça, L., C. Klaij, G. Vaz, M. Hoekstra, and F. Pereira (2016). On code verification of rans solvers. *Journal of Computational Physics 310*, 418–439.

Ghia, U., K. Ghia, and C. Shin (1982). High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics 48*(3), 387–411.

Giacomini, M., A. Karkoulias, R. Sevilla, and A. Huerta (2018). A superconvergent HDG method for Stokes flow with strongly enforced symmetry of the stress tensor. *Journal of Scientific Computing 77*(3), 1679–1702.

Giacomini, M. and R. Sevilla (2020). A second-order face-centred finite volume method on general meshes with automatic mesh adaptation. *International Journal for Numerical Methods in Engineering 121*(23), 5227–5255.

Giacomini, M., R. Sevilla, and A. Huerta (2020). Tutorial on hybridizable discontinuous Galerkin (HDG) formulation for incompressible flow problems. In L. De Lorenzis and A. Düster (Eds.), *Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids*, Volume 599, pp. 163–201. Cham: Springer International Publishing.

Jacuzzi, E. and K. Granlund (2019). Passive flow control for drag reduction in vehicle platoons. *Journal of Wind Engineering and Industrial Aerodynamics 189*, 104–117.

Jameson, A. (2012). Computational fluid dynamics: Past, present and future. Presented at the 2012 Future Directions in CFD Research, National Institute for Aerospace August 6–8, 2012, Hampton, VA.

Jasak, H. (2009). Openfoam: Open source cfd in research and industry. *International Journal of Naval Architecture and Ocean Engineering 1*(2), 89–94.

Jespersen, D. C., T. H. Pulliam, and M. L. Childs (2016). OVERFLOW: Turbulence Modeling Resource Verification Results. Technical Report ARC-E-DAA-TN35216, NASA Ames Research Center.

Johnson, T. A. and V. C. Patel (1999). Flow past a sphere up to a reynolds number of 300. *Journal of Fluid Mechanics 378*, 19—-70.

Kadapa, C., W. G. Dettmer, and D. Perić (2016). A fictious domain/distributed Lagrange multiplier based fluid-structure interaction scheme with hierachical B-spline grids. *Computer Methods in Applied Mechanics and Engineering 301*, 1–27.

Kadapa, C., W. G. Dettmer, and D. Perić (2020). Accurate iteration-free mixed-stabilised formulation for laminar incompressible Navier-Stokes: Applications to fluid-structure interaction. *Journal of Fluids and Structures 97*, 103077.

Kelley, C. T. and D. E. Keyes (1998). Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis 35*(2), 508–523.

Kessels, P. C. J. (2016, August). Finite element discretization of the spalart-allmaras turbulence model. Master's thesis, Delft University of Technology, Delft, Netherlands. Master of Science Thesis, Faculty of Aerospace Engineering.

Kjellgren, C. (1997). A semi-implicit fractional step finite element method for viscous incompressible flow. *Computational Mechanics 20*, 541–550.

L. Eça, M. Hoekstra, A. H. and D. Pelletier (2007). A manufactured solution for a two-dimensional steady wall-bounded incompressible turbulent flow. *International Journal of Computational Fluid Dynamics 21*(3-4), 175–188.

La Spina, A., M. Kronbichler, M. Giacomini, W. A. Wall, and A. Huerta (2020). A weakly compressible hybridizable discontinuous Galerkin formulation for fluid-structure interaction problems. *Computer Methods in Applied Mechanics and Engineering 372*, 113392.

Launder, B. and D. Spalding (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering 3*(2), 269–289.

Lee, M. and R. D. Moser (2013). Turbulence database cluster. https://turbulence.oden.utexas.edu/.

Lee, M. and R. D. Moser (2015). Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *Journal of Fluid Mechanics 774*, 395–415.

Lee-Rausch, E. M., C. L. Rumsey, and M. A. Park (2015). Grid-adapted fun3d computations for the second high-lift prediction workshop. *Journal of Aircraft 52*(4), 1098–1111.

LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge.

Lodares, D., J. Manzanero, E. Ferrer, and E. Valero (2022). An entropy-stable discontinuous Galerkin approximation of the Spalart-Allmaras turbulence model for the compressible Reynolds averaged Navier-Stokes equations. *Journal of Computational Physics 455*, 110998.

McBride, D., N. Croft, and M. Cross (2007). Finite volume method for the solution of flow on distorted meshes. *International Journal of Numerical Methods for Heat & Fluid Flow 17*, 213–239.

Menter, F. R. (1994). Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal 32*(8), 1598–1605.

Montlaur, A., S. Fernández-Méndez, and A. Huerta (2008). Discontinuous Galerkin methods for the Stokes equations using divergence-free approximations. *International Journal for Numerical Methods in Fluids 57*(9), 1071–1092.

Morton, K. and T. Sonar (2007). Finite volume methods for hyperbolic conservation laws. *Acta Numerica 16*, 155–238.

Moser, R. D., J. Kim, and N. N. Mansour (1999). Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$. *Physics of Fluids 11*(4), 943–945.

