Titre: Title:	Development of an Efficient Stabilized Navier-Stokes Solver to Simulate Turbulent Flows in Process-Intensified Devices
Auteur: Author:	Laura Prieto Saavedra
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Prieto Saavedra, L. (2025). Development of an Efficient Stabilized Navier-Stokes Solver to Simulate Turbulent Flows in Process-Intensified Devices [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/67662/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/67662/
Directeurs de recherche: Advisors:	Bruno Blais
Programme: Program:	Génie chimique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Development of an Efficient Stabilized Navier-Stokes Solver to Simulate Turbulent Flows in Process-Intensified Devices

LAURA PRIETO SAAVEDRA
Département de génie chimique

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie chimique

Août 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Development of an Efficient Stabilized Navier-Stokes Solver to Simulate Turbulent Flows in Process-Intensified Devices

présentée par **Laura PRIETO SAAVEDRA**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Jean-Philippe HARVEY, président
Bruno BLAIS, membre et directeur de recherche
Roberto PAOLI, membre
Scott MACLACHLAN, membre externe

DEDICATION

To my parents and my brother…

To my uncle Rafael and my Swiss mom Bruna,
who always believed in me but left way too soon…

To all the women who want to pursue a career in STEM,
do not let anyone stop you.

ACKNOWLEDGEMENTS

This thesis is the product of four years of constant work and would not have been possible without the support of countless of persons. I would first like to express my gratitude to my research director Bruno Blais, who gave me the opportunity to come to Montreal and pursue an unusual project in chemical engineering. I am deeply grateful for his guidance, support, and valuable feedback; with his immense knowledge on fluid mechanics, motivation, and trust, he helped me grow as a researcher (and lecturer) in ways I never imagined.

I would like to thank all the other coauthors of the articles presented in this work: Catherine E. Niamh Radburn, Audrey Collard-Daigneault, Joël Archambault and Peter Munch. Without your contribution, this work would not have been possible. I am deeply grateful to Peter Munch, my closest external collaborator, for his valuable input, constant support, and fruitful discussions that greatly contributed to the success of this thesis. I have had the pleasure of working with him since my master's degree and he was the one who introduced me to matrix-free and multigrid methods before I even thought about pursuing a PhD.

I would like to extend my gratitude to all the members of my research group, the CHAOS Laboratory. I have been part of this group for four years and have had the pleasure of meeting numerous interns, master's students, PhD students, and postdocs that made these years an unforgettable experience. I am grateful for all the hackathons, the collaboration within the group, and the activities outside the university. I am extremely proud of everything we have achieved together in Lethe and I would not have wished to pursue my PhD elsewhere.

I would like to thank my friends in other countries for always being there for me despite the distance. In Montreal, thank you to Ghazaleh, Hélène, Paula and David for all the great times together. Thank you to my Lebanese family in Canada: Carla, Toni and Ali. I never thought I would be introduced to the Lebanese culture in Canada, but you made it possible and what a journey it has been. Thank you for being my family away from home and for being there in all the ups and downs of the last four years.

Lastly, I would like to thank my parents and my brother for their unwavering support and for always encouraging me to pursue what I love. You have taught me that I can achieve everything with dedication and discipline, and you will always be my example to follow. The most challenging part of this journey has been being far away from you.

RÉSUMÉ

De la production de biocarburants aux produits pharmaceutiques, l'industrie chimique s'efforce constamment d'améliorer les procédés en termes de durabilité et d'efficacité. Ce domaine est connu sous le nom d'intensification des procédés (Process Intensification en anglais, **PI)** en génie chimique. Elle consiste en la conception d'équipements qui améliorent le transfert de masse et de chaleur sans affecter négativement l'environnement et les coûts du processus, ce qui est possible grâce à l'innovation en termes de taille, de géométrie et de conditions de fonctionnement. Les équipements Hi-Gee (à forte gravité) , les mélangeurs statiques et les tours de distillation réactives sont quelques exemples de dispositifs intensifiés.

Ces dispositifs intensifiés ont des caractéristiques communes qui peuvent être résumées comme suit : géométries complexes (courbes), phénomènes physiques simultanés (turbulence, transfert d'énergie et de masse, réactions chimiques) et écoulements incompressibles transitoires (en rotation) à un nombre de Reynolds intermédiaire (Re $\in [10^3, 10^5]$). L'hydrodynamique à l'intérieur de ces dispositifs est complexe et les méthodes expérimentales ne sont plus suffisantes pour caractériser complètement le comportement de l'écoulement. De plus, le nombre de variables qui peuvent être optimisées conduit à un grand nombre d'expériences qui doivent être réalisées, ce qui se traduit par un investissement important en temps et en argent. Par conséquent, la modélisation numérique de l'écoulement, également connue sous le nom de mécanique des fluides numérique (Computational Fluid Dynamics en anglais, **CFD**), joue un rôle important pour surmonter ces limitations au sein de la PI.

La simulation des écoulements est un domaine de recherche actif qui se concentre sur l'approximation numérique de la solution des équations de Navier–Stokes. Le présent travail vise à développer un solveur Navier–Stokes stabilisé efficace pour simuler les écoulements turbulents incompressibles rencontrés dans les dispositifs intensifiés. Un tel solveur devrait être capable de modéliser avec précision la physique des écoulements turbulents à un coût de calcul raisonnable pour une variété de configurations et une large gamme de conditions d'écoulement. Ceci nécessite une étude détaillée de la modélisation de la turbulence et de la résolution numérique requises pour obtenir une solution approximée des équations incompressibles de Navier–Stokes.

Dans cette thèse, un solveur pour les équations de Navier–Stokes incompressibles est développé et mis en œuvre dans `Lethe`, un logiciel multiphysique CFD libre. Il considère une approche de simulation implicite des grandes structures (Implicit Large-Eddy Simulation en anglais, **ILES**) pour la modélisation de la turbulence et la méthode des éléments finis (Finite

Element Method en anglais, **FEM**) de Galerkin continue pour la discrétisation spatiale. Pour l'approche ILES dans FEM, l'accent est mis sur les techniques de stabilisation «Streamline-Upwind Petrov–Galerkin»(**SUPG**) et «Pressure-Stabilizing Petrov–Galerkin»(**PSPG**). En outre, elle utilise une nouvelle implémentation sans matrice pour réduire les exigences de calcul (consommation de mémoire et temps de résolution), ce qui permet de réaliser des simulations efficaces pour des problèmes pratiques complexes. Ce solveur inclut aussi un solveur linéaire efficace sans matrice utilisant un préconditionneur monolithique géométrique multigrille (Geometric Multigrid en anglais, **GMG**).

D'abord, une étude détaillée de la précision de l'approche ILES utilisée dans le solveur est menée en simulant un écoulement turbulent complexe sur des collines périodiques à des nombres de Reynolds Re $\in [5600, 10\,600, 37\,000]$. Ce cas de référence comprend des caractéristiques d'écoulement complexes qui sont également présentes dans l'écoulement turbulent à l'intérieur des dispositifs intensifiés : séparation et rattachement de l'écoulement, couches limites complexes, recirculation et forts gradients de pression. La précision des simulations est évaluée en comparant les vitesses moyennes, les contraintes de Reynolds et le point de rattachement avec les données expérimentales et de simulation disponibles dans la littérature. Les prédictions obtenues en utilisant l'approche ILES sont en accord avec les données de la littérature, mettant en évidence les avantages par rapport à l'approche de simulation des grandes structures (Large-Eddy Simulation en anglais, **LES**) traditionnelle en ce qui concerne les exigences de calcul. De plus, les effets du pas de temps, du temps de moyennage total et du raffinement global du maillage sont également étudiés, mettant en évidence les caractéristiques clés des méthodes stabilisées, qui comprennent l'utilisation de schémas d'intégration en temps implicites et des résultats précis avec des maillages grossiers.

Ensuite, la nouvelle implémentation du solveur utilisant la modélisation de la turbulence ILES est réalisée à l'aide d'une approche sans matrice. Elle considère la résolution couplée des équations de Navier–Stokes discrétisées en utilisant une méthode de Newton-Krylov avec un préconditionneur GMG. La vérification et la validation du solveur pour des éléments d'ordre $p = 1, 2, 3$ sont effectuées à l'aide d'une variété de cas de référence de complexité croissante. Il s'agit notamment de solutions manufacturées en régime permanent pour des écoulements laminaires et turbulents (Re $= 1, 10\,000$ et Re $\to \infty$), d'un problème d'écoulement en régime permanent autour d'une sphère à Re $= 150$, et du cas du tourbillon turbulent de Taylor–Green à Re $= 1600$. L'efficacité de la implémentation est évaluée en termes de scalabilité parallèle sur les architectures informatiques modernes et de temps de résolution comparé à une mise en œuvre du solveur basée sur une matrice. Les résultats montrent des besoins en mémoire plus faibles, une meilleure scalabilité et une accélération significative lors de l'utilisation de l'implémentation sans matrice, ce qui donne des résultats prometteurs pour les simulations pratiques à grande échelle.

Pour davantage évaluer la implémentation, l'efficacité du solveur est évaluée en détail pour un cas de référence d'écoulement turbulent : l'écoulement de Taylor–Couette avec un cylindre intérieur rotatif. Ce problème présente un écoulement rotatif et des parois courbées, couramment rencontrés dans les dispositifs intensifiés, tels qu'un réacteur à disque rotatif. Le solveur est utilisé pour réaliser des expériences numériques utilisant des discrétisations d'ordre supérieur et des maillages avec jusqu'à 716M degrés de liberté. Les résultats pour l'enstrophie, l'énergie cinétique, la vorticité et les distributions du critère Q sont évalués par rapport aux données de calcul disponibles dans la littérature. De plus, de nouvelles données concernant la dissipation numérique introduite par le schéma sont fournies. Les résultats mettent en évidence les capacités des méthodes d'ordre supérieur dans le contexte de la méthode ILES à obtenir moins de dissipation numérique à un coût de calcul inférieur. Ils soulèvent des questions quant à ce qui constitue une quantité acceptable de dissipation numérique pour la modelisation d'écoulements industriels.

Finalement, le préconditionneur GMG, essentiel pour l'implémentation sans matrice, est étudié plus en détail dans le contexte du raffinement local du maillage. Deux variantes, le lissage local et le grossissement global, sont comparées en utilisant un écoulement autour d'une sphère et un écoulement de Taylor–Couette avec un raffinement de maillage dynamique et local, respectivement. La comparaison porte sur le nombre d'itérations, le coût total de l'algorithme GMG, le coût de calcul de chaque partie de l'algorithme et la scalabilité. Cette étude présente la première comparaison de ces deux variantes dans le contexte d'un solveur stabilisé monolithique sans matrice pour les équations de Navier–Stokes.

ABSTRACT

From biofuel production to pharmaceuticals, the chemical industry is constantly trying to improve processes in terms of sustainability and efficiency. This domain is known as Process Intensification **(PI)** in chemical engineering. It consists of the design of equipments that enhance mass and heat transfer without negatively affecting the environment and process costs through innovation in terms of size, geometry, and operating conditions. Some examples of process-intensified devices include Hi-Gee (high-gravity) equipment, static mixers, and reactive distillation towers.

These process-intensified devices have common characteristics that can be summarized as follows: complex (curved) geometries, concurrent physical phenomena (turbulence, energy and mass transfer, chemical reactions) and (rotating) transient incompressible flows at an intermediate Reynolds number ($\mathrm{Re} \in [10^3, 10^5]$). The hydrodynamics within these devices are complex, and experimental methods are no longer sufficient to fully characterize the flow behavior. Moreover, the number of variables that can be optimized leads to a large number of experiments that need to be performed, which ultimately translate into a large investment of time and money. Therefore, numerical modeling of the flow, also known as Computational Fluid Dynamics **(CFD)**, plays an important role in overcoming these limitations within PI.

The simulation of flows is an active research field that focuses on the numerical approximation of the solution of the Navier–Stokes equations. The present work aims at developing an efficient stabilized Navier–Stokes solver to simulate the incompressible turbulent flows encountered in process-intensified devices. Such a solver should be able to accurately model the physics of turbulent flow at reasonable computational cost for a variety of configurations and a wide range of flow conditions. This requires a detailed study of the turbulence modeling aspect and the numerical resolution behind the incompressible Navier–Stokes equations.

In this thesis, a solver for the incompressible Navier–Stokes equations is developed and implemented in `Lethe`, an open-source CFD multiphysics software. It considers an Implicit Large-Eddy Simulation **(ILES)** approach for turbulence modeling and the continuous Galerkin Finite Element Method **(FEM)** for spatial discretization. For the ILES approach in FEM it focuses on the Streamline-Upwind Petrov–Galerkin **(SUPG)** and the Pressure-Stabilizing Petrov–Galerkin **(PSPG)** stabilization techniques. Moreover, it uses a novel matrix-free implementation to reduce computational requirements (memory consumption and time to solution), leading to efficient simulations for complex practical problems. This also requires the development of an efficient matrix-free linear solver using a Geometric Multigrid **(GMG)** preconditioner.

A detailed study of the accuracy of the ILES approach used within the solver is conducted by simulating a complex turbulent flow over periodic hills at Reynolds numbers $\mathrm{Re} \in \{5600, 10\,600, 37\,000\}$. This benchmark includes complex flow features that are also present in the turbulent flow within process-intensified devices: flow separation and reattachment, complex boundary layers, recirculation, and strong pressure gradients. The accuracy of the simulations is assessed by comparing average velocities, Reynolds stresses, and the reattachment point, against experimental and computational data available in the literature. Good predictions are obtained using the ILES approach, highlighting advantages compared to traditional Large-Eddy Simulation **(LES)** when it comes to computational requirements. Moreover, the effect of the time-step size, the total averaging time, and global mesh refinement is also studied, highlighting key characteristics of stabilized methods, which include the usage of implicit time-stepping schemes and accurate results with coarse meshes.

A novel implementation of the solver using ILES turbulence modeling is achieved using a matrix-free approach. It considers the coupled resolution of the discretized Navier–Stokes solver using a Newton-Krylov method with a monolithic GMG preconditioner. Verification and validation of the solver for elements of order $p = 1, 2, 3$ is carried out using a variety of benchmarks of increasing complexity. These include steady-state manufactured solutions for laminar and turbulent flows ($\mathrm{Re} = 1, 10\,000$ and $\mathrm{Re} \to \infty$), a steady-state flow around a sphere problem at $\mathrm{Re} = 150$, and the Taylor–Green vortex turbulent benchmark at $\mathrm{Re} = 1600$. The efficiency of the implementation is evaluated in terms of parallel scalability on modern computing architectures and time to solution by comparison to a matrix-based implementation of the solver. The results show lower memory requirements, better scalability, and significant speedup when using the matrix-free implementation, yielding promising results for large-scale practical simulations.

To further assess the implementation, the efficiency of the solver is evaluated in detail for a turbulent flow benchmark: the Taylor–Couette flow with an inner rotating cylinder. This problem exhibits rotating flow and curved walls, commonly encountered in process-intensified devices, such as a spinning disc reactor. The solver is used to perform numerical experiments using higher-order discretizations with up to 716M degrees of freedom. Results for the enstrophy, kinetic energy, and vorticity and Q criterion distributions are evaluated against computational data available in the literature, obtaining accurate results. Moreover, new data concerning the numerical dissipation introduced by the scheme is provided. The results highlight the capabilities of higher-order methods in the context of ILES to obtain less numerical dissipation at lower computational cost and raise questions in terms of what is an acceptable amount of numerical dissipation for industrial flows.

The GMG preconditioner, essential for the matrix-free implementation, is further studied in the context of local mesh refinement. Two variants of GMG, local smoothing and global coarsening, are compared using a flow around a sphere and a Taylor–Couette flow with a dynamic and a local mesh refinement, respectively. The comparison is made in terms of the number of iterations, the total costs of the GMG algorithm, the computational cost of each of the individual building blocks, and the scalability. This study presents the first comparison of both of these variants in the context of a monolithic matrix-free stabilized solver for the Navier–Stokes equations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

Symbols

$\boldsymbol{g}$	Gravitational acceleration
ρ	Density
μ	Dynamic viscosity
$\boldsymbol{u}$	Velocity, largest eddies velocity
p^*	Reduced pressure
P	Pressure
$\boldsymbol{f}$	Source term
ν	Kinematic viscosity
Ma	Mach number
c	Speed of sound
Re	Reynolds number
l	Characteristic length, integral scale
t	Time
$\boldsymbol{u}'$	Velocity fluctuation, subgrid scale velocity
$\bar{\boldsymbol{u}}$	Mean average velocity, filtered velocity
p'	Pressure fluctuation, subgrid scale pressure
$\bar{p}$	Mean average pressure, filtered pressure
η	Kolmogorov microscale, radius ratio between inner and outer cylinder
$\boldsymbol{v}$	Smallest eddies velocity, vector test function
q	Scalar test function
Ω	Lipschitz domain
$\partial\Omega$	Boundary of domain Ω
k	Order of accuracy, number of elements
Ω_k	Domain of an element
$\boldsymbol{n}$	Outward-pointing normal
D	Dirichlet boundary condition
N	Neumann boundary condition
e	Solution error
h	Mesh size, spatial element length
τ, τ_u	Stabilization parameter
R_{m}	Strong residual of the momentum equation
R_{c}	Strong residual of the continuity equation

Δt	Time step
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$	Block matrices
$\delta \boldsymbol{U}$	Newton correction for the velocity
$\delta \boldsymbol{P}$	Newton correction for the pressure
$\boldsymbol{R}_u$	Residual terms related to velocity
$\boldsymbol{R}_p$	Residual terms related to pressure
α	Step length for line-search
$\boldsymbol{L}$	Lower triangular matrix
$\boldsymbol{D}$	Diagonal matrix
$\boldsymbol{U}$	Upper diagonal matrix
$\boldsymbol{S}$	Schur complement
$\mathcal{A}$	Operator
$\boldsymbol{x}, \boldsymbol{b}$	Vector
$\boldsymbol{R}_k, \boldsymbol{R}_k^\top$	Gather and scatter operators
$\boldsymbol{S}_k, \boldsymbol{S}_k^\top$	Operator to transform from local DoFs to quadrature points
$\boldsymbol{Q}_k$	Operator with integration weights and Jacobian
$\boldsymbol{J}_k$	Jacobian
ω_q	Quadrature weights
$\boldsymbol{\xi}$	Reference element coordinates
$\boldsymbol{\phi}$	Vector basis functions
ψ	Scalar basis functions
$\mathcal{P}_{(l-1)}^{l}$	Prolongation operator from level $l-1$ to level l
$\mathcal{R}_{l}^{(l-1)}$	Restriction operator from level l to level $l-1$
$\boldsymbol{r}^{(l)}$	Residual level l
ω	Smoother relaxation parameter
$\boldsymbol{P}$	Preconditioner
$\lambda_{\min}$	Minimum eigenvalue
$\lambda_{\max}$	Maximum eigenvalue
β	Spatially independent pressure gradient
$\boldsymbol{u}_B$	Bulk velocity
$\boldsymbol{\tau}$	Deviatoric stress tensor
Δy^+	Dimensionless distnace from the wall
y_{cc}	Half of distance from the wall to the wall-nearest grid point
u_τ	Friction velocity
τ_w	Wall-shear stress
c	Constant

T_f	Number of flows through times
$\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{Q}$	Function spaces
$\boldsymbol{u}_h$	Discretized velocity
p_h	Discretized pressure
h_{conv}	Size of an element related to convective transport
h_{diff}	Size of an element related to diffusive transport
h_k	Volume of a cell
p	Element order
Pe_h	Element Peclet number
ℓ	Refinement level
Q_p	Hexahedral element of order p
N_T	Number of transient iterations
N_N	Number of Newton iterations
$\overline{N}_N$	Average Newton iterations per transient iteration
$\overline{N}_L$	Average linear solver iterations per Newton iteration
ω	Vorticity
E_k	Kinetic energy
ξ	Enstrophy
T	Total time multiplied by number of nodes
r_i	Inner cylinder radius
r_o	Outer cylinder radius
U_i	Inner cylinder wall velocity
L	Length of the cylinders
d	Difference in the outer and inner cylinder radius
Γ	Aspect ratio
ω_i	Angular velocity of inner cylinder
Ta	Taylor number
Ta_crit	Critical Taylor number
ϵ_N	Normalized numerical dissipation

Acronyms

AMG	Algebraic Multigrid
ASM	Additive Schwarz Method
BDF2	Second-order Implicit Backward Differentiation
BiCGStab	Biconjugate Gradient stabilized method
CG	Continuous Galerkin

CFD	Computational Fluid Dynamics
CFL	Courant-Friedrics-Lewy
DD	Domain Decomposition
DEM	Discrete Element Method
DG	Discontinuous Galerkin
DNS	Direct Numerical Simulation
DoFs	Degrees of Freedom
FDM	Finite Difference Method
FVM	Finite Volume Method
FEM	Finite Element Method
GC	Global Coarsening
GLS	Galerkin Least-Squares
GMG	Geometric Multigrid
GMRES	Generalized Minimal Residual Method
GPU	Graphics Processing Unit
ID	Inverse Diagonal
ILES	Implicit Large-Eddy Simulation
ILU	Incomplete LU
IRK	Implicit Runge-Kutta
LBB	Ladyzhenskaya-Babuska-Brezzi
LBM	Lattice-Boltzmann Method
LES	Large-Eddy Simulation
LS	Local Smoothing
LSIC	Least-Squares Incompressibility Constraint
LSC	Least-Squares Commuters
MB	Matrix-based
MF	Matrix-free
MMS	Method of Manufactured Solutions
MPI	Message Passing Interface
PCD	Pressure-Convection Diffusion
PDE	Partial Differential Equation
PI	Process Intensification
PMG	p-Multigrid
PSPG	Pressure-Stabilizing Petrov–Galerkin
RANS	Reynolds-Averaged Navier–Stokes
SDR	Spinning Disc Reactor

SEM	Spectral Element Method
SGS	Subgrid-Scale
SI	Smoothing Iterations
SIMD	Single Instruction Multiple Data
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SPH	Smoothed Particle Hydrodynamics
SUPG	Streamline-Upwind Petrov–Galerkin
VANS	Volume-Averaged Navier–Stokes
VMS	Variational Multiscale Methods

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

The need for understanding the behavior of all the fluids that surround us has been present since prehistory. The study of the movement of fluids, both liquids and gases, was essential for the development of ancient civilizations (e.g., water distribution, irrigation, design of maritime transportation) and continues to be relevant to humankind today. Nowadays, advances in fluid dynamics are directly linked to global problems such as changing meteorological conditions, the availability of resources, and the growing population. Researchers and scientists around the world dedicate their lives to find strategies that can solve or mitigate these global problems in different domains. This thesis is one of these efforts in the field of process engineering; it aims to improve the Computational Fluid Dynamics **(CFD)** aspect needed to design more environmentally friendly devices used in a variety of chemical processes.

Several efforts to design innovative and compact devices for the process industry that improve efficiency and safety while reducing environmental impact have been made in the last 70 years in the context of Process Intensification **(PI)**. Some recent concrete examples of PI are: i) the invention of reactive distillation, which combines a chemical reaction reactor and distillation into one operation, reducing capital investment and saving energy [1], ii) rotating packed bed reactors for heterogeneous catalytic reactions that aim to increase the efficiency of gas-liquid mass transfer by reducing the size of the reactors and improving its functionality [2], and iii) microchannel reactors for reforming and CO_2 capture technologies that provide solutions to decrease greenhouse gas emissions [3].

Most of these devices have been developed through two main pillars of science: experiments and theory. However, maximizing mass, momentum, and heat transfer while reducing the size of devices or introducing complex geometries makes experimental methods challenging to fully investigate the characteristics of the flow [2]. Therefore, PI can benefit from advanced CFD models as the third pillar in the design pipeline. CFD cannot only give insight into the behavior of the flow when the device has a complex geometry but can also help narrowing down alternative designs to some promising ones that are worth further studying and validating through experimental approaches.

The basis of the development of CFD models are the Navier–Stokes equations that describe fluid flow. They are a set of Partial Differential Equations **(PDEs)** developed by different scientists over several decades in the 1800s, which we have been trying to solve ever since. It was only until the 1960s that the rise of computers allowed numerical solutions of PDEs;

this field, known as scientific computing, quickly reached a relevant role in the design of engineered systems [4]. The first industries to actively involve CFD in their industrial developments were the aerospace and military industries, leading to significant advances in these fields [4]. However, nowadays the need for CFD in other domains is imminent, and further developments, for example, in the usage of CFD for the design of devices used in chemical processes, are needed.

Solving the Navier–Stokes equations is a complex task that requires knowledge in physics, mathematics, and computer science. From a physics point of view, the behavior of a fluid flow is highly dependent on the application; in process-intensified devices, it often encompasses turbulent flow, interaction between multiple physics and complex geometries. From a mathematics perspective, several resolution strategies have been developed in the past; however, it is still an active research field since their efficiency is in practice directly linked to computer science, where the rapid development of computer systems in the last decades required to reimplement existing solvers and to tailor them to modern computer architectures (many core systems and Graphics Processing Unit **(GPUs)**) to be able to effectively utilize the resources available today.

This work initially aimed to study variational multiscale methods, also known as advanced stabilization techniques, in the context of a (high-order) finite element method for turbulence modeling when simulating practical flows in chemical processes. However, the focus was shifted to improving the efficiency of the Navier–Stokes solver, since we realized that the research was bounded by the computational resources (memory and time) when dealing with complex turbulent benchmarks. As a consequence, the overall aim of this work became to develop an efficient Navier–Stokes solver suitable to efficiently simulate turbulent flows encountered in process-intensified devices.

This thesis is organized as follows. In Chapter 2, a comprehensive review of the literature and the state of the art regarding the usage of CFD in process intensification, the modeling of turbulent flows, and implementation trends is given. Chapter 3 summarizes the knowledge gaps, the research question, and objectives, and gives an overview of the structure of the thesis. In Chapter 4, important aspects of the methodology of this work are discussed. The main contributions of this thesis are presented in the form of publications in Chapters 5, 6, and 7. Further unpublished results are presented in Chapter 8. Chapter 10 discusses the scientific results and concludes this thesis by summarizing the contributions and impact of the work, stating its limitations and pointing to future research directions.

CHAPTER 2 LITERATURE REVIEW

This chapter presents a review of the literature on the topics relevant to this work. Section 2.1 presents the state-of-the-art and challenges of the usage of CFD in the field of PI, as well as a concrete example of a process intensified device: a spinning disc reactor. Section 2.2 presents basic concepts of fluid mechanics, such as the governing equations and the physics of turbulent flow. Section 2.3 discusses different numerical methods and resolution strategies available to numerically solve the Navier–Stokes equations for turbulent flow; the main focus is on the Finite Element Method **(FEM)**, which is the method chosen for this thesis. Section 2.4 reviews the trends in the implementation of such solvers with details on hardware-aware matrix-free computations and efficient linear solvers.