Moukalled, F., L. Mangani, and M. Darwish (2016). *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*, Volume 113 of *Fluid Mechanics and its Applications.* Springer.

Mulder, W. A. and B. Van Leer (1985). Experiments with implicit upwind methods for the Euler equations. *Journal of Computational Physics 59*(2), 232–246.

Navah, F. and S. Nadarajah (2018). A comprehensive high-order solver verification methodology for free fluid flows. *Aerospace Science and Technology 80*, 101–126.

Navah, F. and S. Nadarajah (2020). On the verification of cfd solvers of all orders of accuracy on curved wall-bounded domains and for realistic rans flows. *Computers & Fluids 205*, 104504.

Nguyen, N. C., J. Peraire, and B. Cockburn (2010). A hybridizable discontinuous Galerkin method for Stokes flow. *Computer Methods in Applied Mechanics and Engineering 199*(9-12), 582–597.

Nguyen, N. C., J. Peraire, and B. Cockburn (2011). An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations. *Journal of Computational Physics 230*(4), 1147–1170.

Nithiarasu, P. and C. B. Liu (2006). An artificial compressibility based characteristic based split (CBS) scheme for steady and unsteady turbulent incompressible flows. *Computer Methods in Applied Mechanics and Engineering 195*(23–24), 2961–2982.

Oliver, T., K. Estacio-Hiroms, N. Malaya, and G. Carey (2012). Manufactured solutions for the favre-averaged navier-stokes equations with eddy-viscosity turbulence models. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.*

Panneer Selvam, R. (1997). Finite element modelling of flow around a circular cylinder using LES. *Journal of Wind Engineering and Industrial Aerodynamics 67–68*, 129–139.

Patankar, S. V. and D. B. Spalding (1972). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer 15*, 1787–1806.

Peters, E. L. and J. A. Evans (2019). A divergence-conforming hybridized discontinuous galerkin method for the incompressible reynolds-averaged navier-stokes equations. *International Journal for Numerical Methods in Fluids 91*(3), 112–133.

Preacher, H. R., B. P. L. Clair, and A. E. Hamielec (1970). Some relations between drag and flow pattern of viscous flow past a sphere and a cylinder at low and intermediate reynolds numbers. *Journal of Fluid Mechanics 44*(4), 781–790.

Rajani, B. N., A. Kandasamy, and S. Majumdar (2009). Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modelling 33*(3), 1228–1247.

Reed, W. H. and T. R. Hill (1973). Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory.

Reynolds, O. (1895). Iv. on the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London. (A.) 186*, 123–164.

Roos, F. W. and W. W. Willmarth (1971). Some experimental results on sphere and disk drag. *AIAA Journal 9*(2), 285–291.

Roy, C., C. Ober, and T. Smith (2002). Verification of a compressible cfd code using the method of manufactured solutions. In *32nd AIAA Fluid Dynamics Conference and Exhibit*.

Roy, C., E. Tendean, S. Veluri, R. Rifki, E. Luke, and S. Hebert (2007). Verification of rans turbulence models in loci-chem using the method of manufactured solutions. In *18th AIAA Computational Fluid Dynamics Conference*.

Rumsey, C. (2014). SA expected results 2D zero pressure gradient flat plate. https://turbmodels.larc.nasa.gov/flatplate_sa.html.

Saghafian, M., P. K. Stansby, M. S. Saidi, and D. D. Apsley (2003). Simulation of turbulent flows around a circular cylinder using nonlinear eddy-viscosity modelling: steady and oscillatory ambient flows. *Journal of Fluids and Structures 17*(8), 1213–1236.

Schlichting, H. and K. Gersten (2000). *Boundary-Layer Theory* (8th ed.). Springer.

Schwamborn, D., A. Gardner, H. Geyr, A. Krumbein, H. Luedeke, and A. Stürmer (2008, 01). Development of the tau-code for aerospace applications. In *50th NAL International Conference on Aerospace Science and Technology*.

Schwamborn, D., T. Gerhold, and R. Heinrich (2006). The dlr tau-code: Recent applications in research and industry. In *ECCOMAS CFD 2006 conference.*

Sevilla, R. and T. Duretz (2023). A face-centred finite volume method for high-contrast Stokes interface problems. *International Journal for Numerical Methods in Engineering 124*(17), 3709–3732.

Sevilla, R. and T. Duretz (2024). Face-centred finite volume methods for Stokes flows with variable viscosity. *International Journal for Numerical Methods in Engineering 125*(10), e7450.

Sevilla, R., M. Giacomini, and A. Huerta (2018). A face-centred finite volume method for second-order elliptic problems. *International Journal for Numerical Methods in Engineering 115*(8), 986–1014.

Sevilla, R., M. Giacomini, and A. Huerta (2019). A locking-free face-centred finite volume (FCFV) method for linear elastostatics. *Computers & Structures 212*, 43–57.