2.1 Computational Fluid Dynamics in Process Intensification

According to Boffito et al. [3], PI can be defined as an approach that offers strategies to design more efficient equipments by enhancing heat and mass transfer without negatively affecting the environment, investment and process costs. This is the reason why PI plays an important role in achieving sustainability in the chemical industry as part of the Sustainable Development Goals of the United Nations [5]. PI touches on a broad range of technologies used in today's industries including but not limited to: static mixers, HiGee reactors, microreactors, reactive distillation, spinning disc reactors and microwave-assisted chemistry [3]. Recent publications discuss the role of CFD modeling and artificial intelligence in PI in general; see, for example, the work by Tula et al. [6] and López Guajardo et al. [7]. In these general reviews, special emphasis is placed on the importance of combining experimental data with advanced visual aids, CFD, and machine learning to advance the design of process-intensified technologies.

In the literature, there are review articles with the state of the art of CFD simulations for specific processes or devices; for example: on biomass pyrolisis by Xiong et al. [8], on liquid biofuel production processes by Quiroz-Pérez et al. [9], on rotating packed bed reactors by Ghadyanlou et al. [10], on photocatalytic processes by Oliveira de Brito Lira et al. [11] and on photobioreactors by Ranganathan et al. [12]. There are two main challenges of current CFD models that are recurrent among those publications: i) the inability to account for all the physics involved in the process intensified devices (flow, turbulence, energy and mass transfer, and chemical reactions), and ii) the limitation in terms of computational resources (memory and time) needed to accurately simulate the devices. In the case of turbulence modeling, what particularly stands out are simulations without turbulence modeling (assuming laminar flow) or the preference for Reynolds-Averaged Navier–Stokes **(RANS)** approaches (Reynolds stress model, κ-ω, κ-ϵ) with only very few studies using Large-Eddy Simulation **(LES)**.

The first publications on process intensification date back to the 1930s and are patents that focus on HiGee (high-gravity) forces generated by rotation [3]. The reason for this is that equipment designed to operate at high speed rotation helps to overcome limitations imposed by gravity and to improve separation and micromixing [3, 13]. Since then, this kind of technology has been widely applied for different processes; see recent comprehensive reviews by Lu et al. [14] and Boodhoo et al. [13]. Some well-known specific examples of Hi-Gee technologies include spinning disc reactors, rotor-stator reactors, and rotating packed beds. These technologies generate centrifugal forces by rotating an enclosed surface, a flat surface, or an inclined surface. Under these conditions, liquids form thin films or small droplets in the gas-liquid and liquid-liquid processes that enhance heat and mass transfer due to the enlarged surface area per unit volume [13]. In what follows, a more detailed review of one of these technologies is presented in order to understand the scales at which these devices operate and the challenges from a hydrodynamics perspective.

Spinning Disc Reactors

In a Spinning Disc Reactor **(SDR)**, high gravity fields of the order of $10-1000\boldsymbol{g}$ are created by rotating a flat disc that can have a rough or grooved surface. A continuous stream of liquid is impinged directly on a horizontal surface forming a very thin film, typically of around $30-500$ µm and surface areas as high as 30 000 $\mathrm{m^2/m^3}$ [15]. The intensification in the SDR is related to the short diffusional path length offered by the thin film and the induced turbulence of surface waves, which leads to enhanced mixing, heat and mass transfer. A schematic representative of an SDR is given in **Figure 2.1**. Due to short residence times (ranging from milliseconds to seconds), these devices are particularly suitable for fast reactions that benefit from precise control in terms of residence time, fast mixing of reactants, fast heating or cooling, and immobilized catalysts [13,14]. Their usefulness has been proven for nanoparticle precipitation, crystallization, polymer production, pharmaceutical product synthesis, and photocatalytic applications [3,13].

Several investigations have been performed using these devices to better understand topics such as the film flow trajectory, the micromixing, the residence-time distribution, and the heat transfer efficiency. Thin flow hydrodynamics has traditionally been studied via advanced experimental techniques, such as high-speed video imaging. These experiments have allowed scientists to observe 2D and 3D waves, and spiral flow paths under different operating condi-

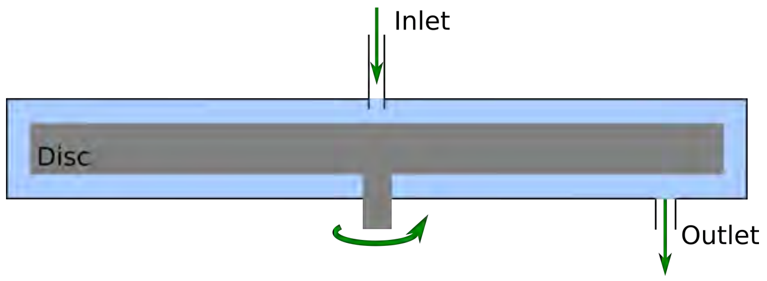

Figure 2.1 Schematic of a spinning disc reactor and thin-film flow around the disc.

tions [15]. However, there is still a lack of appropriate online/inline monitoring tools, such as miniaturized probes and sensors that allow one to measure specific variables of interest [15]. This makes further design and testing of the SDR devices challenging, especially, since further innovations include the addition of multiple smaller discs on a single shaft that allow for a longer residence time while reducing energy consumption, or the design of discs with new geometries and surfaces that improve the flow and mixing of the film [15].

Most of the studies in the literature are experimental; however, some numerical studies performed in the past decade that focus on hydrodynamics can be found. Most notably, the one by Kleiner et al. [16] and the one by Hop et al. [17]. The former carried out a single-phase CFD study to better understand the hydrodynamics and heat transfer of an SDR. They used OpenFOAM, a finite volume open-source software, to simulate the flow through a two-dimensional geometry of an SDR, using a RANS turbulence modeling approach. They obtained results for the radial and tangential velocity and temperature profiles that match experimental results from the literature. In the latter case, a three-dimensional SDR was simulated using OpenFOAM and an LES simulation approach. They validated residence-time distributions within such a device, obtaining good agreement with experimental results for a range of rotational Reynolds numbers.

Other recent studies focus on other aspects of SDR. For example, Wang et al. [18] used the Volume of Fluid **(VOF)** method of the ANSYS workbench and a κ-ϵ model (also a RANS approach) to understand the behavior of liquid flow. Mazzei et al. [19] focused on the study of the heat transfer capabilities of an SDR using ANSYS Fluent and a κ-ϵ turbulence model for single-phase and two-phase boiling conditions. All of the previous studies agree that CFD is an essential tool to optimize the design and scaling up of SDRs. However, a lack of CFD studies with advanced turbulence modeling techniques of these devices is observed.

2.2 Fluid Mechanics

The mathematical model that concerns this work and describes fluid flow is given by the Navier–Stokes equations. For an incompressible flow with density ρ and dynamic viscosity μ, its velocity $\boldsymbol{u}$ and reduced pressure $p^* = P/\rho$ can be described by the following Partial Differential Equation **(PDE)** system:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.1}$$

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} = \boldsymbol{f}, \tag{2.2}$$

where $\boldsymbol{f}$ is a source term (i.e., gravity) and $\nu = \mu/\rho$ is the kinematic viscosity. This PDE system is composed of a continuity equation and a momentum equation that account for the conservation of mass and momentum, respectively. The Navier–Stokes equations have been known for decades, and considerable effort has been put into solving them analytically in three dimensions without success. The viscous term represents the momentum by diffusion. The nonlinear quadratic term in $\boldsymbol{u}$ represents the momentum by convection and is the main source of difficulties when solving this PDE system numerically. One of the main assumptions of this model is that the flow is incompressible, i.e., the changes of density of the fluid are considered negligible and the information propagates instantaneously across the domain. To characterize a flow as incompressible the Mach number is used:

$$\mathrm{Ma} = \frac{\|\boldsymbol{u}\|}{c}, \tag{2.3}$$

with $\boldsymbol{u}$ the velocity of the flow and c the speed of sound. According to Gravemeier [20], a flow can be considered incompressible if $\mathrm{Ma} \leq 0.3$. This makes the scope of incompressible flows very wide. Another important dimensionless number in fluid mechanics is the Reynolds number:

$$\mathrm{Re} = \frac{\text{Inertial forces}}{\text{Viscous forces}} = \frac{\rho\|\boldsymbol{u}\|l}{\mu} = \frac{\|\boldsymbol{u}\|l}{\nu}, \tag{2.4}$$

where l is the characteristic length of the geometry (e.g., diameter of a pipe) and $\|\boldsymbol{u}\|$ the mean flow velocity. It was defined by Reynolds in 1883 [21], when he observed through experiments in a straight smooth path that there was a transition from a well-ordered flow, known as laminar flow, to a more chaotic flow, known as turbulent flow. If the Reynolds number is low, the flow is laminar and is driven by viscous forces, whereas if the Reynolds number is high, viscous forces are small and the regime is considered turbulent. In our daily life (e.g., cigarette smoke, stirring a coffee cup) and in most relevant engineering applications (e.g., stirred reactor, combustion furnaces, pipe flow) turbulent flows are encountered [20].

2.2.1 Turbulent Flow

The transition between laminar and turbulent flow is complex, and there are at least two types of transition to turbulence. There are flows where turbulence appears in the form of small patches of chaotic motion, which then grow and merge until a fully developed turbulence state is achieved [21]. This is the usual transition in the boundary layers and pipes. The second type of transition is given by flows in which turbulence develops uniformly throughout the fluid due to an instability in the flow that eventually leads to a complicated laminar flow that breaks up into more complex structures [21].

Although it is difficult to define turbulence, Davidson [21] attempted to do so as follows: "... when ν is sufficiently small, all flows develop a random, chaotic component of motion. We group these complex flows together, call them turbulence, and then note some of their common characteristics". These main characteristics of turbulence flow are summarized as [21]:

i) the velocity field fluctuates randomly in time and is highly disordered in space, leading to a wide range of length scales;

ii) the velocity field is unpredictable; i.e., a minor change to the initial conditions has a large effect on the motion of the fluid.

In fact, turbulence was characterized experimentally by setting probes along the flow and measuring the velocity at different points [21]. This allowed scientists to identify that instantaneous values of the velocity were extremely sensitive to any change in initial or boundary conditions, which led to different results in every experiment. However, the statistical averages did not show this sensitivity, leading to a possible robust mathematical description of the flow through them. This was the beginning of numerical modeling of turbulence and the importance of statistical quantities when describing this kind of flow [20].

2.2.2 The Different Scales and the Energy Cascade

Following the idea presented in the previous section, we can define at each location in a steady-on-average flow the following velocity decomposition:

$$\boldsymbol{u}(\boldsymbol{x},t) = \bar{\boldsymbol{u}}(x) + \boldsymbol{u}'(\boldsymbol{x},t), \tag{2.5}$$

where $\boldsymbol{u}$ is the velocity, $\boldsymbol{u}'$ is the random component of the motion that represents a fluctuation, and the mean average velocity is given by $\bar{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}'$ [21]. Note that the latter is a smooth ordered function of position only. In any instant, $\boldsymbol{u}$ consists of a collection of eddies (also known as vortices) of different sizes. There are small and large eddies, where the latter can have a size on the order of the characteristic length scale of the mean flow (e.g., pipe diameter). The nature of these eddies is linked to vorticity. In fact, another definition of turbulence states that turbulence is an array of vortex tubes and sheets that are separated by fluid that is irrotational or displays some vorticity [22].

Inside and between the eddies, the fluid layers are in relative motion, and local viscous stresses cause energy dissipation. The energy dissipation across these structures is known as the Richardson energy cascade [21]. The energy cascade states that the largest eddies are subjected to inertial instabilities that cause them to break up into smaller vortices. Then these smaller vortices go through the same process, passing their energy to yet smaller eddies. This entire process is driven by inertial forces, i.e., the viscous forces do not play a role. The energy cascade ends when the eddy size is so small that the Reynolds number is equal to one. At this point, viscous forces become more important, and viscous dissipation enters the whole picture. In **Figure 2.2**, a schematic of the so-called eddies is presented along with their breakdown and energy dissipation.

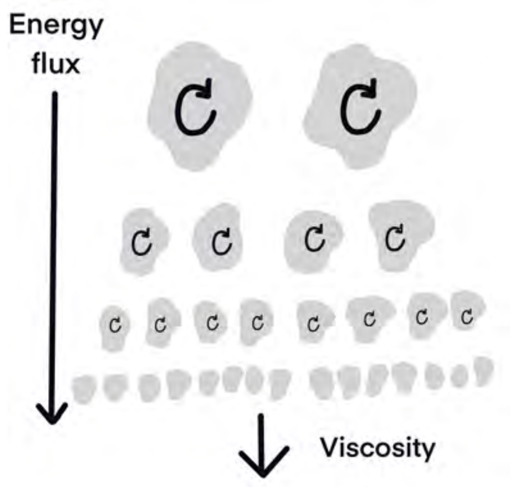

Figure 2.2 Schematic of the energy cascade with large and small eddies.

One of the most important questions in turbulence is how to know exactly the range of sizes of those eddies, as it determines the range of scales that characterize the flow. The most important theoretical estimate of the range of length scale in turbulent flow is given by the following expression:

$$\frac{l}{\eta} \sim \mathrm{Re}_l^{3/4}, \tag{2.6}$$

where l represents the integral scale, η is known as the Kolmogorov microscale of turbulence and Re_l is they Reynolds number based on l and the relative velocity measured, $\boldsymbol{u}$, which is related to the largest eddies [21]. If we consider a flow with a Reynolds number equal to $\mathrm{Re}_l = 10\,000$ and a characteristic length of $l = 1$ cm, the Kolmogorov scale is equal to 0.001 cm, i.e., the smallest scale is 1000 times smaller than the largest scale in this flow. If the Reynolds number is increased, the difference keeps growing.

Another important estimate is the turnover time that represents the lifespan of an eddy:

$$t_{\text{large eddy}} = \frac{l}{\boldsymbol{u}}, \quad t_{\text{small eddy}} = \frac{\eta}{\boldsymbol{v}}, \tag{2.7}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ represent the typical velocity of the largest and smallest eddies, respectively [21]. This estimate indicates that there is a wide range of time scales as well. Therefore, turbulence is characterized as a multiscale phenomenon in both space and time. This is, in fact, one of the main challenges when trying to simulate flows with this regime on a computer, as it represents a very time and memory consuming calculation [21].

2.3 Numerical solution

The solution of the governing equations for fluid flow is a challenging task in practical settings. Analytical solutions exist only for simple two-dimensional problems. Experiments have also been carried out for more complex settings and are efficient ways of obtaining reliable data, but also have limitations. However, both simple analytical solutions and experimental results are not enough to fully understand fluid flow, and CFD has become an essential tool for understanding more complex physical phenomena. This section is dedicated to the CFD aspects relevant to this work. It starts by reviewing available strategies to model turbulence, followed by the discussion of discretization methods and resolution strategies to solve the incompressible Navier–Stokes equations with a special focus on FEM.

2.3.1 Turbulence Modeling

Turbulence modeling is an active research domain that aims to develop approximations to the Navier–Stokes equations that encompass sufficient dynamical information for a reliable and efficient prediction of the physical phenomena of turbulence. The main approaches to model turbulence are presented in **Figure 2.3**.

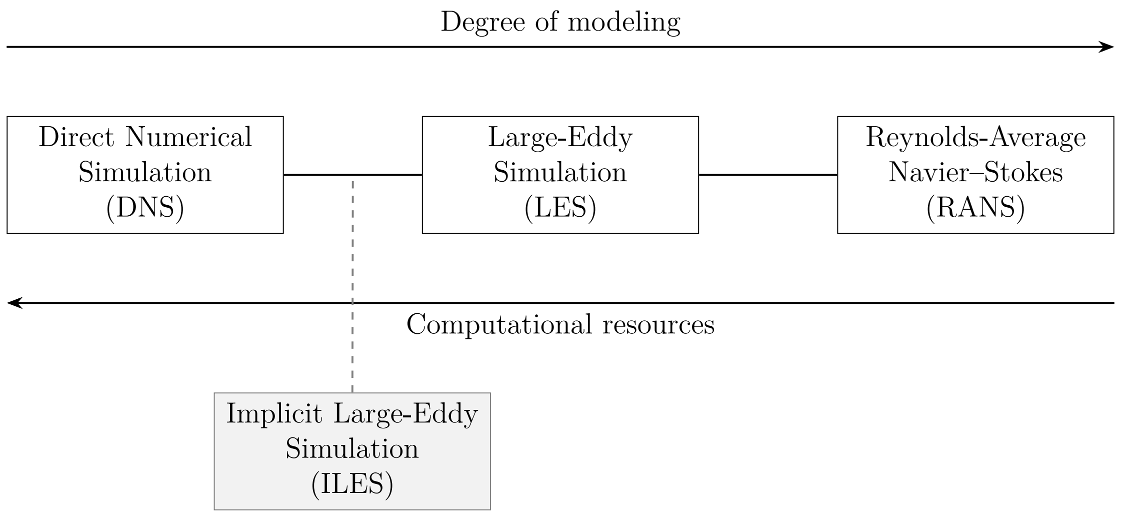
Degree of modeling
Direct Numerical Simulation (DNS)
Large-Eddy Simulation (LES)
Reynolds-Average Navier–Stokes (RANS)
Computational resources
Implicit Large-Eddy Simulation (ILES)

Figure 2.3 Classification of the main turbulence modeling approaches available today.

The Direct Numerical Simulation **(DNS)** approach solves the full Navier–Stokes equations numerically for all length and time scales; it gives the most accurate representation of the flow at a high computational cost. Therefore, it is useful to understand small problems with simple geometries and low Reynolds numbers and to obtain benchmark data to assess the accuracy of other approaches. Its applicability, especially when it comes to industrial practical flows, is very limited even with the exascale computers available today.

The approach with the highest degree of modeling is RANS. It solves a version of the Navier–Stokes equations that considers statistical means of the flow obtained by inserting the Reynolds decomposition (see Appendix A for a detailed derivation). Then, it solves for averaged flow quantities over time (e.g., $\bar{\boldsymbol{u}}$) and requires one to model new terms introduced by averaging known as the Reynolds stresses ($-\rho\overline{\boldsymbol{u}' \otimes \boldsymbol{u}'}$). Typical models include the κ-ϵ model, the κ-ω model, and the Reynolds stress model. It has the lowest computational requirement and, therefore, is very popular in industry. Despite this, RANS approaches are known to have problems with transient flow structures, which are encountered in all turbulent flows, and which are essential when modeling industrial devices [23, 24].

The third approach, which is usually placed between RANS and DNS in terms of degree of modeling and computational resources, is the Large-Eddy Simulation (LES) approach. It solves the filtered form of the Navier–Stokes equations for the large scales of the flow, while the small scales and their effect on the large scales are modeled using Subgrid-Scale **(SGS)** models (see Appendix A for the derivation of the filtered form). The main assumptions of LES are that large eddies are responsible for most of the transport of momentum, energy, and passive scalars, whereas smaller eddies are more universal in their behavior [25]. In LES, most of the effort is focused on the development of suitable SGS models. The first model developed for this task was the Smagorinsky model, followed by the dynamic Smagorinsky model [24, 26]. Since then many other SGS models have been developed in order to improve the approximation. These advanced models are complex to implement and require more computational effort, which has limited their popularity [25]. Therefore, other subcategories of LES have been developed, such as Implicit Large-Eddy Simulation **(ILES)**, also known as Monotonic-Integrated LES (MILES), where an implicit filtering that depends on the discretization is applied, very LES (VLES) and hybrid LES/RANS approaches, such as Detached Eddy Simulation (DES) [24].

The ILES approach has gained popularity over the last two decades. It is commonly referred to as underresolved DNS in the literature, which means that it can be placed between the DNS and LES approaches regarding the degree of modeling and the computational resources. It does not contain a subgrid model and in most of the cases the refinement of the mesh determines the length scales that are resolved; therefore, it is often called a "parameter-free" approach. It originates in the boundaries between physical modeling and numerical discretization [25]. Therefore, its detailed description is highly dependent on the numerical method chosen for the discretization of space and time of the Navier–Stokes equations; however, the general idea is that when the cell size is not small enough to resolve all the eddies up to the scale where there is viscous dissipation, which is usually the case, artificial numerical dissipation is included. A comprehensive review of this approach in the context of the finite-volume method can be found in [25]. In the context of FEM, this approach of modeling turbulence is encompassed by the stabilization techniques domain; for a recent review, see [27]. Although several publications using ILES approaches can be found in the literature, there is a need of further studying them, specifically by comparing them to explicit LES approaches in order to identify the most promising techniques [27].

2.3.2 Discretization in Space using the Finite Element Method

FEM is only one of the possible discretization methods that are commonly used for the numerical resolution of the Navier–Stokes equations. There are other traditional methods, such as the Finite Difference Method **(FDM)** and the Finite Volume Method **(FVM)**, and newer methods, which include the Lattice-Boltzmann Method **(LBM)** and Smoothed Particle Hydrodynamics **(SPH)**. FEM was initially developed for simulating structural mechanics problems and started to be used for the solution of Stokes and Navier–Stokes equations in the 1970s [28]. Since then, the development of this method in this context has advanced greatly, from numerical analysis to important aspects concerning numerical linear algebra and high-performance computations.

As with all other traditional methods, FEM gives an approximation to the original problem and eventually leads to a system of matrix equations. The basis of FEM is to find the weak derivative or variational formulation of the problem using a weighted residual approach (for a detailed explanation, see, for example, the book by Elman et al. [29]). The first step to find this formulation is to define a bounded Lipschitz domain Ω with boundary $\partial\Omega$ and divide it into elements Ω_k. For the Navier–Stokes equations, we require that for an appropriate set of test functions, a scalar test function q for the continuity equation and a vector test function $\boldsymbol{v}$ for the momentum equation, the following holds:

$$\int_\Omega q(\nabla \cdot \boldsymbol{u}) = 0, \tag{2.8}$$

$$\int_\Omega \boldsymbol{v} \cdot (\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} - \boldsymbol{f}) = 0, \tag{2.9}$$

for all q and $\boldsymbol{v}$ in a suitably chosen space. If $\boldsymbol{u}$ and p are a classical solution, then they must also satisfy this weak formulation. If $(\boldsymbol{v}, q)$ is sufficiently smooth, then the smoothness required for the weak solution $(\boldsymbol{u}, p)$ can also be reduced by using integration by parts for the diffusive term and the pressure term:

$$\int_\Omega \boldsymbol{v} \cdot \nabla p^* = -\int_\Omega p^* \nabla \cdot \boldsymbol{v} + \int_{\partial\Omega} p^* \boldsymbol{n} \cdot \boldsymbol{v}, \tag{2.10}$$

$$-\nu \int_\Omega \boldsymbol{v} \cdot \Delta \boldsymbol{u} = \nu \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - \nu \int_{\partial\Omega} (\boldsymbol{n} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v}, \tag{2.11}$$

where $\boldsymbol{n}$ denotes the outward-pointing normal to the boundary and $\nabla \boldsymbol{u} : \nabla \boldsymbol{v}$ is a component-wise scalar product. Rewriting all expressions, the final variational formulation reads:

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

$$\int_\Omega \boldsymbol{v} \cdot (\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \boldsymbol{f}) + \nu \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - \int_\Omega p^* \nabla \cdot \boldsymbol{v} - \int_{\partial\Omega} (\nu(\boldsymbol{n} \cdot \nabla \boldsymbol{u}) - p^* \boldsymbol{n}) \cdot \boldsymbol{v} = 0, \tag{2.13}$$

for all suitable set of test functions $(\boldsymbol{v}, q)$. The suitable spaces for the solution and the test spaces are given as:

$$\begin{aligned}
\mathcal{U} &:= \{\boldsymbol{u} \in H^1(\Omega)^d, \boldsymbol{u} = \boldsymbol{w} \;\text{ on } \partial\Omega\},\\
\mathcal{V} &:= \{\boldsymbol{v} \in H^1(\Omega)^d, \boldsymbol{v} = 0 \;\text{ on } \partial\Omega\},\\
\mathcal{P} &:= \{p^* \in L^2(\Omega)\},\\
\mathcal{Q} &:= \{q \;\in L^2(\Omega)\},
\end{aligned}$$

where $H^1(\Omega)$ is the Sobolev space of square-integrable functions and square-integrable first derivatives. Since there are no pressure derivatives, $L_2(\Omega)$ is an appropriate space for the pressure. The same space is chosen for q which ensures that the continuity equation is finite. Furthermore, one must specify boundary conditions. It is common to have the following Dirichlet (D) and Neumann (N) boundary conditions in $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$:

$$\boldsymbol{u} = \boldsymbol{w} \text{ on } \partial\Omega_D, \quad \nu(\boldsymbol{n} \cdot \nabla \boldsymbol{u}) - \boldsymbol{n}p^* = 0 \text{ on } \partial\Omega_N. \tag{2.14}$$

If only Dirichlet boundary conditions are specified, i.e., the velocity is specified everywhere on the boundary, then the pressure solution is only unique up to a hydrostatic constant. However, if flow in a channel needs to be modeled, then a Dirichlet outflow boundary condition with $\boldsymbol{w} \cdot \boldsymbol{n} > 0$ leads to difficulties in the outflow and a Neumann outflow boundary condition, also known as the zero traction boundary condition, is necessary.

An important aspect of FEM is the choice of finite-dimensional velocity and pressure subspaces. In the context of the Navier–Stokes, it is common to represent the solution of the velocity and the pressure by a polynomial approximation. However, the combination of the polynomial approximation must be inf-sup stable. Inf-sup instability is related to the Ladyzhenskaya-Babuska-Brezzi **(LBB)** condition which states that the velocity space needs to be rich enough to exclude spurious pressure modes from the numerical solution [30]. In practical terms, this means that to achieve an inf-sup stable discretization one must choose mixed-order polynomials of degree p for the velocity and degree $p-1$ for the pressure, also referred to as Taylor–Hood velocity-pressure pair (e.g., Q_pQ_{p-1} for hexahedral elements).

The choice of test and solution spaces leads to different variants of FEM. A classic choice is to choose the same space for both, which leads to Galerkin FEM. If the basis for this space is globally continuous over the whole domain, we obtain the classical Continuous Galerkin **(CG)** formulation. On the other hand, if the bases are local, i.e., discontinuities between elements are allowed, this gives rise to Discontinuous Galerkin **(DG)** formulation, which is usually considered a mix between FVM and CG (note that additional boundary terms arise in the variational formulation to account for the discontinuity between elements). For a graphical representation of both CG and DG on a 1D mesh with four cells, see **Figure 2.4**. Both CG and DG solvers for the incompressible Navier–Stokes equations can be found in the literature. One disadvantage of CG in the case of transient problems is that due to the globally defined functions, an implicit mass matrix is always obtained, regardless of the type of time-stepping scheme used, which requires more effort for its solution [31]. A disadvantage of both CG and DG when it comes to transport problems relies on the fact that the basis functions are symmetric in space; this does not allow to account for the information of flow in specific directions. For CG this issue is resolved using complex stabilization techniques (similar to the notion of upwinding in FDM and FVM), while DG provides more flexibility to deal with such problems through stabilization via fluxes [31].

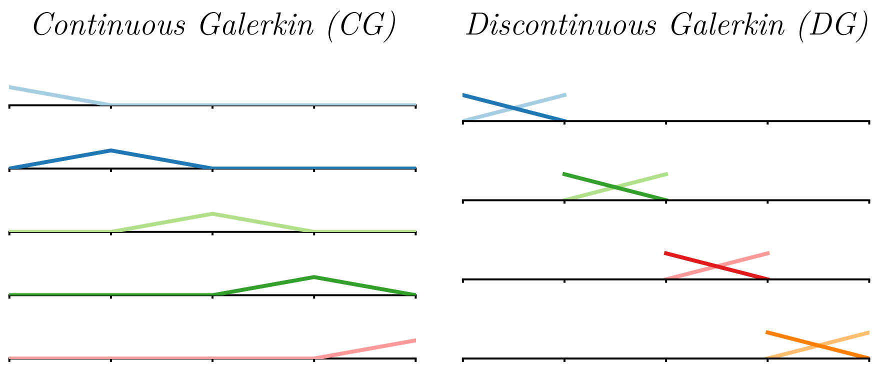

Figure 2.4 Graphical comparison of linear Lagrange elements as basis functions for a Continuous Galerkin (CG) and a Discontinuous Galerkin (DG) formulation in 1D.

High-Order Approximations

One of the main advantages of FEM over other numerical methods is that high-order approximations are easy to realize by increasing the order of the basis for the FE spaces on unstructured meshes. The idea of high-order approximations was first presented in the late 1970s in the context of spectral methods by Gottlieb and Orszag [32]. Since then, high-order methods have become popular because they can deliver higher accuracy with a lower cost than low-order methods [33, 34].

According to Wang et al. [34], a high-order method in the context of CFD is that where the solution error e is proportional to the mesh size h to the power of k ($e = O(h^k)$) with $k \geq 3$. Most of the CFD codes (using either FVM, FDM or FEM) used in industry are second-order accurate ($k = 2$), and are very robust since they have been tested and developed for more than four decades [34]. Despite this, there are still practical problems where high accuracy is required, e.g., vortex-dominated flows, whose solution using low-order methods is prohibitively expensive [34].

In the publication by Wang et al. in 2013 [34], the high-order scientific community stated the main challenges of high-order methods at the time for CFD applications. They can be summarized as follows:

i) high-order methods are more complicated than the low-order counterpart;

ii) high-order methods are less robust and slower to converge to steady-state;

iii) high-order methods have a high memory requirement if implicit time stepping is used;

iv) there are no robust high-order mesh generators available.

Almost twelve years later, the field of high-order methods has advanced greatly. It has become more popular in both academia and industry and the challenges have been alleviated.

Significant development of these methods in open-source libraries have increased robustness and now facilitate the ease of implementation of high-order solvers for problems in various fields. A recent publication by Fischer et al. [35], compares the performance of high-order FEM PDE software on modern architectures for large-scale application problems and demonstrates the capability of these libraries. Advances in computer hardware have led to novel implementations, such as matrix-free approaches (discussed in Section 2.4), which render high-order methods more computationally efficient than their low-order counterparts in terms of memory usage and time to solution. Moreover, the field of curvilinear mesh generation is also advancing, opening the doors to the application of these methods to complex geometries [36].

In FEM, the research focus on high-order methods has been on what is called the Spectral Element Method **(SEM)**, a subcategory of CG, and on DG formulations and their variants. Current open-source CFD SEM software include `Neko` [37] and `NekRS` [38]. In the case of DG, examples are `Nektar++` [39] or `ExaDG` [40].

Stabilization Techniques in the Context of CG FEM

The first upwind FEM formulations were proposed for advection-diffusion equations and were based on the idea of modifying the test functions to achieve the upwind effect by weighting more an element upstream than an element downstream. This domain is now known as stabilized FEM techniques; a recent review can be found in the publication by Fehn et al. [27].

A summary of the main methods is provided in **Table 2.1**. The basic idea of stabilization in FEM is to add a term to the weak formulation of the Navier–Stokes equations that adds diffusion (or viscosity) only in the flow direction for each of the elements (similar to the idea of upwinding in FDM and FVM) (see the publication by Brooks and Hughes [41]). The stabilization term takes the form of an integral over an element k that includes a modified test function multiplied by the residual of the momentum or continuity equations to obtain consistent formulations, that is, when the residual of the equations is zero, the scheme does not add any artificial diffusion.

The Streamline-Upwind Petrov–Galerkin **(SUPG)** method was the first stabilization method to be proposed in the literature in the context of the incompressible Navier-Stokes equations by Brooks and Hughes in 1982 [41]. Since the space of test functions no longer coincides with the space of interpolation functions, that is, the test function $\boldsymbol{v}$ is now $(\boldsymbol{u} \cdot \nabla)\boldsymbol{v}$, the method is not symmetric and does not correspond to a Galerkin formulation, but rather to a Petrov–Galerkin one. This method avoids spurious oscillations in the velocity field that appear as the Reynolds number is increased and requires inf-sup stable FE pairs. Later in 1986, the Pressure-Stabilizing Petrov–Galerkin **(PSPG)** stabilization method was proposed by Hughes et al. [43] to circumvent the LBB condition by introducing a pressure-pressure coupling and allowing the use of same order polynomials for the pressure and the velocity. Since then, it is common to see in the literature the use of both SUPG and PSPG stabilization techniques at the same time.

For pairs of velocity and pressure FE spaces that fulfill the LBB condition (such as Taylor-Hood FE pairs), it is common to obtain solutions that violate the conservation of mass. Therefore, another type of stabilization, the Grad-Div stabilization, was introduced by Franca and Hughes in 1988 [44] to enhance mass conservation. A generalization of the previous stabilization techniques was introduced later on as the Galerkin Least-Squares **(GLS)** technique. It considers a modified test function with all the original terms of the momentum equation in addition to the Least-Squares Incompressibility Constraint **(LSIC)** term (equivalent to the Grad-Div term). The main drawback of this formulation is that it comprises a time derivative for the test function when transient problems are solved, which requires a space-time FEM formulation [42].

All the stabilization techniques introduced require a stabilization parameter τ. This parameter is commonly derived by dimensional analysis [27]. In general, the units for the stabilization terms that involve the strong residual of the continuity equation are m^2/s, while those that involve the strong residual of the momentum equation are s [45]. However, there is no consensus on the definition of the stabilization parameter, apart from the fact that

Table 2.1 Summary of the basic stabilization techniques available in the literature for continuous Galerkin approaches [27,28,42]. R_m: strong residual of the momentum equation, R_c: strong residual of the continuity equation.

Method	**Main characteristics and additional terms**
Streamline-Upwind Petrov-Galerkin (SUPG)	Reduces oscillations in the velocity field when simulating flows with high Reynolds numbers: $$\sum_k \int_{\Omega_k} \tau_{\mathrm{SUPG}} (\boldsymbol{u} \cdot \nabla)\boldsymbol{v} \cdot R_\mathrm{m}$$
Pressure-Stabilizing/ Petrov-Galerkin (PSPG)	Allows to use equal-order elements for the pressure and the velocity, i.e, circumvents the inf-sup condition: $$\sum_k \int_{\Omega_k} \tau_{\mathrm{PSPG}} (\nabla q) \cdot R_\mathrm{m}$$
Grad-Div	Improves mass conservation for FE spaces that fulfill the LBB condition: $$\sum_k \int_{\Omega_k} \tau_{\text{GRAD-DIV}} (\nabla \cdot \boldsymbol{v}) R_\mathrm{c}$$
Galerkin-Least Squares (GLS)	Considers a least-squares term based on the momentum equation: $$\sum_k \int_{\Omega_k} \tau_{\mathrm{GLS}} (\partial_t \boldsymbol{v} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{v} + \nabla q - \nu \Delta \boldsymbol{v}) \cdot R_\mathrm{m}$$ and a least-squares term based on the incompressibility constraint (LSIC): $$\sum_k \int_{\Omega_k} \tau_{\mathrm{LSIC}} (\nabla \cdot \boldsymbol{v}) R_\mathrm{c}$$ The LSIC term is identical to the Grad-Div term. The GLS term contains the SUPG and PSPG stabilizations as well as a derivative in time and a diffusive term.

it must vanish when the mesh is refined. This leads to a wide range of definitions in the literature; one widely used definition in the context of SUPG/PSPG FEM is the one defined by Tezduyar in 1992 [46]:

$$\tau = \left[\left(\frac{1}{\Delta t} \right)^2 + \left(\frac{2\|\boldsymbol{u}\|}{h} \right)^2 + 9 \left(\frac{4\nu}{h^2} \right)^2 \right]^{-1/2}, \tag{2.15}$$

where Δt and h are considered the temporal and spatial element lengths, and the first term is simply neglected for steady-state simulations. In the case of the Grad-Div (or LSIC) stabilization term, a common choice is:

$$\tau = \frac{h}{2\|\boldsymbol{u}\|}. \tag{2.16}$$

Apart from the classic stabilization techniques, many other stabilization techniques have been developed in the last decades, which are part of a new category known as Variational Multiscale Methods **(VMS)**. VMS methods were first introduced by Hughes in 1995 [47]. Between 1995 and 2000, research on these methods focused on error estimations [48, 49] and an extension to time-dependent problems [50]. The first publication linking the LES and VMS methods was that by Hughes in 2000 [51], where the variational multiscale method was used for the incompressible isothermal Navier–Stokes equations. In VMS methods, the separation of the scales is done by a decomposition of resolved and unresolved scales by projections into appropriate function spaces [45]. For detailed reviews on VMS methods, see Rasthofer et al. [52], Ahmed et al. [45], Codina et al. [53], and John [30].

Despite the fact that VMS methods have their origin in the FEM domain, several authors state that the mathematical idea of decomposing the scales is applicable to many other physics and therefore they are not limited to FEM, but could, in fact, be applied to FVM and FDM [53]. Some of these methods are the orthogonal residual-based method proposed by Codina in 2002 [54], the residual-based method presented by Bazilevs et al. in 2007 [55], the local projection stabilization methods by Braack and Lube in 2008 [56], the residual-free bubble methods proposed by Gravemeier in 2003 [20], and the algebraic VMS methods first introduced by Gravemeier in 2003 and further studied in [52, 57].

System of Equations

Regardless of the FE spaces chosen, the variational form of the Navier–Stokes equations requires linearization procedures to deal with the non-linearity that arises due to the convective term. Common methods used in the literature are Picard's and Newton's method. The former is based on the idea of a successive approximation by estimation of the weaker nonlinear terms at a previous value of the solution, i.e., for the convective term the following approximation is used $(\boldsymbol{u}^{(n+1)} \cdot \nabla)\boldsymbol{u}^{(n+1)} \approx (\boldsymbol{u}^{(n)} \cdot \nabla)\boldsymbol{u}^{(n+1)}$ for a nonlinear iteration $n+1$. The latter approach finds successive approximations by finding the roots of the nonlinear function using information from the Jacobian of the system, that is, the convective term is linearized as follows $(\boldsymbol{u}^{(n)} \cdot \nabla)\delta\boldsymbol{u}^{(n+1)} + (\delta\boldsymbol{u}^{(n+1)} \cdot \nabla)\boldsymbol{u}^{(n)}$.

Both methods are widely used in the literature and have advantages and disadvantages when used for the Navier–Stokes equation. Newton's method is generally preferred due to its better order of convergence and robustness, however Picard's method is easy to implement and does not require the computation of a Jacobian that can be nontrivial for complex discretized systems, e.g., the Navier–Stokes discretization [58]. In terms of resources needed, Newton's method has larger memory requirements and larger costs of assembling the system and performing vector-matrix products [30]. However, several mechanisms exist to ensure the robustness and efficiency of the Newton iteration, such as a line-search mechanism that ensures an adequate decrease in the residual by shortening the steps or inexact Newton variants, where, for example, the Jacobian matrix can be reused across Newton iterations [59]. For a general discussion of Newton's method in the context of a CG formulation of the Navier–Stokes equations, the reader is referred to the pseudocode presented in Appendix B.

For a CG formulation, once the appropriate FE spaces are chosen, the basis functions for both trial and solution spaces are defined, and the linearization is carried out via a Gateaux derivative (Newton's method), the final linear system that has to be solved at each Newton iteration has the following structure:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^\top & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{U} \\ \delta\boldsymbol{P} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{R}_u \\ \boldsymbol{R}_p \end{bmatrix}, \tag{2.17}$$

where the matrix is the Jacobian of the variational formulation of the Navier–Stokes equations, which can be found via a Gateaux derivative, and $\boldsymbol{R}_u$ and $\boldsymbol{R_p}$ are the corresponding residual terms involving the velocity and pressure, respectively (see Appendix C for the linearization procedure and the exact definition of each of the blocks of the matrix in case of a CG formulation). This linear system is solved to find the Newton corrections for the velocity ($\delta\boldsymbol{U}$) and pressure ($\delta\boldsymbol{P}$), which are used to find the next approximation to the solution according to: $\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \alpha\delta\boldsymbol{U}^n$ and $\boldsymbol{P}^{n+1} = \boldsymbol{P}^n + \alpha\delta\boldsymbol{P}^n$, where α is the step length, determined by a line-search algorithm. The resulting system of equations is a saddle-point problem due to the zero block in the diagonal; such a problem is difficult to solve from a linear algebra perspective (see [60] for a detailed review of the challenges and possible solution methods).

If stabilization techniques are used, the resulting linear system is modified as follows:

$$\begin{bmatrix} \widehat{\boldsymbol{A}} & \widehat{\boldsymbol{B}} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{U} \\ \delta\boldsymbol{P} \end{bmatrix} = - \begin{bmatrix} \widehat{\boldsymbol{R}}_u \\ \widehat{\boldsymbol{R}}_p \end{bmatrix}, \tag{2.18}$$

with additional terms being added to the blocks $\boldsymbol{A}$ and $\boldsymbol{B}$, a new definition of the block $\boldsymbol{C}$ and most notably, a block $\boldsymbol{D} \neq \boldsymbol{0}$ if PSPG stabilization is used as a consequence of the pressure-pressure coupling introduced. Notice that the computation of both the Jacobian and the residual comprise additional terms. For a concrete expression of the terms added to the system when using SUPG and PSPG stabilization, see Appendix D.

2.3.3 Resolution Strategies

There are two main resolution strategies in the literature to solve the resulting system of equations (Eq. (2.17)): i) splitting or segregated methods, which solve in a decoupled way for the velocity and the pressure, and ii) fully coupled or monolithic methods for which the global system of equations is solved for both the velocity and the pressure at the same time. Several general reviews of both methods can be found, e.g., [30, 60, 61]. In the following, a description of each of these strategies is presented.

Segregated Approaches

The main motivation for this strategy is to avoid the solution of saddle-point problems, since it is challenging from a linear algebra perspective; instead it solves a simpler scalar linear system of equations for each component of the velocity and the pressure. This is achieved by decomposing the equations into a convection-diffusion type problem for the velocity and a Poisson problem for the pressure. This is not a trivial task since the pressure and velocity are coupled by the incompressibility constraint; therefore, there are several methods that have been develop to overcome this limitation. Research on these methods started in the late 1960s with the work of Chorin and Temam [62]. There are four main groups that belong to this category: algebraic splitting schemes, pressure-correction schemes, velocity-correction schemes, and consistent splitting schemes. For detailed reviews, see Guermond et al. [63] and Karniadakis et al. [61].

Pressure-correction schemes are time-marching techniques composed of two sub-steps for each time step. In the first sub-step the pressure is treated explicitly or ignored, and in the second sub-step it is corrected by projecting the velocity onto an appropriate space. In a way, the first step accounts for viscous effects and the second one accounts for incompressibility. In contrast, for velocity-correction schemes, the viscous term is treated explicitly or ignored in the first substep, and the velocity is corrected in the second substep. It is important to note that even though the problem is divided into two subproblems that are easier to solve, the inf-sup stability condition still has to be satisfied to obtain an accurate approximation [63]. From the linear algebra perspective, these approaches can take advantage of the vast literature on iterative methods for Poisson problems and advection-diffusion equations, using for example, multigrid methods that have optimal complexity. The main difficulties with these approaches are the boundary conditions and the stability and accuracy in time [30, 64]. This is the main strategy used in several open-source specialized CFD software (e.g., `Neko` [37], `NEK5000` [65], `Nektar++` [39], `NekRS` [38] and `ExaDG` [40]).

Coupled Approaches

Coupled or monolithic approaches solve instead for the velocity and pressure at the same time, i.e, they solve the saddle-point problem as it is obtained after the linearization needed for Newton or Picard's method. In general, the resulting system matrices are non-symmetric and require suitable iterative solvers, e.g., Generalized Minimal Residual Method **(GMRES)**, Biconjugate gradient stabilized method **(BiCGStab)**, to find a solution. The usage of direct solvers in this context is only useful for small two-dimensional problems with less than 500K unknowns [30]. In order to enable rapid convergence, the iterative solvers require suitable preconditioners. The development of preconditioners relies heavily on the structure of the matrix and has been an active research field for more than three decades, for a comprehensive review, see [29, 60].

In this context, one can distinguish between two types of preconditioners: block preconditioners and monolithic preconditioners. Block preconditioners, also known as physics-based preconditioners, separate the linear operator into its physical components and are based on the Schur complement of the system matrix [30, 64, 66]. The 2x2 block system matrix can be factorized into a lower triangular matrix ($\boldsymbol{L}$), a diagonal matrix ($\boldsymbol{D}$) and an upper triangular matrix ($\boldsymbol{U}$) as follows:

$$\begin{bmatrix} \widehat{\boldsymbol{A}} & \widehat{\boldsymbol{B}} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} = \boldsymbol{LDU} = \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{C}\widehat{\boldsymbol{A}}^{-1} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{A}} & 0 \\ 0 & \boldsymbol{S} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \widehat{\boldsymbol{A}}^{-1}\widehat{\boldsymbol{B}} \\ 0 & \boldsymbol{I} \end{bmatrix}, \tag{2.19}$$

where the Schur complement is given as $\boldsymbol{S} = \boldsymbol{D} - \boldsymbol{C}\widehat{\boldsymbol{A}}^{-1}\widehat{\boldsymbol{B}}$ and comprises the coupling between the physics of the problem. The advantage of this decomposition is that small decoupled systems associated with $\boldsymbol{A}$ and $\boldsymbol{S}$ need to be solved, for which general preconditioning techniques, such as Algebraic Multigrid **(AMG)**, can be used. For large-scale computations, the use of the exact Schur complement is not possible and approximations are used [67]. The scalability and efficiency of these approximations are considered to be main challenge of block preconditioners [68]. These preconditioners can often be used within a Krylov solver; therefore, they have also been defined as projection methods that avoid the difficulties of traditional projection methods with respect to boundary conditions and control of the time step size [66].

Common methods in this category are: Pressure-Convection Diffusion **(PCD)**, Least-Squares Commuters **(LSC)**, and Semi-Implicit Method for Pressure Linked Equations **(SIMPLE)** variants (SIMPLEC and SIMPLER). According to Liu et al. [66], the PCD and LSC preconditioners are scalable and only slightly affected by a change in the Reynolds number. However, the biggest disadvantage of PCD is that it needs to assemble an additional matrix, which makes it computationally inefficient. LSC attempts to solve this issue; it works well for mixed formulations that respect the inf-sup condition, but does not appear to have the same performance for stabilized formulations using equal-order interpolations [66]. The SIMPLE approaches are commonly known as a segregated approaches for the resolution of the Navier–Stokes in the context of FVM. Its simplest version uses the diagonal of $\boldsymbol{A}$ to define an approximation of the Schur complement. However, it has been shown that there are problems as the diagonal of $\boldsymbol{A}$ does not contain enough information about the convective term, leading to non-robustness with respect to Reynolds number [30, 66].

The second type of preconditioners are monolithic preconditioners, which work on the complete system matrix and comprise Incomplete LU **(ILU)** factorizations and multilevel methods, such as multigrid methods and domain decomposition methods. ILU factorization is a black box preconditioner that approximates the system matrix $\boldsymbol{F}'$ by a sparse lower diagonal matrix $\boldsymbol{L}'$ and a sparse upper diagonal matrix $\boldsymbol{U}'$. Entries of $\boldsymbol{L}$ and $\boldsymbol{U}$ that are too small are discarded. The challenge arises when deciding what "too small" means, which can be different depending on the PDE. This type of preconditioning is particularly effective in combination with GMRES [58], and several versions of it can be found in the literature, such as the no-fill ILU or the ILU(k) algorithm [69].

Multigrid methods include AMG and Geometric Multigrid **(GMG)** variants. They are based on the correction of different components of the error by defining a grid hierarchy or levels [70]. They are more difficult to implement, as they rely on the definition of the levels and are composed of several components: smoothers, coarse-grid solvers, and intergrid operators, that play an important role on the overall efficiency of the algorithm. Another type of preconditioners are Domain Decomposition **(DD)** methods, which decompose the domain into overlapping subdomains, where the problem is solved to obtain local solutions that are then used to construct a global approximation [60].

AMG and ILU preconditioners are particularly popular since open-source implementations are available via specialized libraries, e.g., `Trilinos`, `PETSc`. In contrast GMG and DD methods are more difficult to implement as they are directly related to the data structures and strategies used to generate the mesh. The development of efficient AMG, GMG and DD methods to solve the Stokes or incompressible Navier–Stokes equations is an active research field (see for example [71–74] for a few recent publications).

In the literature, not many publications compare different preconditioning techniques for the resolution of the Navier–Stokes equations system obtained after discretization with stabilization techniques. One exception, is the publication by Cyr et al. [68] where five preconditioning techniques are compared. They consider an additive Schwarz domain decomposition method, a fully coupled algebraic multigrid, PCD, LSC and SIMPLEC. They highlight a better scalability for the PCD, LSC and AMG compared to the other preconditioners; however, they highlight that block preconditioners avoid the implementation of complicated smoothers needed for the multigrid method. The LSC preconditioner was not found to be robust with respect to stabilization; most notably it did not converge for standard SUPG-PSPG stabilizations.

2.3.4 Discretization in Time

The strategies to discretize in time are directly related to the resolution strategies previously presented. In the case of segregated approaches, which are time-marching techniques, the time-stepping schemes are commonly called fractional-step methods. Once the convection and incompressibility operators are decoupled, several combinations of time-stepping schemes can be used for each resolution stage, for example, a combination of forward and backward Euler schemes [29], Crank-Nicolson schemes, or second-order backward differentiation schemes [30]. A review of several of these combinations along with stability analyses can be found in the books by John [30] and Karniadakis and Sherwin [61].

In coupled approaches, one can distinguish between θ-schemes (explicit Euler, implicit Euler, Crank-Nicolson), Second-order Implicit Backward Differentiation **(BDF2)** or Gear scheme, the generalized-α scheme [75], and high-order methods which include Implicit Runge-Kutta **(IRK)** methods and Rosenbrock methods [30, 76]. Implicit methods are generally better for the Navier–Stokes equations due to the stiffness nature of the problem [76]. Explicit schemes need to fulfill a stability condition (usually imposed according to the definition of the local Courant-Friedrichs-Lewy **(CFL)** number) that requires small time steps. In contrast, implicit schemes do not have to fulfill these stability conditions and are particularly suitable for adaptive time-step strategies. According to Elman [29], an efficient resolution of the Navier–Stokes equation necessarily requires adaptive time stepping to accurately capture all the underlying physics of the Navier–Stokes equations at different timescales.

The usage of an implicit Euler method in combination with high-order discretizations is not recommended, as it might lead to inaccurate results compared to higher-order schemes according to John [30]. Second-order schemes, such as the Crank-Nicolson scheme and the BDF2 scheme are among the most popular schemes because they are easy to implement, have a low memory requirement and are cheap in terms of the time needed to perform a single step [76]. The Crank-Nicolson scheme is A-stable and therefore, it might encounter numerical oscillations with rough initial data or boundary conditions [30]. These oscillations can only be damped out with very small time steps. The BDF2 scheme is A-stable and is highly popular due to its high order and improved stability properties compared to the Crank-Nicolson scheme.

High-order methods, such as the implicit Runge-Kutta method and Rosenbrock methods, have also been studied in the literature; they are unconditionally stable and have high-order accuracy, which makes them particularly suitable for high-order computations [77]. According to John [30], all higher-order schemes outperform θ-schemes in terms of accuracy and time-to-solution if adaptive time-stepping control is used. However, they require to solve larger systems when the order of convergence increases, leading to a higher computational requirement [76].

All of the methods mentioned, with the exception of the BDF2 scheme, are self-starting, i.e., one does not need to assume or compute a value to start the time-stepping iteration. However, the BDF2 scheme is very popular because it is L-stable and second-order accurate for both the velocity and the pressure. According to Hay et al. [76], BDF schemes are much more computationally efficient than IRK schemes for three-dimensional applications if used within their stability requirements. Methods to start the time-integration scheme include the interpolation of the analytical solution or the usage of lower-order schemes until the desired order is reached. According to Fehn [64], this does not imply a loss of accuracy since problems usually start with zero initial fields. An additional option that has been gaining popularity in the last couple of years, are space-time FEM discretizations which use the finite element method and DG also for the time-discretizations (see for example [78]).

2.4 Implementation Trends

Since the overall aim of this work is to achieve an efficient and scalable solver for the incompressible Navier–Stokes equations, it is necessary to look at the state-of-the-art in terms of open-source specialized FEM CFD software and their implementation trends. For this, an overview of the software is presented in **Table 2.2**. All considered software supports a matrix-free implementation instead of or in addition to a traditional matrix-based implementation. Most solvers use a segregated approach as the main resolution strategy, and only a few currently support Graphics Processing Unit **(GPU)** computations. In what follows, the main components of a state-of-the-art FEM solver are reviewed to understand the motivation behind matrix-free approaches.

Table 2.2 Overview of open-source specialized FEM CFD software. **Discretization:** SEM: Spectral Element Method, CG: Continuous Galerkin, DG: Discontinuous Galerkin; **Resolution strategy:** M: monolithic approach, S: segregated approach; **MB:** matrix-based implementation; **MF:** matrix-free implementation.

Software	**Language**	**CPU**	**GPU**	**FGPAs**	**Discretization**	**Resolution**	**MB**	**MF**
`Neko` [37]	Fortran 2008	✓	✓	✓	SEM	S	×	✓
`NEK5000` [65]	Fortran 77	✓	×	×	SEM	S	×	✓
`Nektar++` [36]	C++	✓	×	×	CG/DG	S	✓	✓
`NekRS` [38]	C++	×	✓	×	SEM	S	×	✓
`ExaDG` [40]	C++	✓	×	×	DG	M/S	×	✓

2.4.1 Main Components of a Finite Element Code

If a CG formulation is used, along with a monolithic resolution approach to solve the discretized Navier–Stokes equations, a finite element code must contain eight basic components; see **Figure 2.5**. The first component is the *mesh* or *triangulation*, which is a collection of cells and their lower-dimensional boundary objects (e.g., in 2D a cell can be a square made of lines and vertices; in 3D a cell can be a cube made of faces, lines, and vertices). In the literature, both hexahedra and tetrahedra are common; moreover, it is common to support structured, block-structured and unstructured meshes. Efficient data structures are needed, as it is important to be able to loop over all the components of a triangulation to compute matrix or vector entries in an efficient way in a parallel setting [79]. Moreover, these data structures are also essential to support adaptive mesh refinement. In the case of curved cells, information about the shape should also be available in the form of a manifold.