Sevilla, R., M. Giacomini, A. Karkoulias, and A. Huerta (2018). A superconvergent hybridisable discontinuous Galerkin method for linear elasticity. *International Journal for Numerical Methods in Engineering 116*(2), 91–116.

Sevilla, R. and A. Huerta (2016). Tutorial on Hybridizable Discontinuous Galerkin (HDG) for second-order elliptic problems. In J. Schröder and P. Wriggers (Eds.), *Advanced Finite Element Technologies*, Volume 566 of *CISM International Centre for Mechanical Sciences*, pp. 105–129. Springer International Publishing.

Shang, J. (2004). Three decades of accomplishments in computational fluid dynamics. *Progress in Aerospace Sciences 40*(3), 173–197.

Shur, M. L., M. K. Strelets, A. K. Travin, and P. R. Spalart (2000). Turbulence modeling in rotating and curved channels: Assessing the spalart-shur correction. *AIAA Journal 38*(5), 784–792.

Spalart, P. R. and S. R. Allmaras (1992). A one-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit.* American Institute of Aeronautics and Astronautics.

Spalart, P. R. and S. R. Allmaras (1994). A one-equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale* (1), 5–21.

Spalart, P. R. and A. V. Garbaruk (2020). Correction to the spalart–allmaras turbulence model, providing more accurate skin friction. *AIAA Journal 58*(5), 1903–1905.

Stokes, G. (1845). On the theories of the internal friction of fluids in motion and of the equilibrium and motion of elastic solids. *Transactions of Cambridge philosophical society 8*, 287–305.

Sørensen, K. A., O. Hassan, K. Morgan, and N. P. Weatherill (2003). A multigrid accelerated time-accurate inviscid compressible fluid flow solution algorithm employing mesh movement and local remeshing. *International Journal for Numerical Methods in Fluids 43*(5), 517–536.

Tabata, M. and K. Itakura (1998). A precise computation of drag coefficients of a sphere. *International Journal of Computational Fluid Dynamics 9*(3-4), 303–311.

Taneda, S. (1956). Experimental investigation of the wake behind a sphere at low reynolds numbers. *Journal of the Physical Society of Japan 11*(10), 1104–1108.

Tezduyar, T. E., S. Mittal, S. E. Ray, and R. Shih (1992). Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering 95*(2), 221–242.

Tezduyar, T. E., S. Mittal, and R. Shih (1991). Time-accurate incompressible flow computations with quadrilateral velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering 87*(1–2), 363–384.

Tiberga, M., A. Hennink, J. L. Kloosterman, and D. Lathouwers (2020). A high-order discontinuous galerkin solver for the incompressible RANS equations coupled to the $\kappa - \epsilon$ turbulence model. *Computers & Fluids 212*, 104710.

Tong, O., A. Katz, Y. Yanagita, and D. Work (2017). Verification and validation of a high-order strand grid method for two-dimensional turbulent flows. *Computers & Fluids 154*, 335–346. ICCFD8.

Toro, E. (2009). *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer-Verlag Berlin Heidelberg.

Vieira, L. M., M. Giacomini, R. Sevilla, and A. Huerta (2020). A second-order face-centred finite volume method for elliptic problems. *Computer Methods in Applied Mechanics and Engineering 358*, 112655.

Vieira, L. M., M. Giacomini, R. Sevilla, and A. Huerta (2024). A face-centred finite volume method for laminar and turbulent incompressible flows. *Computers & Fluids 279*, 106339.

Vila-Pérez, J., M. Giacomini, and A. Huerta (2023). Benchmarking the face-centred finite volume method for compressible laminar flows. *International Journal of Numerical Methods for Heat & Fluid Flow 33*(6), 2198–2231.

Vila-Pérez, J., M. Giacomini, R. Sevilla, and A. Huerta (2021). Hybridisable discontinuous Galerkin formulation of compressible flows. *Archives of Computational Methods in Engineering 28*(2), 753–784.

Vila-Pérez, J., M. Giacomini, R. Sevilla, and A. Huerta (2022). A non-oscillatory face-centred finite volume method for compressible flows. *Computers & Fluids 235*, 105272.

Von Kármán, T. (1939, 02). The analogy between fluid friction and heat transfer. *Transactions of the American Society of Mechanical Engineers 61*(8), 705–710.

Wilcox, D. C. (1988). Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal 26*(11), 1299–1310.

Wurst, M., M. Keßler, and E. Krämer (2015). A high-order discontinuous galerkin chimera method for laminar and turbulent flows. *Computers & Fluids 121*, 102–113.

Zhang, F., Z. Zhou, H. Zhang, and X. Yang (2022). A new single formula for the law of the wall and its application to wall-modeled large-eddy simulation. *European Journal of Mechanics - B/Fluids 94*, 350–365.

ZhenHua, J., Y. Chao, Y. Jian, Q. Feng, and Y. Wu (2016). A spalart–allmaras turbulence model implementation for high-order discontinuous galerkin solution of the reynolds-averaged navier-stokes equations. *Flow, Turbulence and Combustion 96*, 623–638.