The second component comprises the *polynomial approximation* defined by the finite element spaces of the FEM discretization. It defines the number of Degrees of Freedom **(DoFs)** and their position in the unit cell. The third component is a *numerical integration* rule, also known as quadrature, to be able to compute each of the entries of the element matrices and right-hand side vector, which are defined as integrals. The fourth component is the *isoparametric mapping* which maps information between the real and the reference cell. This mapping is required for any efficient general purpose FEM solver, as it allows one to define the polynomial approximation and quadrature rules in only one reference cell [58]. The fifth

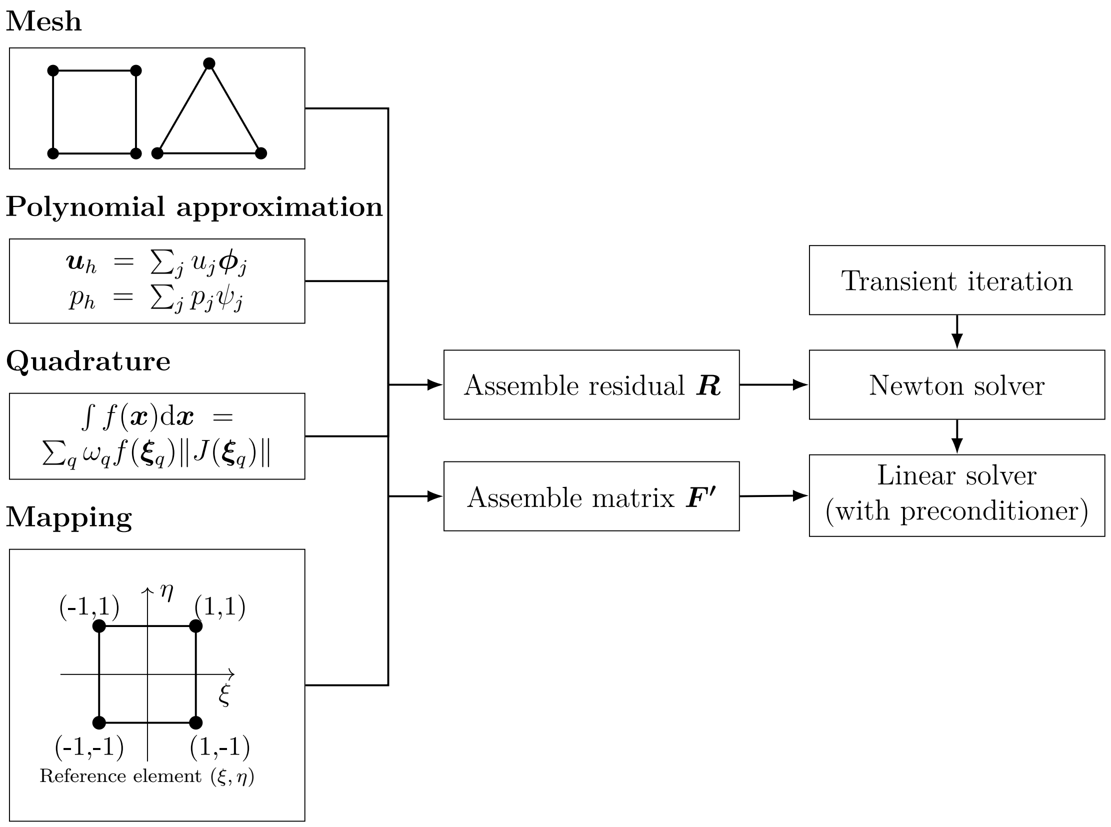
Mesh
Polynomial approximation
$\boldsymbol{u}_h = \sum_j u_j \boldsymbol{\phi}_j$
$p_h = \sum_j p_j \psi_j$
Quadrature
$\int f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \sum_q \omega_q f(\boldsymbol{\xi}_q) \| J(\boldsymbol{\xi}_q) \|$
Mapping
η
(-1,1)
(1,1)
ξ
(-1,-1)
(1,-1)
Reference element (ξ, η)
Assemble residual $\boldsymbol{R}$
Assemble matrix $\boldsymbol{F}'$
Transient iteration
Newton solver
Linear solver (with preconditioner)

Figure 2.5 Main eight components of a monolithic Navier–Stokes FEM solver.

component comprises the *assembly* of the system matrix and the right-hand side, and it can only be performed efficiently if all the previous components are present. The remaining three components are the main *time iteration loop*, the Newton *nonlinear solver*, and the *linear solver including preconditioner.*

Although FEM software requires all of these components, the overall efficiency of the solver is largely based on the assembly of the matrix and the efficiency of the linear solver. The reason for this is twofold. On one hand, practical problems require large meshes, which lead to a large number of DoFs, and therefore to a large (sparse) matrix. The number of DoFs per cell increases as the order of the polynomial increases, leading to more dense matrices in the case of high-order methods [80]. In general, these matrices have a large memory requirement. On the other hand, the resolution of such problems requires iterative methods as linear solvers, such as GMRES and BiCGStab. These iterative methods are based on matrix-vector multiplications. In order to perform this operation, one has to load the complete system matrix from the main memory. Since the matrices are large and generally, do not fit in cache, this quickly leads to a memory-bandwidth bottleneck. Recent trends aim at overcoming this bottleneck by avoiding the storage of the matrices and taking advantage of the large amount of floating point operations per second that can be performed on modern computing architectures.

2.4.2 Matrix-Free Approaches

The main idea of matrix-free approaches is to completely replace the computation of traditional matrix-vector products by computing each of the resulting vector values on the fly. To understand this idea, one must understand how to look at a matrix-vector multiplication as an operator evaluation. In what follows, a brief explanation based in the work by Kronbichler et al. [81] is presented. The multiplication of a system matrix $\boldsymbol{A}$ with a vector $\boldsymbol{x}$ can be written as a sum over all the elements k of a mesh:

$$\boldsymbol{b} = \mathcal{A}(\boldsymbol{x}) = \sum_k \boldsymbol{R}_k^\top \boldsymbol{A}_k \boldsymbol{R}_k \boldsymbol{x}, \tag{2.20}$$

where $\boldsymbol{R}_k$ and $\boldsymbol{R}_K^\top$ allow one to gather local DoFs from the vector $\boldsymbol{x}$, and to add partial results from the local operation to the vector $\boldsymbol{b}$, respectively. This formulation using an operator allows us to realize that the computations take place at the element level. A usual matrix-based assembly of a global matrix $\boldsymbol{A}$ is performed by looping over all cells, computing element entries, and adding values into the global DoFs, to later store this large (sparse matrix). Instead, for a matrix-free approach, one can simply compute the element contribution ($\boldsymbol{A}_k \boldsymbol{x}_k$) and store only the resulting entries of the vector $\boldsymbol{b}$. For an efficient evaluation of the element matrix-vector multiplication, one can further decompose it as three operator evaluations as follows:

$$\boldsymbol{A}_k \boldsymbol{x}_k = \boldsymbol{S}_k^\top \boldsymbol{Q}_k \boldsymbol{S}_k \boldsymbol{x}_k. \tag{2.21}$$

This is called the cell-quadrature approach and aims at separating this operation in order to take advantage of computations at the quadrature level on the reference element that can be reused for several cells [81]. To better understand this, consider the operator $(\nabla \boldsymbol{v}, \nabla \boldsymbol{u})$ multiplied by a vector $\boldsymbol{x}$. Taking the same basis functions $\boldsymbol{\phi}$ for the test function $\boldsymbol{u}$ and the solution $\boldsymbol{v}$, and using the mapping and the quadrature definitions, we can obtain the three matrix-vector multiplications that can replace the usual $\boldsymbol{A}_k\boldsymbol{x}_k$ computation:

$$\begin{aligned} b_i = \int_{\Omega_k} \nabla\boldsymbol{\phi}_i^\top \nabla\hat{\boldsymbol{\phi}}_j \hat{\boldsymbol{X}}_j^k \mathrm{d}\boldsymbol{x} &= \int_{\Omega_k} \left(\boldsymbol{J}_k^{-\top}\nabla_{\boldsymbol{\xi}}\hat{\boldsymbol{\phi}}_i\right)^\top \left(\boldsymbol{J}_k^{-\top}\nabla_{\boldsymbol{\xi}}\hat{\boldsymbol{\phi}}_j\right) \hat{\boldsymbol{X}}_j^k \left(\det \boldsymbol{J}_k\right) \mathrm{d}\boldsymbol{\xi} \\ &= \sum_q \underbrace{(\hat{\nabla}_{\boldsymbol{\xi}_q}\hat{\boldsymbol{\phi}}_i)^\top}_{\boldsymbol{\mathcal{S}}_k^\top} \underbrace{(\nabla_{\boldsymbol{\xi}_q}\hat{\boldsymbol{\phi}}_i)^\top \boldsymbol{J}_k^{-1}\omega_q(\det \boldsymbol{J}_k)\boldsymbol{J}_k^{-\top}}_{\boldsymbol{\mathcal{Q}}_k} \underbrace{\sum_j (\hat{\nabla}_{\boldsymbol{\xi}_q}\hat{\boldsymbol{\phi}}_j)}_{\boldsymbol{\mathcal{S}}_k} \hat{\boldsymbol{U}}_j^k, \end{aligned}$$

where ∇ is the gradient in the physical space, all quantities with a $\hat{\square}$ are vectors at the element level, $\boldsymbol{\xi}$ represents the coordinates of the reference element, $\boldsymbol{J}_k$ is the Jacobian and ω_q the quadrature weights. One can read this equation from right to left to understand the role of each of the matrix-vector multiplications: i) transform the vector values on the local DoFs to a vector of gradients in quadrature points ($\boldsymbol{\mathcal{S}}_k$), ii) multiply the gradients by the integration weights and Jacobian information ($\boldsymbol{\mathcal{Q}}_k$), and iii) apply the transposed of the gradient to have a vector of Laplacian values on the cell DoFs ($\boldsymbol{\mathcal{S}}_k^T$). In the end, a complete evaluation of an operator that computes a matrix-free multiplication for one cell can be visualized as shown in **Figure 2.6**.

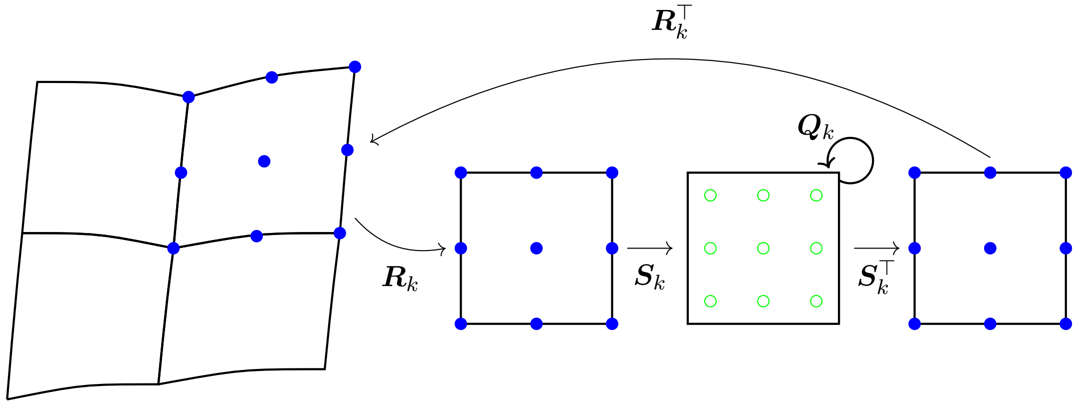

Figure 2.6 Matrix-free operator to compute a matrix-vector multiplication in one second-order cell in two dimensions.

This approach can be further optimized using sum factorization techniques for tensor-product elements using tensor-product quadrature. The general idea of sum factorization is to reuse information across directions within a cell, since test and solution functions can be decomposed as a tensor-product of one-dimensional functions [81]. In addition, these algorithms are suitable for vectorization, as they localize the computation for each cell making it feasible to perform matrix-free operator evaluations for several cells at the same time, using Single Instruction Multiple Data **(SIMD)** instructions.

General FEM libraries, such as `deal.II` [82] and `MFEM` [83], include implementations that allow users to implement PDE-specific operators that are then used for matrix-free calculations. These libraries are now used for matrix-free solvers in a variety of domains that are not exclusive to CFD; recent publications include models for battery particles [84], simulations of metal additive manufacturing processes [85], simulations of many-particle solid-state sintering processes [86] and hyperelasticiy [87]. Despite all this recent research, matrix-free approaches are not new. The first publication suggesting that one could replace a traditional sparse matrix by a global operator is that by Orszag et al. [88] in the 1980s in the context of SEM. In the context of CFD, these ideas were first used by Paul Fischer et al. in the 1990s within the `NEK5000` project [89–93], but are now widespread in the domain (see Table 2.2 for an overview of open source CFD software that uses this approach).

The reason why matrix-free approaches have gained popularity in the last decade is due to their suitability to use recent hardware. This hardware includes vectorization instructions (e.g., of the type AVX-512 which allows processing 8 double-precision floating-point numbers at the same time) and GPUs (e.g., see `libCEED` code [80]), which are optimal for minimizing memory movement and focus on arithmetic intensity, and are therefore highly used for performance computing parallelization [94]. An important challenge that arises with the usage of these approaches is the development of suitable (matrix-free) preconditioners. The reason for this is that most of the black-box preconditioners used for the resulting system of FEM discretizations need by construction the system matrix, e.g., AMG or ILU preconditioners. Therefore, a vast amount of research can now be found for preconditioners that need only the mesh or minimal information from the system matrix. Among these preconditioners, two groups can be distinguished: Domain Decomposition (DD) methods (see [95] for a review of these methods for fluid simulations) and Geometric Multigrid (GMG) methods. In the following, a review on the state-of-the-art of GMG methods is presented.

Geometric multigrid preconditioners

In the last 50 years, an extensive research on multigrid methods has been conducted. These methods were originally developed for elliptic PDEs, where they became particularly popular because of their optimal convergence rate. Since then, a vast amount of work has been done to be able to use them for complex problems. Geometric multigrid is only one variant among different multigrid methods (e.g., AMG, p-Multigrid **(PMG)**). For a general, but more detailed description of multigrid algorithms, the reader is referred to the key manuscript by Briggs [70], the review of multigrid methods by Benzi et al. [60] and the book by Saad [69]. The main difference between these methods is the strategy used to define the multigrid levels.

In geometric multigrid (often called h-MG) the levels are defined based on the hierarchy of meshes obtained through the process of refinement of a mesh for a particular problem. Once a hierarchy of meshes is obtained, the method is used to correct different components of the solution error by solving the problems at each of the MG levels in an organized way (V-cycle). Therefore, the main components of the algorithm are intergrid operators to transfer data between levels, relaxation methods (or smoothers) to solve problems at each level, and a coarse-grid solver for the coarsest-level problem.

Monolithic geometric multigrid methods for flow problems have been studied since the late 1990s [96]. One of the main research focus has been the selection and design of smoothers, since unlike elliptic problems, classical smoothers cannot be applied directly. Common methods include Vanka smoothers, Braess-Sarazin smoothers, and Uzawa smoothers [97,98]. The main principle of these preconditioners is to apply relaxation to individual block matrices. These block matrices are commonly defined as “patches” which are sets of DoFs that can be chosen according to certain criteria, e.g. all DoFs associated to a cell or those plus the ones belonging to neighboring cells; therefore, they are considered to be DD methods. Vanka smoothers were first presented by Vanka in 1986 [99] to overcome difficulties related to the zero pressure-pressure block, and different definitions of patches have been developed for different elements throughout the years. In the literature multiplicative (Gauss-Seidel type) and additive (Jacobi type) Vanka smoothers can also be found, where the latter are highly popular due to their suitability for parallelism [100].

In the following, some recent literature on the GMG will be reviewed. In 2022, Kohl et al. [97], constructed a monolithic GMG solver with an inexact Uzawa smoother for the Stokes problem considering tetrahedral elements, high-order FE discretizations (with PSPG stabilization), and a matrix-free implementation. They demonstrated good parallel scalability using up to 147K processes and systems with more than 3.6×10^{12} unknowns using both stabilized equal order elements and Taylor-Hood pairs. In the same year, Voronin et al. [101] developed a low-order GMG preconditioner considering Vanka and Braess-Sarazin as smoothers to solve the Stokes equations; the same approach was later extended by the same group to AMG in [71]. In 2024, Voronin et al. introduced a *ph*-MG multigrid for high-order discretizations of stationary Stokes systems in [102]. They considered Vanka smoothers with star patches and the results showed better performance and reduced setup and solve times for higher orders and unstructured problems in comparison to only GMG.

Another work that considers Taylor-Hood elements for the Stokes and generalized Stokes equations is that by Jodlbauer et al. [103]. They implemented a matrix-free monolithic GMG solver using Chebyshev-Jacobi smoothers to solve the system with a zero pressure-pressure block and obtained satisfactory results. They indicated that the method could serve as a basis for developing a method with similar ideas for Navier-Stokes problems, but more work would need to be done to ensure robustness for convection-dominated flows. Adler et al. [104] considered a formulation without stabilization and compared their multigrid implementation with a segregated method; they observed that multigrid was more efficient and robust and capable of solving steady-state and transient cases up to 170 million DoFs. However, they stated that the main limitation of the work was the low order (P_2 for the velocity, P_1 for the pressure and lowest-order Nédélec elements for the magnetic field) and the lack of stabilization of the discretization they used.

In 2023, Abu-Ladeh et al. [100] presented a Newton-Krylov-multigrid method to solve time-dependent Stokes, Navier–Stokes, and resistive magnetohydrodynamics equations, using implicit Runge–Kutta discretizations in time and standard mixed FE discretizations in space. Their multigrid algorithm considered an overlapping additive Schwarz relaxation of Vanka type. They evaluated scalability up to 640 cores and proposed as future research the extension of the method to high-order discretizations. The most recent publication on the topic is that by Rafiei et al. [98], where an improved Vanka relaxation scheme is introduced. A novel patch with overlap in both pressure and velocity DoFs is presented and it shows significant improvement in both triangular and quadrilateral meshes.

CHAPTER 3 THESIS OBJECTIVES AND ORGANIZATION

In this chapter, the gaps in the literature are summarized, and the objectives of the research carried out in this thesis are stated. In addition, the organization of the following chapters, which are directly linked to the research objectives, is described.

3.1 Gaps in the Literature

The main findings of the previous literature review can be summarized as follows:

- In process intensification, several publications point to the importance of including CFD in the design pipeline of intensified devices for a diverse range of chemical processes, as experimental studies are not enough to fully characterize the flow due to complex geometries. Common challenges encountered when trying to simulate process-intensified equipment are the lack of multiphysics simulation software and the lack of computational resources for accurate simulations. Moreover, in current studies, a lack of turbulence modeling or the usage of RANS approaches, which are not ideal for transient flows, can be observed.
- There is a wide range of strategies for modeling turbulent flows. In recent years, ILES approaches have gained popularity because of their "parameter-less" construction and have been proven to be promising for turbulent flows. However, more comparisons are needed between implicit and explicit LES approaches to identify the most promising techniques.
- The main advantage of high-order FEM discretizations is that they are particularly useful for unstructured meshes. Therefore, high-order FEM is attractive when it comes to the simulation of complex geometries with curved boundaries. Despite this, most of the literature on the simulation of flow within process-intensified devices uses second-order accurate FVM schemes.
- State-of-the-art open-source FEM CFD software focuses on high-order discretizations and matrix-free implementations. However, most of the software uses a segregated approach as a resolution technique. No software has explored the coupled resolution strategy of the incompressible Navier–Stokes equations using a continuous Galerkin formulation with stabilization techniques, a matrix-free implementation, and a monolithic preconditioner.

The main hypothesis of this project is that a stabilized Navier–Stokes solver may accurately predict the behavior of turbulent flows as part of computational modeling useful for achieving process intensification in process engineering. Moreover, if this solver is implemented in a matrix-free approach using a GMG preconditioner, we may achieve an efficient and scalable implementation in modern computer architectures that allow us to run practical problems in the context of PI with reasonable resources.

3.2 Research Objectives

The main objective of this thesis is to develop and implement an efficient and accurate Navier–Stokes solver to model the turbulent flow encountered within process-intensified devices. The objective is divided into four sub-objectives:

SO1: Establish the accuracy of an implicit large-eddy simulation approach that considers stabilization techniques for a complex turbulent flow benchmark known as the periodic hills flow.

SO2: Achieve a scalable and efficient implementation of a stabilized Navier–Stokes solver using a matrix-free approach and a suitable geometric multigrid preconditioner.

SO3: Assess the efficiency of the solver by simulating a rotating turbulent flow encountered in process-intensified equipment.

SO4: Extend the matrix-free solver for turbulent-flow problems with locally refined meshes.

3.3 Structure of the Thesis

In **Chapter 4**, we provide information on the methodology used to achieve all the previously mentioned objectives. It introduces the open-source software `Lethe`, the basis of all this work, and highlights the contributions to the software throughout the work on this thesis, along with the computational resources.

Chapter 5 (Article 1) studies the behavior of the continuous Galerkin solver using a complex turbulent benchmark known as periodic hills. It evaluates the behavior of the implicit large-eddy simulation approach which considers both SUPG and PSPG stabilization techniques by comparing simulation data to both experimental and computational data available in the literature. Moreover, it studies the effect of the mesh refinements, time step, and averaging time on average velocities and Reynolds stresses.

In **Chapter 6** (Article 2), the matrix-free stabilized Navier–Stokes solver is developed; along with its GMG preconditioner needed for the linear solver. The accuracy and efficiency of the solver is first validated using a manufactured solution in 3D, followed by a steady-state flow around a sphere problem and the Taylor–Green vortex benchmark. The efficiency of the solver is evaluated through the following aspects: time to solution, strong and weak scalability, and number of transient, nonlinear, and linear iterations.

Chapter 7 (Article 3) aims to assess the efficiency of the solver by simulating a case closer to real applications: the turbulent Taylor–Couette flow. This problem considers rotating flow and curved wall; two factors that are commonly encountered in applications in the field of PI and that are considered a crucial criterion to deem a fluid dynamics solver practical for real-life applications. This article shows the effect of the finite element order for the velocity and the pressure when predicting quantities like enstrophy and kinetic energy, vorticity and Q criterion distributions. In addition, it provides a novel analysis on the numerical dissipation included by the numerical scheme providing valuable data for future studies.

Chapter 8 evaluates the ability of the solver to simulate problems that require local mesh refinement; in particular, it compares two different geometric multigrid implementations, global coarsening and local smoothing, and highlights their advantages and disadvantages depending on the resources available for a transient and a steady-state simulation.

CHAPTER 4 METHODOLOGY

This chapter provides an overview of `Lethe`, the software where the solver was implemented, and highlights the contributions as part of this thesis. It introduces important details on the implementation of the matrix-free infrastructure and the GMG preconditioner, presents the verification and validation strategy, and a brief overview of the computational resources utilized for this work.

4.1 The Software: `Lethe`

`Lethe` is an open-source CFD, Discrete Element Method **(DEM)** and coupled CFD-DEM software actively developed at the CHAOS Laboratory at Polytechnique Montreal [105, 106]. The software is publicly available on GitHub[1] and has a detailed user guide and a developer guide based on `Doxygen`. It is a high-order continuous Galerkin solver to simulate single and multiphase flows. It is built on the top of the well-known finite-element library `deal.II`[2] [82], through which it utilizes `p4est` for its mesh capabilities [107], `MPI` for parallelization and `Trilinos` for matrix-based linear algebra [108]. `Lethe` supports multiple physics: fluid dynamics, heat transfer, and a passive tracer. It also has capabilities for multiphase flow simulation via one-fluid methods that include the volume of fluid method and the Cahn-Hilliard method, and granulate flow via the DEM. These different physics are often coupled for multiphysics simulations needed for practical problems in process engineering.

The fluid solvers of `Lethe` are implemented from a common base class (`NavierStokesBase`). At the beginning of the project, two main fluid solvers were derived from this main class: the matrix-based fluid solver (`lethe-fluid`)– the main solver of Lethe– and the block fluid solver (`lethe-fluid-block`). The main difference between the matrix-based solver and the block solver relies on the preconditioners used to solve the coupled problem; the former supports monolithic preconditioners, while the latter uses a block preconditioner. The matrix-based solver is the most robust fluid solver in `Lethe`. Other fluid solvers have been implemented on the top of this main solver since the beginning of the software in 2018: a Volume-Averaged Navier–Stokes (VANS) solver (`lethe-fluid-vans`), a Navier–Stokes solver with immersed boundary conditions using the Sharp method (`lethe-fluid-sharp`), and a Navier–Stokes solver with immersed boundary conditions using the Nitsche methods (`lethe-fluid-nitsche`). All of these applications have been extensively tested for different flows encountered in chem-

[1] `https://github.com/chaos-polymtl/lethe`
[2] `https://github.com/dealii/dealii`

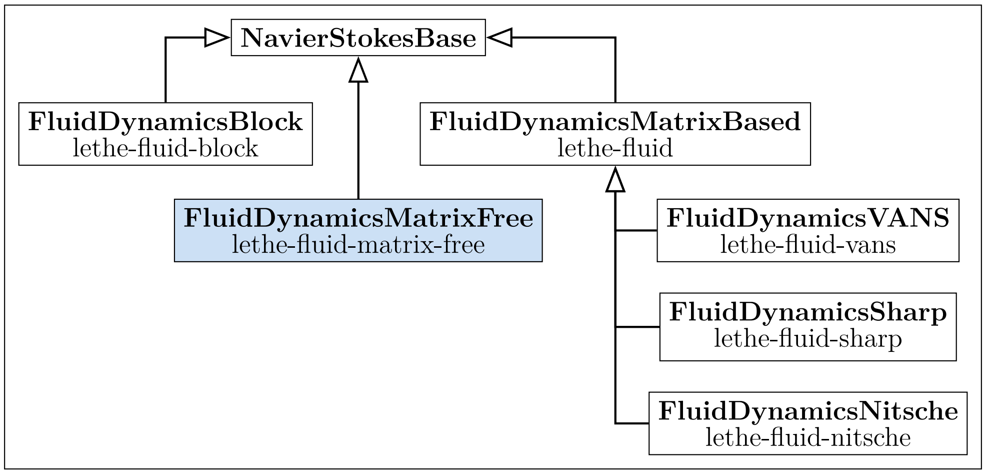

Figure 4.1 Diagram with the fluid solvers of `Lethe` including the class name and the application name at the bottom. The blue box corresponds to the matrix-free solver added in this work.

ical processes; see `Lethe`'s publications for some examples. [3]

Figure 4.1 shows a schematic with all the fluid solvers of `Lethe`. The new matrix-free solver implemented as part of this project can be seen in blue (described in detail in Section 4.2). The fluid solvers use the finite element method to solve the incompressible Navier–Stokes equations via a monolithic coupled solution strategy with SUPG/PSPG, GLS and/or Grad-Div stabilization. The nonlinear problem is solved using a Newton-Krylov solver with one of two preconditioners from the `Trilinos` library: an Algebraic Multigrid (AMG) and an Incomplete LU decomposition (ILU). The simulations can be run via a parameter file that contains information on the geometry, the boundary conditions, the physical properties, among other important parameters. More than 40 examples are available in the documentation of `Lethe` showcasing the usage of the different applications via parameter files.[4] The software has more than 400 functionality tests and application-specific tests that allow for continuous integration. The author of this thesis is a coauthor in a recent publication of `Lethe`, where its capabilities are explained in detail, see [106].

4.2 Matrix-free Fluid Solver

As part of this project, the **`lethe-fluid-matrix-free`** application was implemented in `Lethe` as an alternative to traditional matrix-based fluid solvers (see blue box in Figure 4.1).

[3]`https://chaos-polymtl.github.io/lethe/documentation/publications.html`

[4]`https://chaos-polymtl.github.io/lethe/documentation/examples/examples.html`

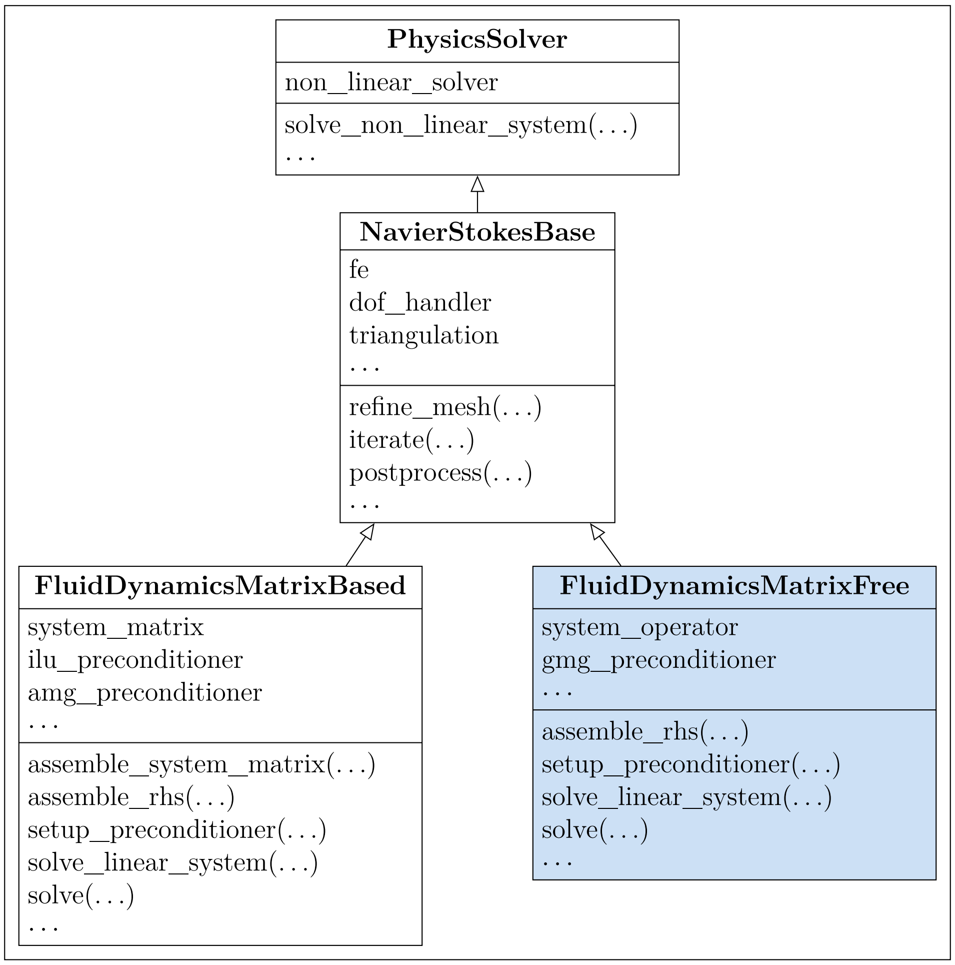

Figure 4.2 General UML diagram of the main classes needed for a fluid solver in `Lethe`.

For this, the matrix-free algorithms of the `deal.II` library were used to implement a matrix-free operator instead of a system matrix to solve the resulting nonlinear problem. The implementation in `deal.II` was first introduced in the library by Kronbichler and Kormann in 2012 [81], and has been further developed since then (see the last `deal.II` release papers for a description of the most recent features supported [82,109]). This matrix-free algorithms have been extensively tested for different PDEs from Poisson equations with CG discretizations to the incompressible and compressible Navier–Stokes equations with DG discretizations (see, for example, [85, 87, 110–112]).

The implementation of the matrix-free fluid solver was adapted according to the existent architecture of `Lethe`. **Figure 4.2** shows a general UML diagram with the main classes that are needed for a fluid solver in `Lethe`. The `PhysicsSolver` class is in charge of the implementation of the Newton non-linear solver, and therefore, it needs access to the matrix (or an operator in the matrix-free context), the right-hand side, the preconditioner and the linear solver. All of these components are implemented in the main fluid dynamics classes for the matrix-based and matrix-free solvers. Both of these classes derive from the `NavierStokesBase` class, which holds the common attributes needed for the resolution of a problem via FEM: a triangulation, the finite element discretization information via the attribute `fe`, and an object in charge of the numbering of the DoFs of the problem (`dof_handler`). The member functions of this class are in charge of creating the mesh and contain the main iteration loop of the solver along with post-processing features. The main differences between the `FluidDynamicsMatrixFree` solver and the `FluidDynamicsMatrixBased` solver can be summarized as follows:

i) instead of a (sparse) system matrix the matrix-free solver needs a system operator;

ii) the matrix-free solver does not require a specific function to assemble the main system matrix;

iii) the matrix-free solver supports a GMG preconditioner in addition to the AMG and ILU preconditioners used by the matrix-based solver.

In the following, implementation details of the operator for the stabilized formulation of the incompressible Navier–Stokes equations and the GMG preconditioner are given.

4.2.1 Matrix-Free Operator

The matrix-free operator was implemented in a different class; its general UML diagram is presented in **Figure 4.3**. It follows the pattern of general matrix-free operators implemented in `deal.II`.[5] It consists of a base class (`NavierStokesOperatorBase`) that holds the `matrix_free` object, the `fe_cell_integrator`, and the functions needed for the linear solver. The `matrix_free` object stores information regarding the relation between local and global DoFs, constraints, mappings, and the shape functions of the finite element. The `fe_cell_integrator` is the object in charge of evaluating the operator using the cell quadrature approach, i.e., it reads/writes from/to global vectors, integrates function values on the unit-cell using the sum-factorization approach, and applies the mapping transformation.

The main function of the base class implements the matrix-vector multiplication `vmult` which calls the `do_cell_integral_range` to perform vector multiplications over a batch of cells in a vectorized fashion. The additional `evaluate_residual` call is in charge of assembling the

[5] `https://www.dealii.org/developer/doxygen/deal.II/namespaceMatrixFreeOperators.html`

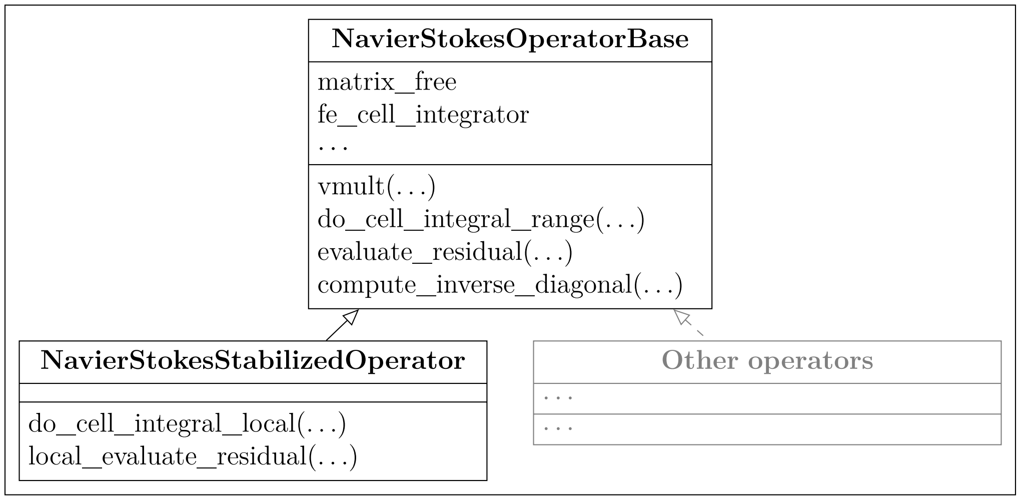
NavierStokesOperatorBase
matrix_free
fe_cell_integrator
...
vmult(...)
do_cell_integral_range(...)
evaluate_residual(...)
compute_inverse_diagonal(...)
NavierStokesStabilizedOperator
do_cell_integral_local(...)
local_evaluate_residual(...)
Other operators
...
...

Figure 4.3 General UML diagram of the matrix-free operator implemented in `Lethe`.

residual of the matrix using the same approach. The two functions (`do_cell_integral_range` and `local_evaluate_residual`) of the derived class are specific to the PDE being solved and, therefore, are implemented in a new class called `NavierStokesStabilizedOperator`. This class is the matrix-free stabilized operator and corresponds to the operator used throughout this work. Similar to all the other fluid solvers of `Lethe`, it uses SUPG/PSPG and GLS stabilizations and can be used for steady and transient problems. In mathematical terms, the operator implements the application of the Jacobian matrix and the assembly of the right-hand side residual vector for the following discretized system of equations using a CG formulation with stabilization:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \delta \boldsymbol{U} \\ \delta \boldsymbol{P} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{R}_u \\ \boldsymbol{R}_p \end{bmatrix}, \tag{4.1}$$

where each of the blocks of the Jacobian system are given as:

$$\begin{aligned} \boldsymbol{A}^k_{i,j} &= (\boldsymbol{\phi}_i, \partial_t \boldsymbol{\phi}_j + (\boldsymbol{u} \cdot \nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j \cdot \nabla)\boldsymbol{u})_{\Omega_k} + \nu(\nabla \boldsymbol{\phi}_j, \boldsymbol{\phi}_i)_{\Omega_k} \\ &\quad + (\tau_{\text{SUPG}}(\boldsymbol{u} \cdot \nabla)\boldsymbol{\phi}_i, \partial_t \boldsymbol{\phi}_j + (\boldsymbol{u} \cdot \nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{\phi}_j)_{\Omega_k} \\ &\quad + (\tau_{\text{SUPG}}(\boldsymbol{\phi_j} \cdot \nabla)\boldsymbol{\phi}_i, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{u} - \boldsymbol{f})_{\Omega_k} \\ &\quad + (\tau_{\text{SUPG}}(\boldsymbol{\phi_j} \cdot \nabla)\boldsymbol{\phi}_i, \nabla p)_{\Omega_k}, \end{aligned} \tag{4.2}$$

$$\boldsymbol{B}^k_{i,j} = (\nabla \cdot \boldsymbol{\phi}_i, \psi_j)_{\Omega_k} + (\tau_{\text{SUPG}}(\boldsymbol{u} \cdot \nabla)\boldsymbol{\phi}_i, \nabla \psi_j)_{\Omega_k}, \tag{4.3}$$

$$\begin{aligned}
\boldsymbol{C}^k_{i,j} &= (\psi_i, \nabla \cdot \boldsymbol{\phi}_j)_{\Omega_k} + (\tau_{\text{PSPG}} \nabla \psi_i, \partial_t \boldsymbol{\phi}_j + (\boldsymbol{u} \cdot \nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{\phi}_j)_{\Omega_k}, && (4.4)\\
\boldsymbol{D}^k_{i,j} &= (\tau_{\text{PSPG}} \nabla \psi_i, \nabla \psi_j), && (4.5)
\end{aligned}$$

and the residual:

$$\begin{aligned}
\boldsymbol{R}^k_{u(i,j)} =& - (\boldsymbol{\phi}_i, \partial_t \boldsymbol{u})_{\Omega_k} - (\boldsymbol{\phi_i}, (\boldsymbol{u} \cdot \nabla)\boldsymbol{u})_{\Omega_k} + (p, \nabla \cdot \boldsymbol{\phi_i})_{\Omega_k} - \nu(\nabla \boldsymbol{u}, \nabla \boldsymbol{\phi_i})_{\Omega_k} + (\boldsymbol{\phi_i}, \boldsymbol{f})_{\Omega_k} \\
&= - \sum_k (\tau_{\text{SUPG}} (\boldsymbol{u} \cdot \nabla)\boldsymbol{\phi}_i, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} - \boldsymbol{f})_{\Omega_k}, && (4.6)\\
\boldsymbol{R}^k_{p(i,j)} =& - (\psi_i, \nabla \cdot \boldsymbol{u})_{\Omega_k} \\
&- \sum_k (\tau_{\text{PSPG}} \nabla \psi_i, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u})_{\Omega_k}. && (4.7)
\end{aligned}$$

For a detailed derivation, the reader is referred to Appendices C and D. The architecture of the base class is made so that other matrix-free operators can be added in the future requiring only the implementation of the cell integral and the cell residual. This is particularly useful for further adaptation of the existing fluid solvers in `Lethe` to the matrix-free approach.

4.2.2 Geometric Multigrid Preconditioner

A GMG preconditioner was incorporated in `Lethe` relying on the implementation of `deal.II` [112,113]. The general architecture uses a base class for a generic matrix-free GMG preconditioner: `MFNavierStokesPreconditionGMGBase`. **Figure 4.4** shows a general UML diagram. The class contains all the main components of a GMG method, including the information of the minimum and maximum multigrid levels, the transfer operators, the level operators, the smoothers and the coarse-grid solver. All of this components are essential for a V-cycle of GMG (see **Algorithm 1**).

A derived class, the `MFNavierStokesPreconditionGMG` class, was created to be used with the stabilized Navier–Stokes operator previously introduced. It has only two specific functions (`create_level_operator` and `initialize`) that allow us to isolate all the components of the multigrid operator that do not depend on the operator, such as, the setup of level operators and the information that needs to be transferred to the different levels in order to be used by the operator (linearization point or time derivatives of previous solutions); this eases the implementation of additional GMG preconditioners that are tailored to the specific operators.

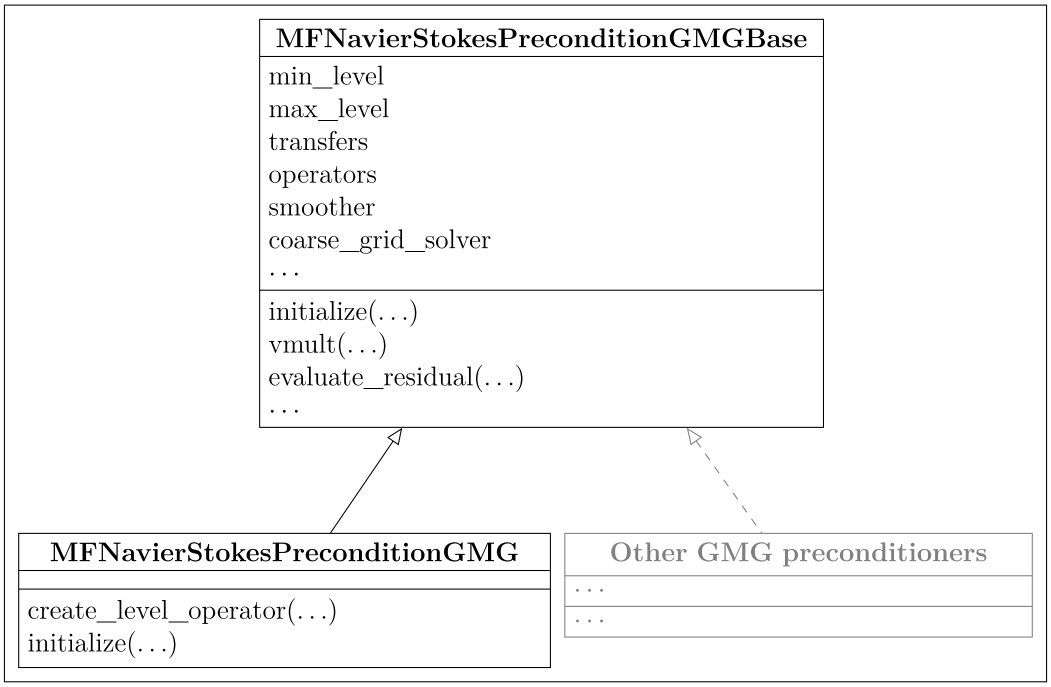

Figure 4.4 General UML diagram of the geometric multigrid preconditioner implemented in `Lethe`.

Algorithm 1 Pseudocode for one v-cycle of the GMG preconditioner with l levels used to solve $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$. It uses level operators $\mathcal{A}^{(l)}$, intergrid operators (restriction $\mathcal{R}_l^{(l-1)}$ and prolongation $\mathcal{P}_{(l-1)}^l$), smoothers and a coarse-grid solver.

```
function GMG($l$, $\mathcal{A}^{(l)}, \boldsymbol{x}^{(l)}$, $\boldsymbol{b}^{(l)}$)
    if $l = 0$ then
        $\boldsymbol{x}^{(0)} \leftarrow$ CoarseGridSolver($\mathcal{A}^{(0)}, \boldsymbol{x}^{(0)}, \boldsymbol{b}^{(0)}$)    // Solve coarse grid
    else
        $\boldsymbol{x}^{(l)} \leftarrow$ Smoother($\mathcal{A}^{(l)}, \boldsymbol{x}^{(l)}, \boldsymbol{b}^{(l)}$)    // Presmooth
        $\boldsymbol{r}^{(l)} \leftarrow \boldsymbol{b}^{(l)} - \mathcal{A}^{(l)}\ \boldsymbol{x}^{(l)}$    // Compute residual
        $\boldsymbol{b}^{(l-1)} \leftarrow \mathcal{R}_l^{(l-1)}\ \boldsymbol{r}^{(l)}$    // Restrict
        $\boldsymbol{x}^{(l-1)} \leftarrow$ GMG ($l-1$, $\mathcal{A}^{(l-1)}$, 0, $\boldsymbol{b}^{(l-1)}$)
        $\boldsymbol{x}^{(l)} \leftarrow \boldsymbol{x}^{(l)} + \mathcal{P}_{(l-1)}^l\ \boldsymbol{x}^{(l-1)}$    // Prolongate
        $\boldsymbol{x}^{(l)} \leftarrow$ Smoother($\mathcal{A}^{(l)}, \boldsymbol{x}^{(l)}, \boldsymbol{b}^{(l)}$)    // Postsmooth
    return $\boldsymbol{x}^{(l)}$
```

Transfer Operators

The prolongation operator is used to transfer the result $\boldsymbol{x}$ from a coarse level $l-1$ to a fine level l, as follows:

$$\boldsymbol{x}^{(l)} = \mathcal{P}_{(l-1)}^l \boldsymbol{x}^{(l-1)} \tag{4.8}$$

and the restriction operator is defined as the transpose of this operator $\mathcal{R}_l^{(l-1)} = (\mathcal{P}_{(l-1)}^l)^\top$. These operators are also used to transfer the residuals $\boldsymbol{r}^{(l)}$ between levels, and to interpolate solutions of previous transient iterations in the case of transient simulations and/or the values of the linearization point within the nonlinear Newton solver.

Level Operators

The level operators $\mathcal{A}^{(l)}$ are never assembled and stored in memory, with the exception of the coarse grid operator $\mathcal{A}^{(0)}$. The matrix-vector multiplications that need to be performed within the multigrid algorithm take advantage of the matrix-free operator previously explained (`NavierStokesStabilizedOperator`) for the computations at each level.

Smoothers

For the pre- and post-smoothing steps of the GMG algorithm, a relaxation scheme of the following type for each level was considered:

$$\boldsymbol{x}_{n+1}^{(l)} = \boldsymbol{x}_n^{(l)} + \omega \boldsymbol{P}^{-1} \boldsymbol{r}_n^{(l)}, \tag{4.9}$$

where $\boldsymbol{r}_n^{(l)}$ is the residual, defined as $\boldsymbol{r}_n^{(l)} = \boldsymbol{b}^{(l)} - \mathcal{A}^{(l)} \boldsymbol{x}_n^{(l)}$ with the level operator $\mathcal{A}^{(l)}$; and $\boldsymbol{P}$ is a preconditioner. For the preconditioner, two options were considered: an Inverse Diagonal **(ID)** and an Additive Schwarz Method **(ASM)**. The former considers a diagonal matrix corresponding to the diagonal of each matrix-free level operator. The latter consists of approximate solves on sub-blocks of the level operator with all the DoFs of a cell (also known as cell-centric patches). The ASM is built in two steps; first, the sparse system matrices are assembled, and second, the blocks of these matrices are extracted, and an LU decomposition is performed on these. In **Figure 4.5**, a general UML diagram of the `PreconditionBase` class, from which the two classes of preconditioners are derived, is presented.

The choice of the relaxation parameter plays an important role on the overall efficiency of the algorithm. For this work, the relaxation parameter ω is computed via:

$$\omega = 2/(\lambda_{\min}(\boldsymbol{P}^{-1}\mathcal{A}) + \lambda_{\max}(\boldsymbol{P}^{-1}\mathcal{A})), \tag{4.10}$$

which is optimal for symmetric positive definite matrices [114]. For the estimation of the maximum eigenvalue $\lambda_{\max}$ we use 20 iterations of the power iteration method. Then, the minimum eigenvalue is set to $\lambda_{\min} = \lambda_{\max}/\psi$, where values of ψ between 2 and 20 are common in literature [115]. For more details on the eigenvalue estimation algorithm the

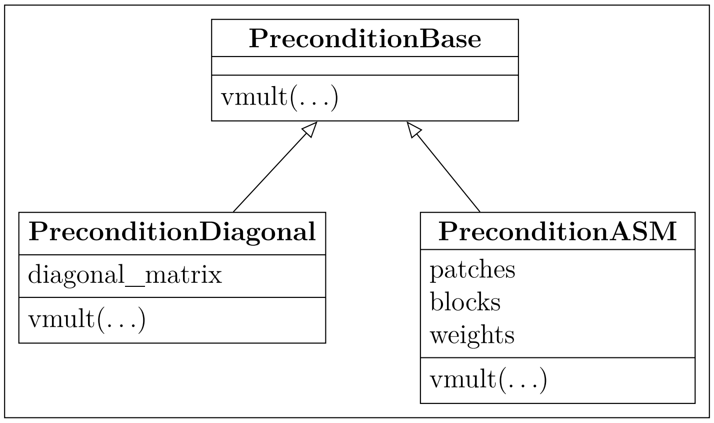

Figure 4.5 General UML diagram of the preconditioner classes implemented for the smoothers within the matrix-free fluid solver.

reader is referred to [116]. Other options that can be found in the literature include: local Fourier Analysis (LFA), Chebyshev iteration method and Krylov iteration method [117].

Coarse Grid Solvers

Five options were implemented for the coarse grid solver:

i) a direct solver as implemented in the Amesos2 package in Trilinos [118];

ii) an AMG method as implemented in the ML package in Trilinos [119];

iii) an ILU method as implemented in the Ifpack package in Trilinos [120];

iv) a GMRES method preconditioned by a PMG with a direct solver as a coarse-grid solver (supported by `deal.II`), an AMG [119], and an ILU [120] method;

v) a single v-cycle of a PMG solver with a direct solver as a coarse-grid solver (supported by `deal.II`).

Locally Refined meshes

There are different variants of this GMG preconditioner when locally refined meshes are considered: Global Coarsening **(GC)** and Local Smoothing **(LS)**. They only differ in the case of locally refined meshes, since their main difference relies on the way the hierarchy of multigrid levels is defined in this case. The global coarsening approach considers the entire domain for each level, while the local smoothing approach uses only the most refined cells at each stage of the mesh refinement. To better understand these strategies, see **Figure 4.6** for an example of an L-shaped domain.

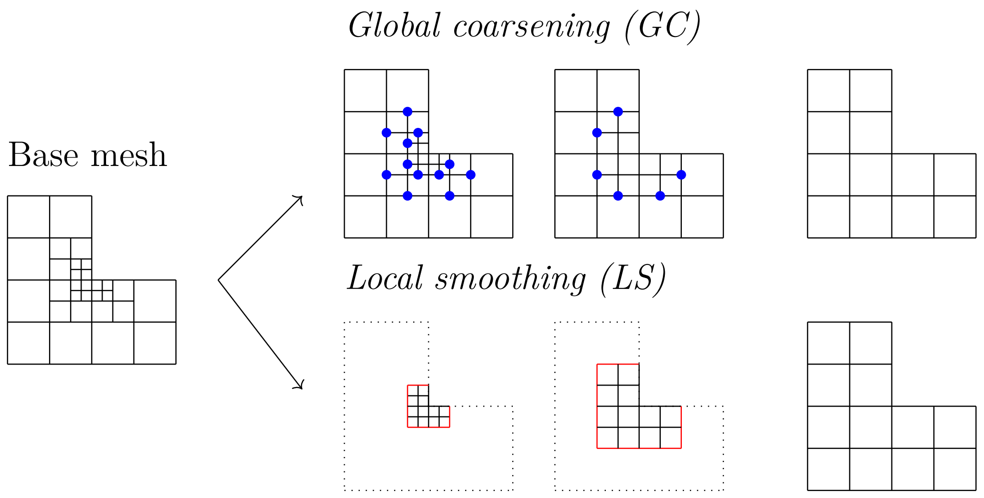

Figure 4.6 Comparison of the definition of the hierarchy of multigrid levels for global coarsening and local smoothing when considering an L-shaped domain and three levels. Blue circles represent hanging nodes and red lines represent interface edges.

The global coarsening approach has hanging nodes (nodes located on the edge of an element that are not shared by the neighbor element); in contrast, the local smoothing approach does not present hanging nodes but has internal interfaces (or edges). For GC the hanging nodes need to be considered by the smoother. For LS, data need to be transferred to and from all multigrid levels, and a different treatment is required for interior DoFs, and those located on the internal interfaces (or edges) between the levels. These two versions of GMG are implemented and are supported by `deal.II`. Furthermore, they were compared and tested in terms of parallel scalability, implementation complexity, and computational efficiency for Poisson problems in a publication by Munch et al. [112]. The author of this thesis is a coauthor of this publication, where details of the implementation differences and a practical application in the scope of Stokes flow with block preconditioners can also be found.

4.3 Verification and Validation

It is essential to verify and validate the implemented solver. Oberkampf et al. [4] provide appropriate definitions for these two methodologies:

- Verification: Determines whether the numerical algorithms are correctly implemented without any errors in the computer code. In addition, it evaluates the accuracy of the input and output data and the numerical accuracy of the solution.

- Validation: assesses the accuracy of the model by comparing with experimental data, the usage of the model to make a prediction, and the adequacy of the model for the intended use.

In the context of PDEs, verification demonstrates that the spatial discretization method produces the expected convergence rate as the mesh is refined. In addition, it determines whether the code is implemented in a corrected manner and produces repeatable results on specific computer hardware. Throughout this work, the verification steps taken can be summarized as follows:

- The order of convergence of the chosen discretization for the incompressible Navier–Stokes equations was verified via the Method of Manufactured Solutions **(MMS)** in 2D and 3D, and via a Taylor vortex problem in 2D. This verification was also performed using structured and unstructured grids. For a CG formulation with stabilization and elements of equal order p, one expects to have an order of $O(\Delta h^{p+1})$ for the velocity and $O(\Delta h^{p})$ for the pressure.

- Continuous integration within the `Lethe` code was used to test the correctness and accuracy of the code every time a new feature was added by continuous compilation (in release and debug mode and considering compilation warnings). This included the creation of several tests for the `lethe-fluid-matrix-free` application as the solver was being developed.

In terms of validation, the following steps were taken throughout this work:

- The ILES capabilities of the solver were validated against experimental and computational reference data found in the literature for different benchmarks: flow around a sphere, Taylor–Green vortex, periodic hills, and Taylor–Couette flow.

- The parallel scalability and time to solution of the solver was evaluated using different steady-state and transient problems in order to establish the adequacy of the model in terms of computational requirements.

- All steady-state and transient problems evaluated throughout this work were used as reproducible evidence that the solver achieves the desired level of accuracy in the solution of incompressible flow problems. This allowed us to infer that the behavior of a wide range of flow problems can be predicted accurately using the solver.

4.4 Computational Resources

The software development was carried out on a local desktop (AMD Ryzen 9 5900X 12-core processor x 24). For larger simulations, computational resources from the Digital Research Alliance of Canada were used to take advantage of distributed memory parallelism. In addition, SuperMUC-NG in Germany was also used for certain simulations to assess the GMG capabilities with local mesh refinement. A summary of the specifications of each cluster is presented as follows:

- Niagara: 2024 nodes each with 40 Intel Skylake cores (2.4 GHz) with 202 GB of RAM per node, EDR Infiniband network in a 'Dragonfly+' topology, and with support for AVX512 instructions for vectorization over 8 cells/doubles (512 bits).
- Beluga: 974 nodes each with 40 Intel Gold 6148 Skylake (2.4 GHz) with 92GB to 752GB of RAM per node, and EDR Mellanox Infiniband network.
- SuperMUC-NG Phase 1: 6336 nodes each with 48 Intel Skylake cores (3.1 GHz) with 96 GB of RAM per node, and with support for AVX512 instructions for vectorization over 8 cells/doubles (512 bits).

CHAPTER 5 ARTICLE 1 : AN IMPLICIT LARGE-EDDY SIMULATION PERSPECTIVE ON THE FLOW OVER PERIODIC HILLS

Laura Prieto Saavedra, Catherine E. Niamh Radburn, Audrey Collard-Daigneault, Bruno Blais.

Published in Computers and Fluids, volume 283, 106390, on the 3rd August 2024. Available at `https://doi.org/10.1016/j.compfluid.2024.106390`. *Part of the content was presented at the VI ECCOMAS Young Investigators Conference in Valencia in July 2021 and the Chemical Engineering Research Days (McGill, Polytechnique and Concordia) in March 2022.*

Declaration: The author of this thesis has contributed to the writing of the draft, the visualization, the validation, the software, the methodology, the investigation, the formal analysis and the data curation.

Abstract: The periodic hills simulation case is a well-established benchmark for computational fluid dynamics solvers due to its complex features derived from the separation of a turbulent flow from a curved surface. We study the case with the open-source implicit large-eddy simulation **(ILES)** software `Lethe`. `Lethe` solves the incompressible Navier–Stokes equations by applying a stabilized continuous finite element discretization. The results are validated by comparison to experimental and computational data available in the literature for Re = 5600. We study the effect of the time step, averaging time, and global mesh refinement. The ILES approach shows good accuracy for average velocities and Reynolds stresses using less degrees of freedom than the reference numerical solution. The time step has a greater effect on the accuracy when using coarser meshes, while for fine meshes the results are rapidly time-step independent when using an implicit time-stepping approach. A good prediction of the reattachment point is obtained with several meshes and this value approaches the experimental benchmark value as the mesh is refined. We also run simulations at Reynolds equal to 10 600 and 37 000 and observe promising results for the ILES approach.

5.1 Introduction

The phenomenon of turbulent separation from a curved surface occurs in a large variety of engineering problems, such as flow over the blades of a turbine, past an obstruction in a pipe, and near an impeller in a mixing tank. Therefore, it is essential that a method capable of simulating turbulent flows is able to capture this phenomenon and the resulting flow characteristics. The periodic hills is an established simulation benchmark for flow separation [121]. In this case, a well-defined flow passes over a series of hills which repeat along a channel in a periodic fashion. As the flow passes over a hill, there is a pressure-induced separation from the curved surface. It then recirculates on the leeward face of the hill and reattaches at the base of the channel before accelerating up and over the next hill. This case includes complex flow features such as the generation of an unsteady shear layer, recirculation, strong pressure gradients, attached and detached boundary layers, and turbulence recycling due to the periodicity assumption [122].

Over the past decade, the main research focus around the periodic hills simulation case has been on developing better wall functions and subgrid-scale models for explicit Large-Eddy Simulations **(LES)** (e.g., [123–125]), with a few studies using ILES [125–130]. In the latter studies, only two use the Finite Element Method **(FEM)**: Krank et al. [130] with high-order discontinuous FEM and Wang et al. [125] with hp-spectral-FEM. In both LES and ILES approaches there is numerical dissipation, which is not the case in Direct Numerical Simulations (DNS), where all the scales are fully resolved. The main difference between LES and ILES is that in the latter, often referred to in the literature as under resolved DNS, there is no subgrid-scale model. Instead, the refinement of the mesh determines the length scales that are resolved. In general, the mesh is finer in areas of interest or where large flow variation occurs (particularly in near-wall regions), so that the smaller eddies can also be resolved. In stabilized approaches, if the cell size is not small enough to resolve all the eddies up to the scale where there is viscous dissipation, which is usually the case, additional dissipation is included numerically according to what is called a stabilization term that comprises a stabilization parameter and the strong residual of the momentum equation.

The aim of this study is two-fold: the first is to demonstrate how accurate results for the periodic hills case can be obtained using less degrees of freedom than a traditional explicit LES approach when using an ILES approach with a stabilized FEM discretisation for different Reynolds numbers. This is achieved by comparing the simulation results with two previous studies: an experimental study by Rapp [131] and a computational finite-volume explicit LES completed by Breuer et al. [132]. The second aim and main contribution is to investigate the effect of numerical parameters, such as time step, overall simulation time for averaging of flow properties, and mesh refinement, on the periodic hills simulation. The solver used in this study is implemented in the open-source multiphase flow simulation software `Lethe`, a stabilized continuous Galerkin FEM solver which uses the `deal.II` Library [133, 134]. This is the first work in the literature that studies closely the effect of such parameters when using a stabilized FEM approach and it highlights the strengths of stabilized methods for the modeling of these kind of turbulent flows when using an implicit time-stepping scheme.

The remainder of this work is organized as follows. Section 5.2 introduces the periodic hills case in detail and summarizes the previous studies available in the literature, and Section 5.3 presents the stabilized formulation, simulation parameters and benchmark data. In Section 5.4 the results of all simulations are presented and analyzed. Finally, Section 5.5 summarizes our conclusions.

5.2 Periodic Hills Case

As the flow passes over the hill, it is subjected to the effects of both the curvature of the hill and the pressure gradient. The adverse pressure gradient on the leeward side of the hill and resulting deceleration of the flow causes the boundary layer to separate from the curved hill surface. The flow then recirculates on the leeward side of the hill and reattaches in the base of the channel before the next hill. There is a short distance remaining before the subsequent hill which allows the boundary layer to recover. The flow then accelerates up and over the second hill, and the flow pattern repeats in a periodic manner. **Figure 5.1** depicts an instantaneous snapshot of the flow over periodic hills.

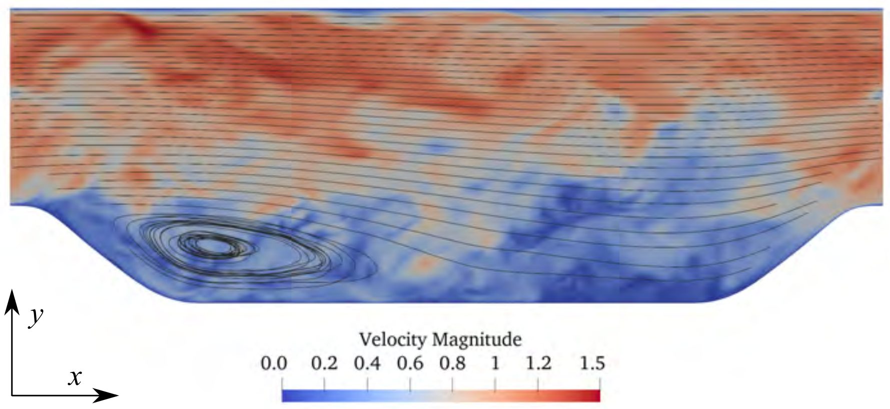

Figure 5.1 An instantaneous snapshot of the turbulent flow over periodic hills as generated by `Lethe` after 800 s of simulation run time. The recirculation zone and turbulence shed from the shear layer can clearly be seen. The black lines represent streamlines of the time-averaged flow.

5.2.1 Geometry

For a couple of decades, many experimental and computational studies have been completed over the same generalized geometry, which was first introduced by Mellen et al. [135] (for more information on this, a history of the periodic hills case is well surmised by Rapp et al. [136] and more recently by Wang et al. [125]). The initial geometry was improved over the years to ensure that the periodic hills case could be used as a benchmark for wall modeling, subgrid-scale modeling and grid parameters [132]. For example, the distance between the hills was increased to have a larger reattachment zone, and the side walls were eliminated to remove the spanwise effects on the flow. This means that there are now many studies, both experimental and computational, that can be used as benchmarks for the periodic hills case and confirm that the case has a well-defined configuration [131, 132, 137, 138]. Since most cases use the same geometry, it was also logical for our case to use this geometry (see **Figure 5.2**), allowing comparison of the simulation with both experimental and simulation data.

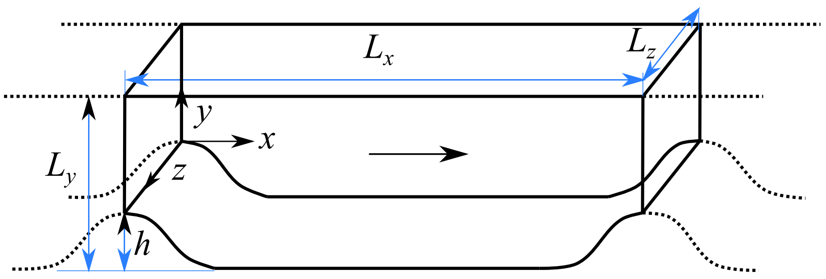

Figure 5.2 Geometry of the periodic hills test case, where $L_x = 9h$, $L_y = 3.035h$ and $L_z = 4.5h$, and h is the maximum height of the hill (adapted from [121]).

The shape of the hills is described using 6 polynomials, each defined for a sub-domain of the x domain [121] (the polynomials can be found in Appendix A in Section 5.7). The top of the first hill is located at $x/h = 0$ with an elevation of $y(x) = h$; $y(x)$ reaches a minimal value of 0 at $x/h = 1.929$. The geometry is flat in the range $x \in [1.929; 7.071]$, with the geometry mirrored at $x/h = 4.5$ (meaning the second hill has a windward face equal and opposite to the leeward face of the first hill). The gap between hills is sufficiently sized to allow the flow to reattach between hills and give some distance for recovery of the boundary layer after reattachment. Therefore, the presence of the second hill does not affect the point of reattachment.

The height and width of the channel are specified as to reduce the computational power and memory required and allow sufficient resolution in both directions. The spanwise domain of $L_z = 4.5h$ allows the side wall effects to be ignored and the spanwise fluctuations to be completely resolved excluding the largest eddies [131, 135, 137]. Flow characteristics which are half a channel width apart are uncorrelated [125].

5.2.2 Boundary Conditions

The top and bottom walls use no-slip boundary conditions, while the boundary conditions at the start ($x = 0$) and the end of the geometry ($x = L_x$) are periodic. This allows the flow regime to achieve periodicity after several flow throughs, as per the definition of the test case, and removes the complication of specifying inlet and outlet conditions. The side walls are considered to have periodic boundary conditions, which allows the model to represent the bulk flow of the channel [121].

It is worth noting that while the velocity components are explicitly periodic at the streamwise boundaries, the pressure is not. The pressure is comprised of a linear force component and a non-linear pressure component:

$$p_{\text{total}}(\boldsymbol{x}, t) = p_{\text{force}}(t) + p_{\text{dynamic}}(\boldsymbol{x}, t) = \beta(t)x + p(\boldsymbol{x}, t) \tag{5.1}$$

where β is a spatially independent pressure gradient term and p is the dynamic pressure resulting from the flow regime. The value of β is dynamically adapted over time to ensure that the specified volumetric flow rate remains constant as the flow develops, and it is included in the momentum equation as a source term. β is calculated following the procedure of Benocci et al. [139] and Wang [140]:

$$\beta^{n+1} = \beta^n - \frac{\alpha}{\Delta t}\left(\overline{U}^0 - 2\overline{U}^n + \overline{U}^{n-1}\right) \tag{5.2}$$

where, α is a relaxation coefficient to control the convergence of speed, $\overline{U}$ is the average velocity at the inlet and Δt is the time step.

5.2.3 Interesting Features

Several features of interest have been identified in the flow pattern at varying Reynolds numbers. The Reynolds number for this flow configuration is given by:

$$\text{Re} = \frac{u_B h}{\nu} \tag{5.3}$$

$$u_B = \frac{1}{2.035h}\int_h^{3.035h} u(y)\mathrm{d}y \tag{5.4}$$

where u_B is the bulk velocity, ν the kinematic viscosity and $u(y)$ the streamwise velocity profile at $x/h = 0$. Periodic hills simulations have been carried out at Reynolds numbers up to 37 000 [124, 141–143]. Most of the recent LES/DNS periodic hills studies use Re = 10595 (e.g., [122,125,130,132,143,144]). The flow regime is still relevant for lower Reynolds numbers however, with the relevant features being observed at Re = 5600, not to mention the plentiful experimental and numerical data available for this specific case summarized in Section 5.2.4. Consequently, this Reynolds number is used for the core of this study, while the higher Reynolds numbers are only investigated shortly to have a complete view of the capabilities of the ILES approach.

One main feature which has a large impact on the overall flow pattern is the point of separation. The point of separation from smoothly contoured walls is affected greatly by the properties of the flow and the environment, including pressure gradient, wall roughness, turbulence in the shear layer and transport of downstream effects [129, 136]. The point of separation has an effect on the length and height of the recirculation bubble, and hence on the reattachment point. As Re increases, the recirculation zone flattens and the reattachment point moves upstream [136]. A variation in the separation point is amongst several factors that affect the development of the flow regime in the leeward side of the hill, and so it leads to a difference in reattachment point [121]. Indeed, for the periodic hills case, it has been shown that a 1% change in separation point results in a 7% change in reattachment point [137].

The accuracy of the reattachment point is a good indicator of the accuracy of the simulation at the near-wall region. However, the turbulent and unsteady nature of the flow leads to a low frequency oscillation of the reattachment point, making it difficult to accurately determine [136]. It is usually obtained as the location on the channel flow where u changes direction (the flow stops reversing within the recirculation bubble and reattaches, flowing forward) [131], or the location on the channel floor where the wall shear stress is zero [126].

Large eddies originate from the separated shear layer and are apparent as large longitudinal rolls on the windward side of the second hill [141]. These are due to the Kelvin–Helmholtz instabilities; the difference in shear stress through the fluid lead to a rotational effect and result in vortex rolls. These elongated vortices are often less than h in diameter, but being able to model these large 3D structures indicates the need to have sufficient spanwise resolution in a simulation.

Fluctuations are observed in the z direction on the windward side of the second hill. This is attributed to the "splattering" effect. Eddies, including the larger Kelvin–Helmholtz eddies, are transported by convection towards the second hill. These eddies are compressed by the presence of the second hill, and as they can no longer continue motion in the x direction, they "splatter" outwards. This is much more noticeable in the z direction than in the y direction as the bulk flow at this location has a significant y-component as it accelerates up and over the second hill [137].

A small recirculation has also been identified at the foot of the second hill, at around $x/h = 7$. As the flow travels over the second hill, the flow accelerates and shear stress increases rapidly. Flow at the base of the second hill decelerates and can reverse, leading to a separation of the boundary layer and secondary vortices [137]. However, this separation is strongly dependent on the flow regime before the second hill and transportation of eddies. Since this varies over time, this recirculation is hardly visible in the averaged flow field [132].

Another potential small recirculation can be observed on the crest of the hill. The sudden increase of wall shear stress as the flow accelerates over the second hill is countered by a change from a favourable to an adverse pressure gradient at approximately $x/h = 8.6$, dropping the shear stress and decelerating the flow. This deceleration can lead to a small flat recirculation zone (only found in numerical simulations at $\mathrm{Re} > 10\,600$, e.g., [132, 145]). Hence, it is not expected to be observed in the `Lethe` simulations at $\mathrm{Re} = 5600$. For a more detailed review of the flow features which arise in the periodic hills simulation case, we refer the reader to dedicated previous studies such as the articles by Fröhlich [137] or by Gloerfelt et al. [138].

5.2.4 Previous Studies of the Periodic Hills Simulation Case

Table 5.1 describes the type and resolution of meshes used in previous periodic hills simulations at $\mathrm{Re} = 5600$, and the resulting reattachment points obtained, to give an indicator of the near-wall resolution. Similarly, **Table 5.2** and **Table 5.3** contain the information for studies using $\mathrm{Re} = 10\,600$ and $\mathrm{Re} = 37\,000$, respectively. Table 5.1 shows that the mesh size varies dramatically across studies of periodic hills at $\mathrm{Re} = 5600$. Meshes used tend to vary from 12.4 million cells up to 218 million cells. The spacing and number of grid points is important to resolve features within the flow completely and accurately. The finer the mesh, the closer to DNS the simulation becomes, and the greater the computational expense. Several studies do compare mesh size and then proceed with the most accurate mesh for the least computational expense [130, 141].

Table 5.1 Mesh type and resolutions used in previous studies with Re = 5600.

Code	**Type**	**Spatial method**	**Number of cells**	**DoFs**	**Mesh type**	**Reattachment point**	**Averaging time [flows through]**	**Time step [s]**	**Author**
LESOCC Case 7	LES	FVM	12.4M	-	Curvilinear	5.09	145	0.002	Breuer et al. [132]
MGLET Case 8	DNS	FVM	218M	-	Cartesian	5.14	38	0.001	Breuer et al. [132]
URDNS 5600	URDNS	DG-FEM ($k = 6$)	65K	22.5M	Curvilinear	4.82±0.09	61	Dynamic	Krank et al. [130]
DNS 5600	DNS	FEM ($k = 7$)	65K	33.6M	Curvilinear	5.04±0.09	61	Dynamic	Krank et al. [130]
Incompact3d	DNS	FD	37.8M	-	Cartesian	∼4.70	150	0.0005	Xiao et al. [146]

[a]**Note:** A dash (-) is placed where the information was not explicitly reported in the respective reference.

Table 5.2 Mesh type and resolutions used in previous studies with Re = 10 600.

Code	**Type**	**Spatial method**	**Number of cells**	**DoFs**	**Mesh type**	**Reattachment point**	**Averaging time [flows through]**	**Time step [s]**	**Author**
LESOCC	LES	FVM (cell-centered)	4.7M	-	Curvilinear	4.56	55	0.002	Fröhlich et al. [137]
STREAMLES	LES	FVM (cell-centered)	4.7M	-	Curvilinear	4.72	55	0.001	Fröhlich et al. [137]
ALDM	ILES	FVM (staggered)	4.5M	-	Cartesian	4.3	60	-	Hickel et al. [126]
LESOCC Case 9	LES	FVM	12.4M	-	Curvilinear	4.69	142	0.0018	Breuer et al. [132]
WMLES_C	ILES	FVM	200K	-	Cartesian	-	90	Dynamic	Z.L. Chen et al. [127]
WMLES_F	ILES	FVM	900K	-	Cartesian	-	40	Dynamic	Z.L. Chen et al. [127]
MDCD/SLAU	ILES	FVM	900K to 6.9M	-	Curvilinear	-	20	0.01	Li et al. [128]
High order impact	LES ILES	FVM	5.4M and 14.3M	-	Curvilinear	4.2 and 4.4 3.75 and 4.4	-	-	Balakumar et al. [129]
DRP11	LES	FD	4.2M 33.5M	-	Curvilinear	–	80	–	Gloerfelt et al. [138]
No-model and WALE	LES	DG-FEM (k=3)	65K	4.19M	Curvilinear	3.9	-	0.0001	De La Llave Plata et al. [143]
URDNS 10 600	URDNS	DG-FEM (k=5)	524K	113M	Curvilinear	4.57±0.06	61	Dynamic	Krank et al. [130]
DNS 10 600	DNS	FEM (k=6)	524K	180M	Curvilinear	4.51±0.06	61	Dynamic	Krank et al. [130]
FD	LES	FDM	67K, 524K, 4.2M and 33.5M	-	Curvilinear	-	55 to 80	Explicit	Gloerfelt et al. [122]
SEDM-Roe	LES	DFEM (k=4)	72K and 10K	9M and 1.3M	Curvilinear	4.37 and 4.18	64 and 96	Explicit	Lodato et al. [144]
SEDM-Aufs	LES	DFEM (k=4)	72K and 10K	9M and 1.3M	Curvilinear	4.21 and 4.43	64 and 96	Explicit	Lodato et al. [144]
SEDM	LES	DFEM (k=6)	72K	24.7M	Curvilinear	-	23	Explicit	Lodato et al. [144]
Nektar++	ILES	hp-FEM (k=4,7)	246K, 500K	-	Curvilinear	-	140	0.001	Wang et al. [125]

Note: A dash (-) is placed where the information was not explicitly reported in the respective reference.

Table 5.3 Mesh type and resolutions used in previous studies with Re = 37 000.

Code	**Type**	**Spatial method**	**Number of cells**	**DoFs**	**Mesh type**	**Reattachment point**	**Averaging time [flows through]**	**Time step [s]**	**Author**
HHTBLEC HHTBLEF	ILES	FVM	200K and 350K	12.4M	-	3.41 and 3.80	90 and 40	-	Z. Chen [147]
PITM	Hybrid RANS /LES	FVM	240K, 480K, 480K and 960K	-	Curvilinear	4.30, 4.26 3.54 and 3.68	-	Explicit	Chaouat et al. [141]
LES	LES	FVM	500K, 5M and 20M	-	Curvilinear	3.5, 3.65 and 3.65	140	Explicit	Mokhtarpoor et al. [142]
DLUM	RANS/LES	FVM	500K	-	Curvilinear	3.8	140	Explicit	Mokhtarpoor et al. [142]
WALE	LES	DG-FEM (k=3)	65K	4.19M	Curvilinear	3.2	-	Explicit	De La Llave Plata et al. [143]

Note: A dash (-) is placed where the information was not explicitly reported in the respective reference.

The majority of previous periodic hills simulations do not discuss or even state the time averaging or time stepping aspects of the simulations, excluding whether to consider if the CFL number is sufficiently low (an important consideration for stability in explicit time stepping methods). If they do state the averaging time and time step, often these are extremely large and small respectively. However, an overly large averaging time or overly small time step may result in a simulation running for much longer and being much more computationally costly than necessary. Additionally, due to the transient and turbulent nature of the simulation, it was theorized that these two parameters would have an effect on the results produced. Therefore, the time after which the average was taken and the time step used in the simulation were deemed to be two parameters worthy of further investigation.

To the authors' knowledge, the only two FEM ILES studies that study some of these parameters for the periodic hills case are by Krank et al. [130] and Wang et al. [125]. The time is considered by the former, who look at the time taken for averages to converge for different DNS and under resolved DNS simulations with both Re = 5600 and Re = 10 600. They also consider the convergence using *h/p* refinement for a discontinuous Galerkin ILES approach, while the latter look at the ILES requirements only for Re = 10 600, but mainly focus on the grid requirements and order of the elements, and do not consider the effect of time at all.

In summary, the majority of previous studies of the periodic hills simulation case have focused on the physical processes controlling the flow regime, or solely have demonstrated the ability of a CFD code to accurately model the simulation case, rather than focused on the parameters of the simulation itself. They hence ran the simulation case at the smallest time step and finest grid that is computationally feasible for the longest time.

Optimization of the parameters for the flow regime has barely been focused on at all; using parameters which maintain accuracy of the simulation while reducing computational expense are important for feasible simulations, particularly for application of the model to industry [148]. Therefore, the effect of the numerical parameters must be understood and so the effect of time step, averaging time and mesh resolution, are investigated within this work.

5.3 Simulation Setup

All simulations in this work were completed using the open-source software `Lethe` [105]. All relevant code required to run the simulations in this report can be found in the public Github repository for `Lethe` (`https://github.com/chaos-polymtl/lethe`). The code specific to post-processing the results obtained from the periodic hills simulations is available in the periodic hills folder within the `Lethe`-utils Github repository (`https://github.com/chaos-polymtl/lethe-utils`). `Lethe` depends on the deal.II library v9.5.0 with Trilinos, p4est, ParMetis, and MPI enabled [133,134]. In this study, the supercomputers Beluga and Niagara of the Digital Research Alliance of Canada were used.

5.3.1 Governing Equations and Numerical Model

`Lethe` solves the incompressible Navier–Stokes equations:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{5.5}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} = -\nabla p^* + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{f} \tag{5.6}$$

with

$$\boldsymbol{\tau} = \nu \left((\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T \right) \tag{5.7}$$

where $\boldsymbol{u}$ is the velocity vector, $p^* = \frac{P}{\rho}$ with P the pressure and ρ the density, $\boldsymbol{\tau}$ the deviatoric stress tensor, ν the kinematic viscosity and $\boldsymbol{f}$ a body force. Being non-linear partial differential equations, they must be discretised in space and time in order to approximate a solution. For this a continuous Galerkin Finite-Element formulation is used along with a SUPG (Streamline-Upwind/Petrov-Galerkin)/PSPG (Pressure-Stabilizing/ Petrov-Galerkin) stabilization approach. This allows the use of equal-order finite elements for the pressure and velocity components and avoids numerical oscillations for advection-dominated problems [105, 149, 150]. The following weak formulation for the Navier–Stokes equations is obtained:

$$\int_\Omega \nabla \cdot \boldsymbol{u} q d\Omega + \sum_K \int_{\Omega_k} \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p^* - \nabla \cdot \boldsymbol{\tau} - \boldsymbol{f} \right) \cdot (\tau_u \nabla q)\, d\Omega_k = 0 \tag{5.8}$$

$$\begin{aligned} &\int_\Omega \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \boldsymbol{f} \right) \cdot \boldsymbol{v} d\Omega + \int_\Omega \boldsymbol{\tau} : \nabla \boldsymbol{v} d\Omega - \int_\Omega p^* \nabla \cdot \boldsymbol{v} d\Omega \\ &+ \sum_K \int_{\Omega_k} \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p^* - \nabla \cdot \boldsymbol{\tau} - \boldsymbol{f} \right) \cdot (\tau_u \boldsymbol{u} \cdot \nabla \boldsymbol{v})\, d\Omega_k = 0 \end{aligned} \tag{5.9}$$

where $\boldsymbol{v}$ and q are the test functions for velocity and pressure, respectively, and K is the total number of elements. Since the problem is transient, the stabilization parameter τ_u takes the following form:

$$\tau_u = \left[\left(\frac{1}{\Delta t} \right)^2 + \left(\frac{2\|\boldsymbol{u}\|}{h_{\text{conv}}} \right)^2 + 9 \left(\frac{4\nu}{h_{\text{diff}}^2} \right)^2 \right]^{-1/2} \tag{5.10}$$

where Δt is the time step, h_{conv} and h_{diff} are the size of the element related to the convective transport and diffusion mechanism, respectively [149, 149]. In `Lethe`, both element sizes (h_{conv} and h_{diff}) are set to the diameter of a sphere of a volume equivalent to that of the cell [105, 150, 151]. The full definition of the methods and demonstration of the order of accuracy of this CFD solver are detailed in a separate publication [105]. For the simulations in this study, Newton's method is used to solve implicitly the non-linear problem. Each linear system of equations is solved with an ILU preconditioned GMRES solver. A second-order backward difference implicit scheme (BDF2) is used for time stepping [76].

5.3.2 Simulation Parameters and Mesh

The flow geometry is used as described in Section 5.2.1 and the boundary conditions as specified in Section 5.2.2. The height of the hill (h) is set equal to 1 for simplicity. Likewise, the volumetric flow rate and kinematic viscosity are set to be $9.1575\,\mathrm{m^3\,s^{-1}}$ and $1.785\,71 \times 10^{-4}\,\mathrm{m^2\,s^{-1}}$ respectively, so that the bulk velocity is $u_B = 1\,\mathrm{m\,s^{-1}}$, and a Reynolds number of 5600 can be maintained. The kinematic viscosity is set to $9.433\,96 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$ for a Reynolds number of 10 600 and to $2.7027 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$ for a Reynolds number of 37 000. For all simulations, first-degree Lagrange elements (Q_1) are used for both pressure and velocity.

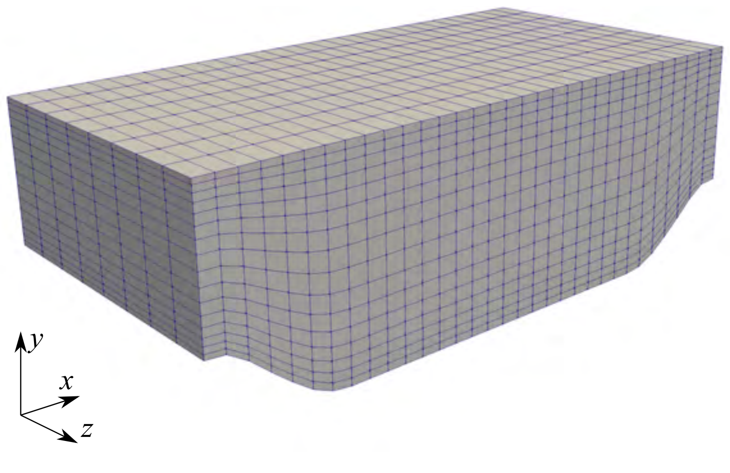

Figure 5.3 Example of the curvilinear mesh used in `Lethe`. This mesh contains only 4K cells for the purpose of visualization. However, the coarsest mesh used in the results section contains 120K cells.

The elements of the mesh are isoparametric hexahedra and are arranged on a curvilinear mesh (see **Figure 5.3**). The mesh used for all simulations is a static, uniformly refined mesh. In the x and z directions, the elements are of equal width across the domain. In the y direction, the spacing of the grid points can be varied, allowing the mesh to become finer as the elements approach the wall. For all meshes used in this study, the ratio between the longest to the shortest dimensions of a cell located in the middle of the geometry never exceeds a value of two.

The quality of the grid is as important as the number of grid points in order to accurately locate features of interest and reduce numerical errors. The resolution of the mesh in the near-wall region is evaluated using the dimensionless distance from the wall Δy^+ which gives an indication of how fine the mesh is in the near-wall region:

$$\Delta y^+ = \frac{y_{cc} u_\tau}{\nu} \tag{5.11}$$

where y_{cc} is the half of distance from the wall to the wall-nearest grid point, ν is the kinematic viscosity and u_τ is the friction velocity given by:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{5.12}$$

where τ_w corresponds to the wall-shear stress and ρ to the density. The spanwise and lengthwise cell lengths Δx^+ and Δz^+ are also important for resolving near the wall and can be calculated in the same fashion by replacing y_{cc} with the appropriate coordinate. The average and maximum values for $\Delta x^+, \Delta y^+$ and $\Delta z+$ over x/h for all meshes used in this study, are reported in Appendix B in Section 5.8.

Apart from the y^+ criteria, that is in fact commonly used in the LES and RANS domains, no other criteria in the literature of stabilised methods were found to assess the quality of the mesh *a posteriori*. According to the literature, simulations of attached boundary layers are not precise for traditional LES approaches if the nearest computed values are not located within the viscous sub-layer ($\Delta y^+ < 5$) [30]. In the periodic hills literature, the recommended ranges for wall-resolved LES simulations are defined as: $\Delta x^+ \approx 50 - 100$, $\Delta y^+ \approx 1$ and $\Delta z^+ \approx 15 - 30$ [122, 137]. In this study, none of the meshes have coordinate resolutions within these ranges. In particular, the average values for Δx^+ and Δz^+ always have significantly higher resolution. In a publication by Krank et al. [130], where a DNS simulation was performed, a value of $\Delta y^+_{max} < 0.86$ and $\Delta x^+_{max} = \Delta z^+_{max} = 7.2$ was reported. This indicates that the mesh resolutions used in this study are, in fact, in between DNS and wall-resolved LES, which is exactly the definition of implicit LES when it comes to mesh resolution.

5.3.3 Comparison Data

In order to verify the accuracy, stability and reliability of this ILES approach in the periodic hills case, the results are compared to established test data from both experiments and other CFD simulation. Experimental data is obtained from Rapp [131], and results from the LESOCC CFD code performed by Breuer et al. [132] provide benchmark computational data. Since data is provided for cross-section at varying x/h values in both benchmarks, the results from the `Lethe` simulation must also be extracted at these points to allow comparison of the data.

Experimental Data

A series of experiments ran by Rapp [131] provide an experimental benchmark case. Not all benchmark data can be obtained from a physical experiment - notably the data obtained by Rapp does not contain values of the turbulent kinetic energy or the spanwise Reynolds normal stresses. The experimental set up involved a series of 10 hills with the curvature and hill spacing described in Section 5.2.1 with hill height $h = 50\,\text{mm}$. The channel height was kept the same, but the channel span was increased to $18h$ in order to sufficiently neglect side wall effects.

Results were collected using Particle Image Velocimetry (PIV) and verified against Laser Doppler Anemometry (LDA) measurements. Piezoelectric pressure probes were used to allow non-intrusive pressure measurement. Results were corrected for error and signal noise. These conscientious, well-considered measurement techniques used give high confidence to the results. The reattachment point was determined at $x/h = 4.83$.

Periodicity of the flow was confirmed by comparing flow between hills 6 and 7 and hills 7 and 8, and the reference data taken from between hills 7 and 8. While it is confirmed that the flow can be assumed to be homogeneous (statistically 2D), the periodicity at $\text{Re} = 5600$ can only be proven to a certain extent due to limitations inherent to the measurement equipment. However, Rapp [131] concluded that the data produced is precise enough to be used to develop better LES models.

Rapp also collected data for higher Reynolds numbers ($\text{Re} = 10\,600$ and $\text{Re} = 37\,000$) using the same set up and experimental techniques. This data is chosen for comparison with our results obtained using the higher Reynolds numbers in Section 5.4.4. Measuring properties near to the wall is specially challenging in these cases as the boundary layer thickness is very small, however, the periodicity of the flow could be proven completely in both cases [131]. The experimental reattachment point was determined to be equal to $x/h = 4.21$ and to $x/h = 3.76$ for Re = 10 600 and Re = 37 000, respectively.

Computational data

An established benchmark test case for computational periodic hills simulations was created by Breuer et al. [132] using the FVM code LESOCC. LESOCC solves the incompressible Navier–Stokes equations using the dynamic Smagorinsky model as sub-grid scale model. A range of simulations were performed by Breuer et al.; the first results of interest arise from Case 7, which was performed at Re = 5600 using a LES approach. This setup uses around 12.4 million active cells (13.1 million grid points) in the mesh with a time step of 0.002 s and the average taken over a time period of 1300 s.

The reattachment point for the benchmark case is 5.09, which is longer than the value of 4.83 given experimentally by Rapp. Since the reattachment point describes the overall performance of a simulation in one number [130], these values can be used as indicators for the method's performance. Breuer et al. note that the pressure distribution in the computational benchmark data is slightly under-predicted due to the SGS model, leading to a delay in the separation point and hence to an over-prediction of the reattachment point. In general, previous implicit LES studies appear to give better reattachment point agreement to the Rapp data than to the Breuer data (e.g., [125]), but in the bulk of the flow the profiles are closer to those of Breuer data (e.g., [130]). Therefore, while both benchmarks do not give precise values, they give a clear indication of the region the reattachment point should fall within.

The second results of interest are known as LESOCC Case 9 and correspond to the Reynolds number of Re = 10 600. The grid used for this case was the same as in the previously explained simulation, but a time step of 0.0018 s and an average taken over a time period of 1300 s were used. The numerical reattachment value was determined to be $x/h = 4.69$ in this case.

5.4 Results and Discussion

A baseline simulation is run to validate the results against the computational and experimental data sets presented in Section 5.3.3. Then, different meshes are considered when investigating the effects of time step and averaging time, followed by some simulations using higher Reynolds numbers. In **Table 5.4**, a summary of the different simulations and their parameters is presented. The computational times of all simulations in core hours are presented in Appendix C in Section 5.9.

Table 5.4 Summary of the simulations performed and respective parameters.

Case	**Mesh**	**Re**	**Time step** [s]	**Time average** [s]
Baseline	Coarse Regular Fine	5600	0.1	1000
Time step	Coarse Regular Fine	5600	0.0125, 0.025, 0.05, 0.1	1000
Time averaging	Coarse Regular Fine	5600	0.025	500 to 1100
High Reynolds numbers	Very coarse Coarse Intermediate	10 600 37 000	0.025	800

[a]**Note:** (cells, DoFs) of each mesh: very coarse (~120K, 500K), coarse (~250K, 1.1M), intermediate (~500K, 2.4M), regular (~1M, 4.5M) and fine (~4M, 16M).

5.4.1 Baseline

The baseline simulation ensures that the results accurately reproduce the physical phenomenon that occurs in the periodic hills case. It uses the coarse, regular, and fine mesh, a time step of 0.1 s, and the average is taken between 207 s and 1000 s (for an averaging period of 793 s or 88 flow throughs). This averaging period is larger than most of the simulations shown in Table 5.1 and is studied in detail in Section 5.4.3.

Considering the average velocity profile in the x direction (**Figure 5.4**), there is a good agreement of the `Lethe` data with both benchmarks in the bulk of flow. At the upper and lower walls, the `Lethe` data exceeds the benchmarks but retains shape at all x values similarly to the LESOCC simulation. The y velocity and Reynolds stresses also agree well with the benchmark data; for the Reynolds normal stress in x direction see **Figure 5.5**. The Reynolds stresses are more sensitive than the average velocity, but overall, there is a good accuracy of the prediction with minimum discrepancies between the meshes.

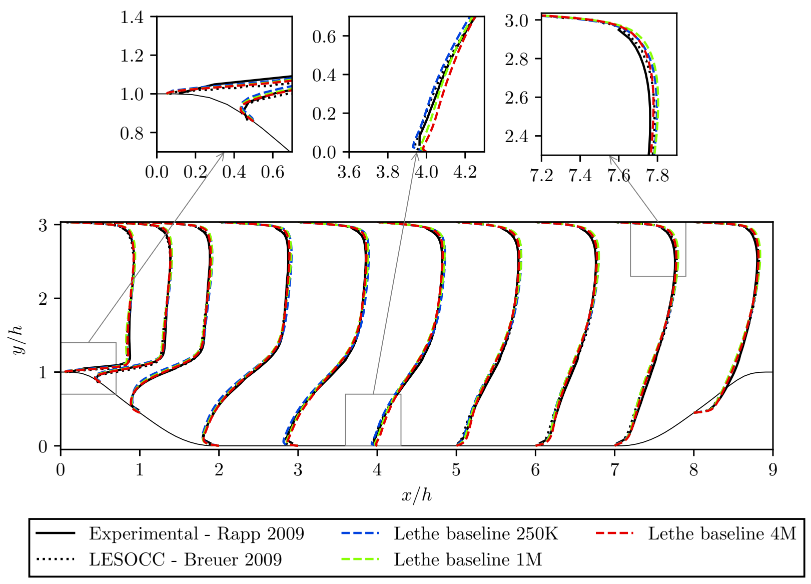

Figure 5.4 Average velocity in the x direction throughout the geometry at Re = 5600, compared against the benchmarks. The profiles are scaled by a factor of 0.8 for ease of visualization.

The reattachment point was determined to be 4.73 with the coarse mesh, 4.40 with the regular mesh, and 4.35 with the fine mesh; all of them shorter than both the Rapp and the Breuer values (4.83 and 5.09 respectively). In the literature, the reattachment point has been shown to be constantly under-predicted by overly coarse meshes [125], however, as we are using a stabilized method, in this case other parameters are playing an important role as well, such as the time step and the averaging period for the estimated quantities. Therefore, these parameters along with the mesh are investigated in the following sections.

5.4.2 Time Step

To study the effect of the time step in the simulation, four simulations per mesh were completed using `Lethe`. Again, the coarse, the regular and the fine meshes were used, and the time steps were defined by sequentially halving the value of the time step from $\Delta t = 0.1\,\mathrm{s}$ to $\Delta t = 0.0125\,\mathrm{s}$. We recall that the BDF2 scheme is second-order accurate in time. Average results were taken again after 1000 s.

The average velocity profiles in x are presented in **Figure 5.6**. For all the meshes, we obtain results that are very similar to the experimental and numerical benchmarks. The biggest difference can be observed again in the near-wall region. Looking at the zoom-in plots, it is possible to observe that, in the case of the coarse mesh, there is a high discrepancy between

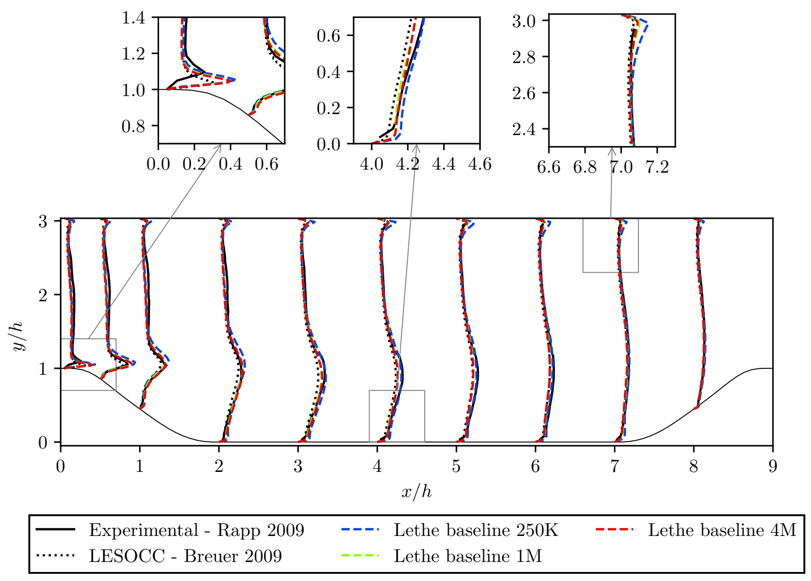

Figure 5.5 Reynolds normal stress in the x direction throughout the geometry at Re = 5600, compared against the benchmarks. The profiles are scaled by a factor of 5 for ease of visualization.

the results corresponding to the different time steps. While for the regular and fine mesh, this difference between the results is reduced. The stresses demonstrate this trend most significantly (see **Figure 5.7** and **5.8**). In the case of the fine mesh, all the time steps with the exception of the largest one converge towards the numerical solution of Breuer et al. [132].

These results can be analyzed taking into account the stabilization term in the FEM formulation, which comprises two components: i) the stabilization parameter τ, which in turn considers the time step Δt and the cell size h, and ii) the residual of the strong form of the momentum equation. In the case of the coarse mesh, as the time step is reduced, so is the dissipation or stabilization, which in general leads to a deterioration of the accuracy in coarse meshes. This phenomenon has been observed in the literature, e.g., in the articles by Hsu et al. [152], Calderer et al. [153] and Gamnitzer et al. [154]. In the case of the fine mesh, the norm of the strong residual is smaller, which reduces the effect of the stabilization, and leads to accurate and similar results for all the time steps. For the case where Re = 5600, we observe that using a stabilized formulation along with an implicit scheme allows us to refine the mesh and use a time step as large as 0.05 without losing the accuracy of the solution.

Considering the reattachment points in **Table 5.5**, it can be seen that the value of the reattachment point increases as the time step decreases when the coarse mesh is used. For the regular mesh, decreasing the time step means that the value converges more towards the experimental value (4.83) by Rapp [131], as the stabilization is affected by the refinement of

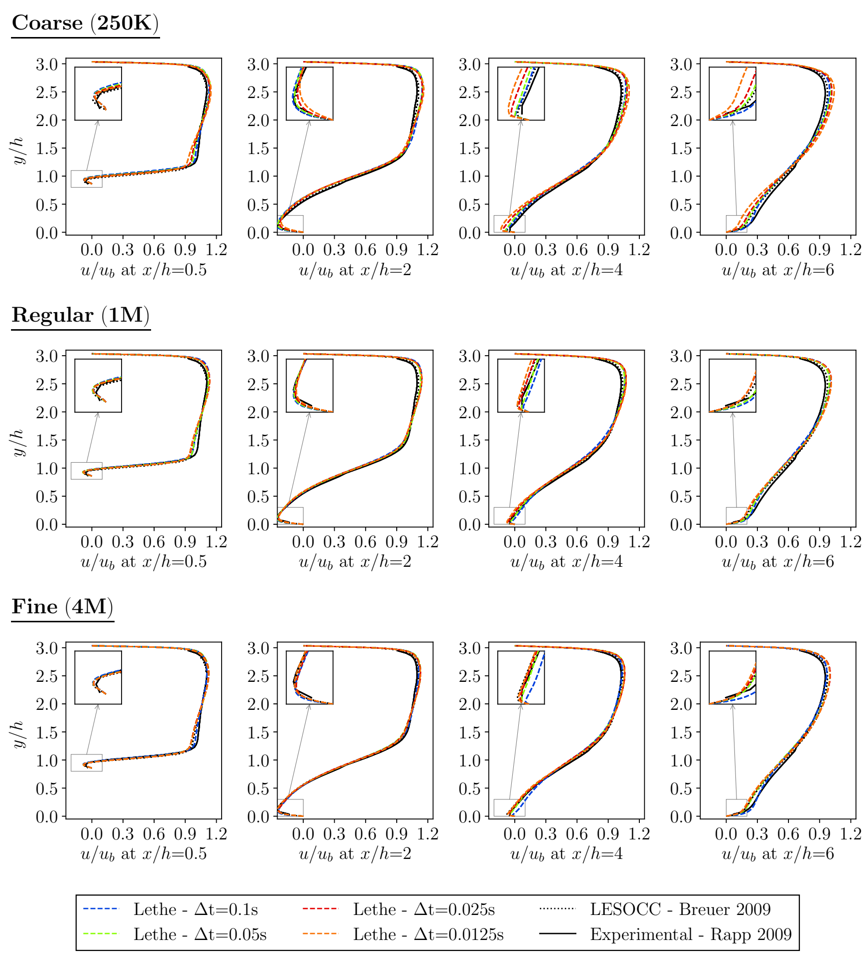

Figure 5.6 Average velocity in the x direction at different points of the geometry with Re = 5600, compared against the benchmarks.

the mesh. Hence the reattachment point becomes more accurate as the time step decreases. Finally, in the case of the fine mesh, all the time steps, apart from the coarsest one, obtain a value that is nearer to that obtained experimentally.

In this study, we use an implicit time-stepping scheme where the CFL condition is not necessary, allowing a much greater time step to be used stability-wise compared to an explicit time-stepping scheme. For stability with an explicit time-stepping method, a CFL number less than 0.8 is required in the periodic hills case as reported by Mokhtarpoor et al. [124], which explains why the time step used is so small in other studies since they mostly use explicit methods. We report the CFL number of each simulation in Table 5.5. It is worth noting that for the fine mesh, only the smallest time step fulfills the usual CFL requirement for this case, while all the others do not but still produce accurate results. This indicates that it is possible to use fine meshes and large time steps, without affecting the accuracy of the

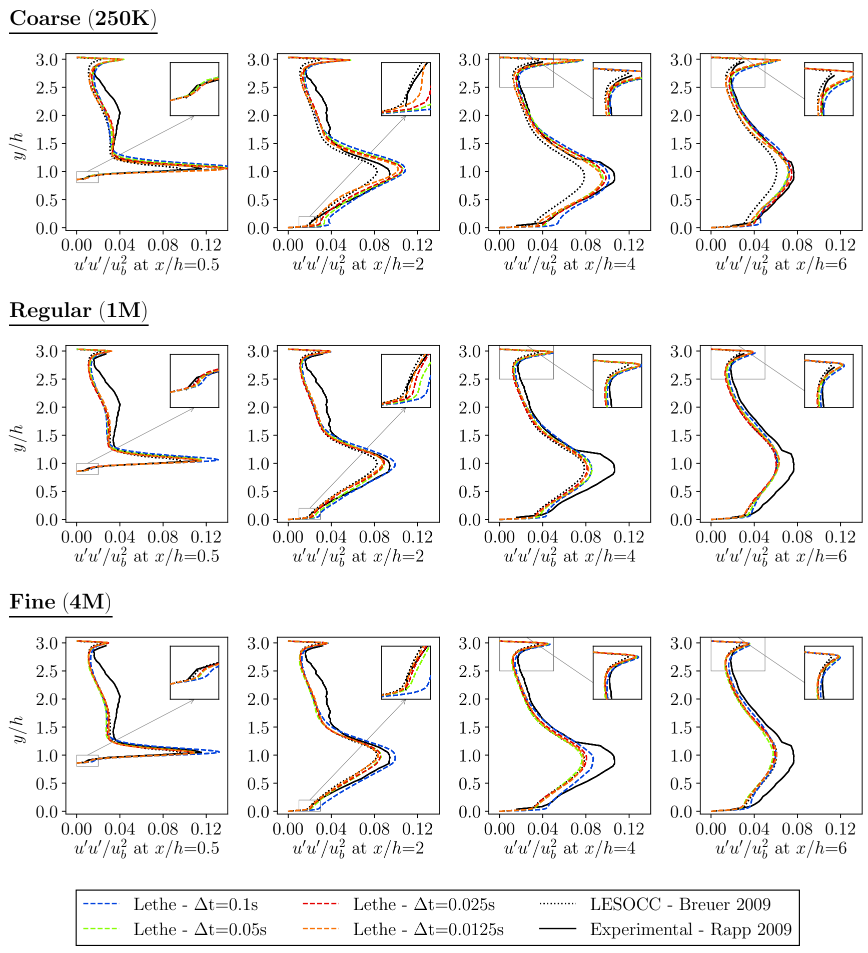

Figure 5.7 Reynolds normal stress in the x direction at different points of the geometry with Re = 5600, compared against the benchmarks.

implicit LES approach. A detailed comparison between explicit and implicit time-stepping schemes in terms of computational cost of the solution is out of the scope of this study. However, it is important to keep in mind that iterations are generally computationally more demanding when using an implicit scheme.

In conclusion, for coarse meshes, reducing the time step leads to a reduction of the accuracy of the average velocity and Reynolds stresses, with a higher impact in the latter for the bulk of the flow. In the case of fine meshes, a similar accuracy is obtained for all the time steps with the exception of the coarsest one. This opens up the possibility of using high CFL values when simulating complex cases.

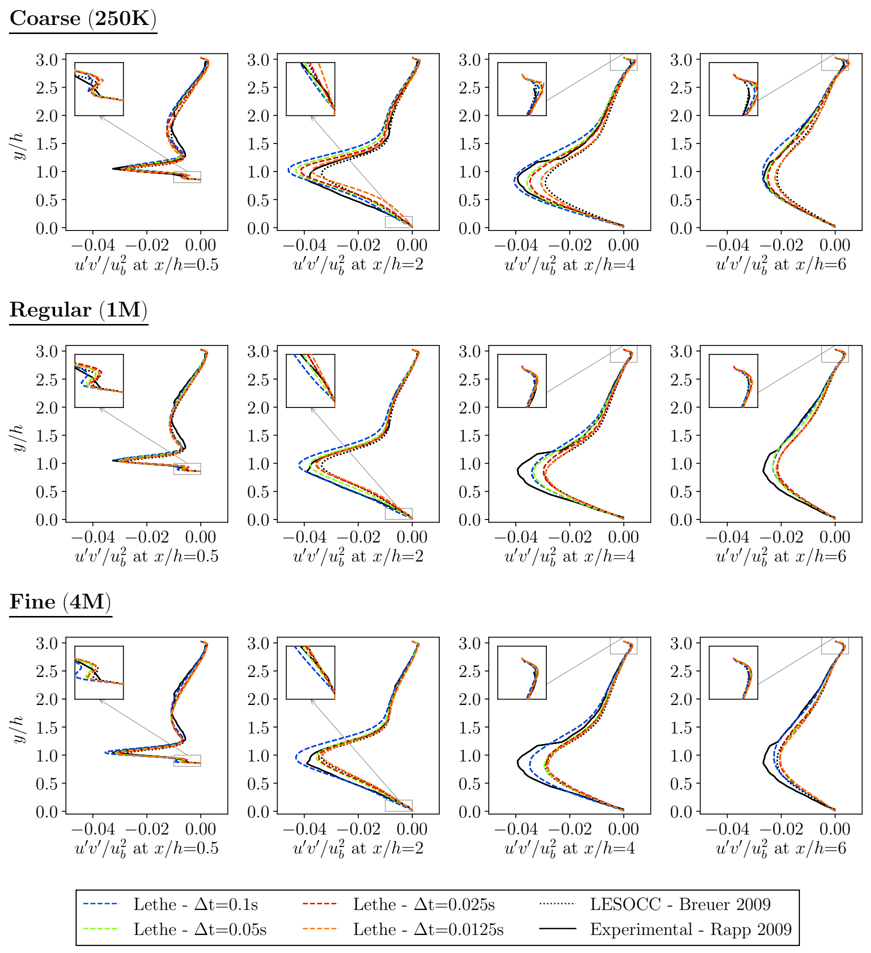

Figure 5.8 Reynolds shear stress at different points of the geometry with Re = 5600, compared against the benchmarks.

5.4.3 Time Averaging

For the periodic hills simulation, the average velocities and Reynolds stresses are taken over time. The values used in the averaging process are taken after the time exceeds 207 s, or after 23 hills have been passed, and so the flow can be considered as periodic. For the reference numerical data, the averaging period used was of 145 flow-through times while in the experimental set up a total of 10 hills were considered. In previous studies, the averaging time period varied by approximately one order of magnitude [130], with no clear consistency on how that time period was decided. In fact, Krank et al. [130] stated that the averaging period suggested by some studies for this case is of around 1000 flow-through times, which is very long and not feasible. Therefore, we decided to study the effect of the time elapsed on the convergence of the average values. For this purpose, we again used the coarse, the

Table 5.5 Near-wall parameters and CFL at varying time steps for an averaging time of 1000 s. The experimental reattachment points are: 4.83 for the experimental benchmark by Rapp [131] and 5.09 for the numerical benchmark by Breuer et al. [132].

	Coarse (250K)		**Regular (1M)**		**Fine (4M)**	
Time step	**RP**	**CFL**	**RP**	**CFL**	**RP**	**CFL**
0.1	4.73 ± 0.05	≈ 2.2	4.40 ± 0.05	≈ 3.2	4.35 ± 0.4	≈ 5.1
0.05	4.87 ± 0.05	≈ 0.9	4.59 ± 0.05	≈ 1.6	4.74 ± 0.4	≈ 2.8
0.025	5.07 ± 0.05	≈ 0.5	4.80 ± 0.05	≈ 0.7	4.83 ± 0.4	≈ 1.2
0.0125	5.37 ± 0.06	≈ 0.2	4.88 ± 0.05	≈ 0.4	4.85 ± 0.4	≈ 0.6

[a]**Note:** RP corresponds to reattachment point.

regular and the fine mesh, and data was extracted at varying times to see the convergence. The lowest averaging period considered was of 500 s or 44 flow throughs and the largest averaging period was 1000 s or 88 flow throughs. To have a fair comparison of the effect of the averaging time for these three meshes, a time step of $\Delta t = 0.025$ s is chosen due to the observations of the previous section.

According to **Figure 5.9**, **5.10** and **5.11**, the results for all the meshes are independent of the length of the time-averaging period. The difference between the results for different time-averaging period is minimal in all cases. The largest differences can be observed near to the walls due to the zoom-in plots and it is slightly more evident when taking a look to the Reynolds stresses. To answer the question of how the time-averaging affects the prediction of the reattachment point, this value was extracted at varying averaging times for each mesh and plotted, as per the method by Krank et al. [130]. This method plots the reattachment points against the averaging time (in number of flows through times). The error e is evaluated using the following expression $e = \pm c/\sqrt{T_f}$, where c is a manually defined constant and T_f is the number of flows through times. Since the reattachment point oscillates due to the turbulence, a final reattachment point is set to a value that the reattachment points extracted tend towards. The constant c is then manually adjusted so that the error is as small as possible with all the extracted reattachment points lying within the error range. The error bandwidth hence decreases as the averaging time increases. The reattachment points converge very quickly, with the expected oscillation at different averaging times being within a very small error range.

The results in **Figure 5.12** show that as we refine the mesh, the reattachment points approach the experimental reattachment point value by Rapp [131]. For this plot, we also added

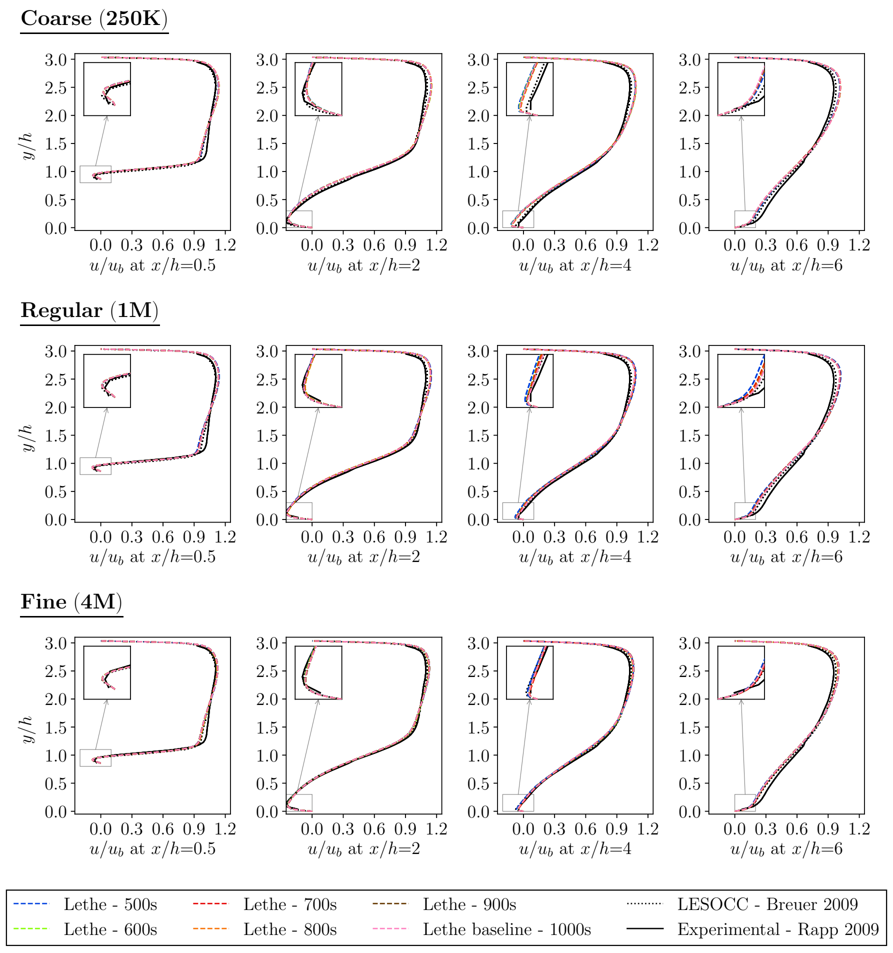

Figure 5.9 Average velocity in the x direction at different points of the geometry with Re = 5600, compared against the benchmarks.

the results for an intermediate mesh with 500K cells, since it allows us to see that the results are not monotonically approaching the experimental value as the mesh is refined, which is in agreement with the oscillatory nature of the physical phenomena. The reattachment point obtained by Breuer et al. [132] is in the error bandwidth of the reattachment points obtained using the coarse mesh in this study. In the literature, Krank et al. [130] used this methodology to compare two configurations: a DNS simulation with 65K cells and 33.6M DoFs and an URDNS simulation with 65K cells and 22.5M DoFs. For the latter, they observed that the reattachment point was closer to the experimental value by Rapp et al. [131] but away from the DNS simulation. This study shows a very clear trend towards the experimental reattachment point as we increase the mesh resolution along with a reduction of the error bandwidth. It can also be observed that again, the bandwidth of all the meshes is reduced significantly after the averaging for 500 s and 600 s, which reassures that a minimum of 700 s

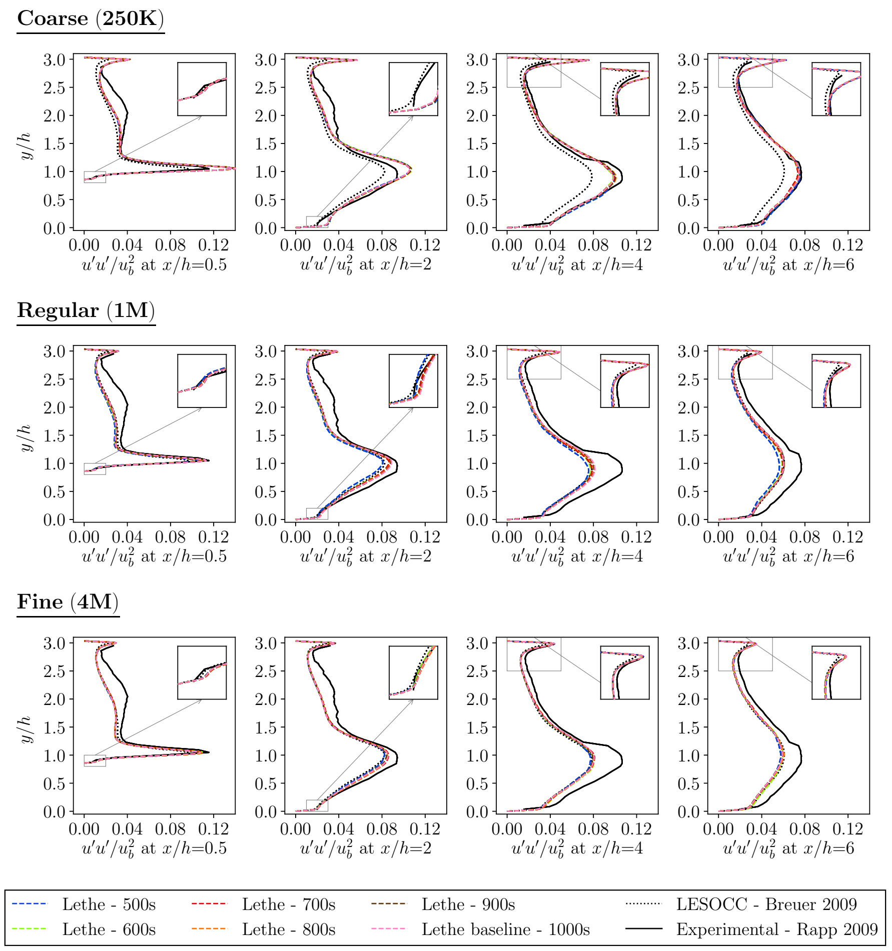

Figure 5.10 Reynolds normal stress in the x direction at different points of the geometry with Re $= 5600$, compared against the benchmarks.

is required to accurately predict specific quantities.

5.4.4 Higher Reynolds Numbers

As mentioned in Section 5.2.4, results with higher Reynolds numbers are available in literature, therefore, we tested our ILES method with SUPG/PSPG stabilization to see how well it can predict important flow properties at Re $= 10\,600$ and Re $= 37\,000$. For this, three meshes are considered: very coarse, coarse and intermediate. A time step of $\Delta t = 0.025\,\mathrm{s}$ and a time average of $800\,\mathrm{s}$ are used in accordance with the findings of the previous sections.

The results for Re $= 10\,600$ are compared to the experimental and computational data by Rapp [131] and Breuer et al. [132], respectively. In the case of Re $= 10\,600$, a good accuracy of the average velocity in the x-direction is obtained for all meshes, with the intermediate

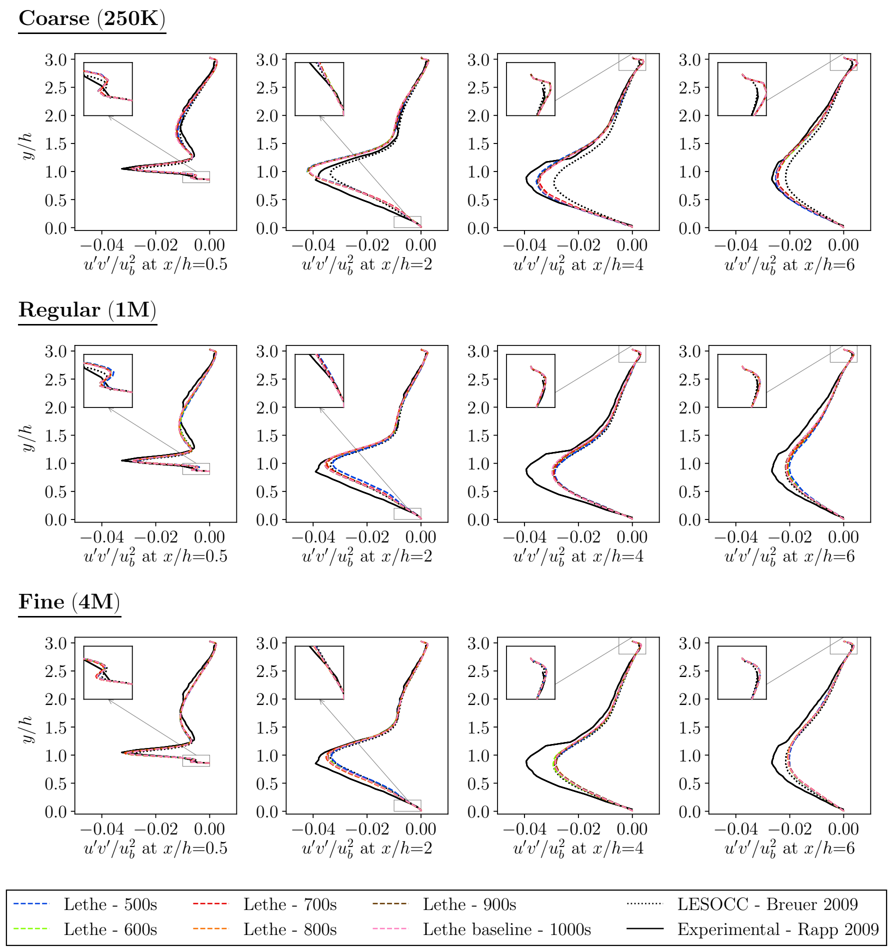

Figure 5.11 Reynolds shear stress at different points of the geometry with Re = 5600, compared against the benchmarks.

mesh having closer results to the experimental data in the bulk of the flow and near to the wall (see **Figure 5.13**). The reattachment points obtained for this case using the different meshes, from very coarse to intermediate, are $x/h = 4.54, 4.90$ and 4.01. According to the results obtained by Rapp ($x/h = 4.21$), the point of reattachment moves upstream with increasing Reynolds number as the recirculation zone flattens, which is what we observe in our results. In addition, the first two values are very close to the reattachment point obtained by Breuer et al. ($x/h = 4.69$). The prediction of the reattachment point is more sensitive to the time-averaging period; hence, it is possible that a larger period is required to obtain more accurate results in the case of this Reynolds number. For the Reynolds stresses (see **Figure 5.14**) all meshes tend to overpredict the stresses throughout the channel, again with the intermediate mesh obtaining closer results to the experimental data.

The results for the case with Re = 37 000 were only compared with the experimental bench-

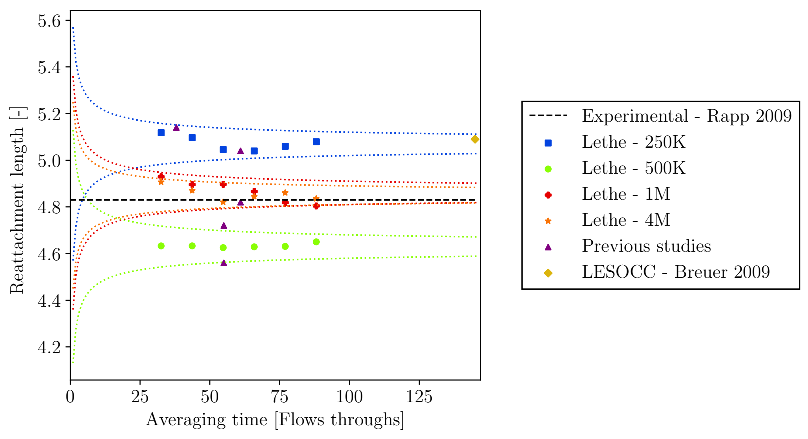

Figure 5.12 Reattachment point extracted at different averaging times for the coarse (250K), intermediate (500K), fine (1M) and very fine (4M) meshes, where the dotted line shows the error in the reattachment point over time.

mark, as Breuer et al. did not conduct any study using LESOCC for this Reynolds number. We see a similar trend on the estimation of the average velocity and the Reynolds stresses, however, with a higher discrepancy of the overall results in the bulk of the flow and near the wall. In **Figure 5.15**, there is an underprediction of the velocity in the lower region of the channel and an overprediction on the upper region, with a higher difference near the walls. A higher difference between the results of the different meshes can be observed in the results for the Reynolds stresses (see **Figure 5.16**), where they are all overpredicted, however, the intermediate mesh is the closest one to the experimental results. As pointed out by Rapp [131], to accurately predict the high near-wall peak obtained with this Reynolds number is of utter importance to have a correct prediction of Reynolds stresses at the windward side of the hill.

The reattachment points obtained in this case were $x/h = 4.20, 4.09$ and 3.49 for the three meshes from very coarse to intermediate. Experimentally, the reattachment point was determined to be equal to 3.76 [131]. Therefore, refining the mesh seems to help to have a better resolution of the flow properties allowing the reattachment point to move upstream, while introducing less dissipation. However, as in the previous case, further refinement of the mesh might lead to less time-dependent results and better accuracy of the predictions. This section demonstrates the capabilities of this ILES approach to simulate higher Reynolds numbers for turbulent flows with complex characteristics and the need of understanding the influence of all the parameters involved in the FEM formulation to be able to improve the predictions and understand phenomena that are not easy to observe experimentally.

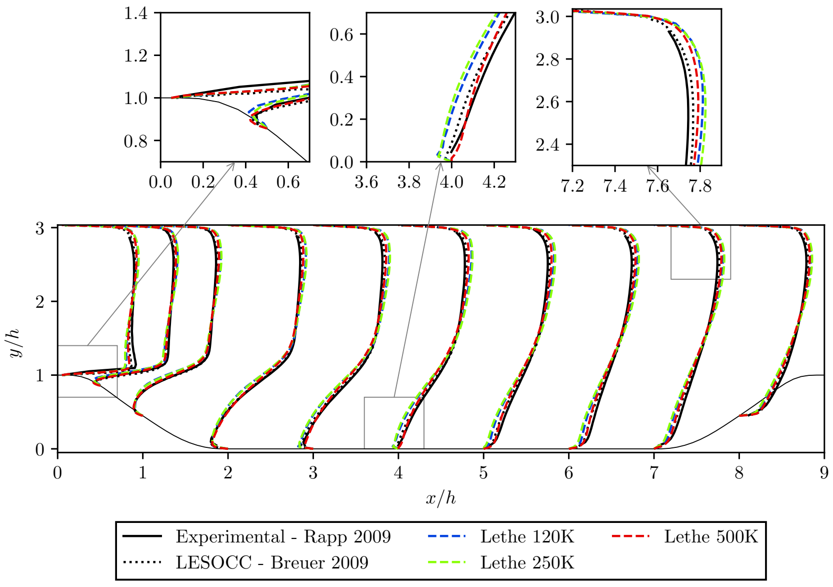

Figure 5.13 Average velocity in the x direction throughout the geometry at Re = 10 600, compared against the benchmarks. The profiles are scaled by a factor of 0.8 for ease of visualization.

5.5 Conclusion

Multiple simulations of the flow over periodic hills at Re = 5600 have been completed using a stabilized finite element ILES approach implemented in the open-source software `Lethe`. The use of this numerical method to model this simulation case was validated by comparison with two previous studies, meeting the first objective of this study. In general, the quality of the prediction depends on three factors: the mesh, the time step (along with the type of time-stepping scheme), and the time averaging period used to obtain the turbulent statistics. We demonstrated that it is possible to simulate these kinds of flows using coarse meshes for the prediction of average quantities or specific values, such as the reattachment points, but special attention needs to be taken when choosing the time step as it significantly affects the predictions when using a stabilized approach. When a finer mesh is used, along with an implicit time-stepping scheme, larger time steps (higher CFL values) than those typically used in periodic hills studies with explicit schemes can be used without losing accuracy both in the bulk of the flow and in the region near to the wall. This does not necessarily imply that the implicit scheme is computationally more efficient, since it requires the full solution of a non-linear system of equations.

The results for the reattachment points extracted from different simulations using different meshes approach the value given by the experimental benchmark as the mesh is refined. The

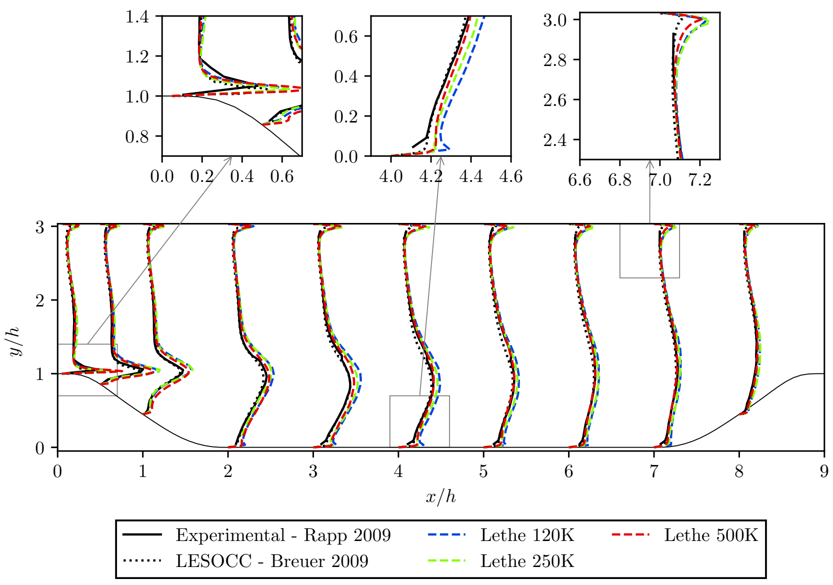

Figure 5.14 Reynolds normal stress in the x direction throughout the geometry at Re = 10 600, compared against the benchmarks. The profiles are scaled by a factor of 5 for ease of visualization.

values for a 1M cells mesh and a 4M mesh are very close, indicating that mesh-independent results were obtained. The method was also tested for Re = 10 600 and Re = 37 000 using very coarse meshes. Although the ILES approach used in this study obtains accurate results for the velocity for Re = 10 600, the Reynolds stresses are strongly affected near the separation and post-reattachment zones. The results have greater discrepancies at Re = 37 000, where significant differences are observed not only at the wall but also in the bulk of the flow. These results could be further improved by further reducing the size of the cells.

To conclude, this study not only has provided a greater understanding of how the numerical parameters affect the simulation results for the periodic hills case, but also shows that ILES methods are able to provide very good solutions for these types of complex turbulent flows with coarse and fine meshes that are coarser than the ones used in the literature. It also highlights the advantages of an implicit time-stepping scheme over an explicit one in the context of stabilized methods. The ILES methods are promising for practical simulations as they provide accuracy, do not require the calibration of a subgrid scale model, and can reduce the computational effort, in terms of mesh size and degrees of freedom of the numerical system, in comparison to traditional LES approaches.

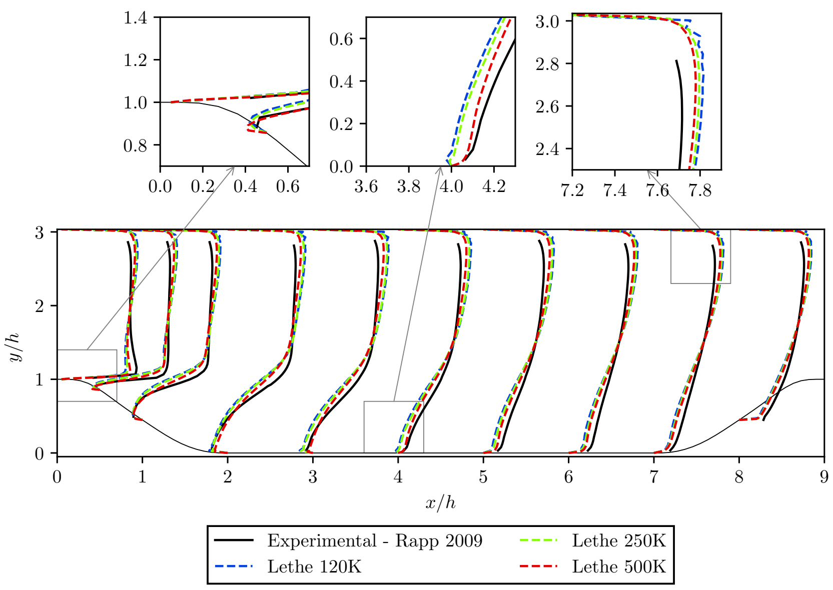

Figure 5.15 Average velocity in the x direction throughout the geometry at Re = 37 000, compared against the benchmarks. The profiles are scaled by a factor of 0.8 for ease of visualization.

5.6 Acknowledgments

The authors would like to acknowledge the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) through the RGPIN-2020-04510 Discovery grant. The authors would also like to acknowledge technical support and computing time provided by the Digital Research Alliance of Canada and Calcul Québec.

5.7 Appendix A : Geometry of Hills

The 6 polynomials used to describe the shape of the hill are:

1. For $x \in [0; 0.3214h]$:
   $y(x) = \min(h; h + 0hx + 2.420h \times 10^{-4}x^2 - 7.588h \times 10^{-5}x^3)$

2. For $x \in [0.3214h; 0.5h]$:
   $y(x) = 0.8955h + 3.484h \times 10^{-2}x - 3.629h \times 10^{-3}x^2 + 6.749h \times 10^{-5}x^3$

3. For $x \in [0.5h; 0.7143h]$:
   $y(x) = 0.9213h + 2.931h \times 10^{-2}x - 3.234h \times 10^{-3}x^2 + 5.809h \times 10^{-5}x^3$

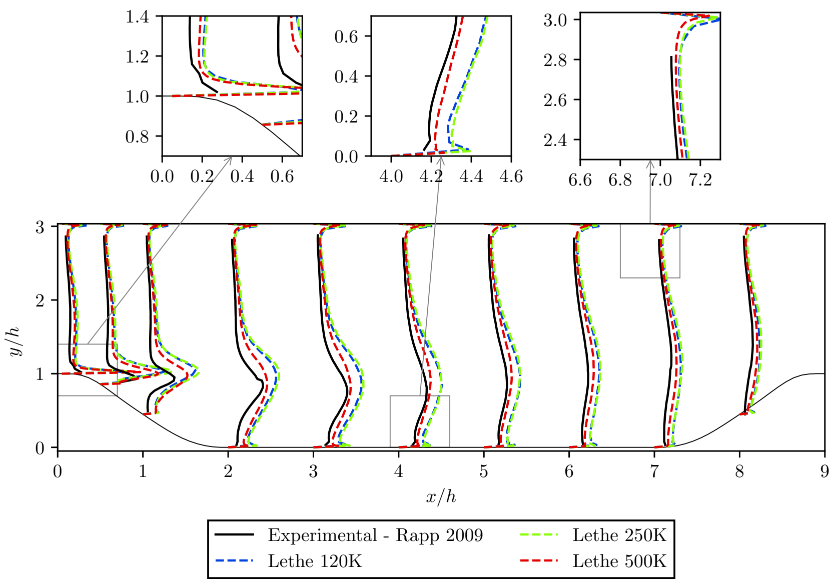

Figure 5.16 Reynolds normal stress in the x direction throughout the geometry at Re = 37 000, compared against the benchmarks. The profiles are scaled by a factor of 5 for ease of visualization.

4. For $x \in [0.7143h; 1.071h]$:
   $y(x) = 1.445h - 4.927h \times 10^{-2}x + 6.950h \times 10^{-4}x^2 - 7.394h \times 10^{-6}x^3$

5. For $x \in [1.071h; 1.429h]$:
   $y(x) = 0.6401h + 3.123h \times 10^{-2}x - 1.988h \times 10^{-3}x^2 + 2.242h \times 10^{-5}x^3$

6. For $x \in [1.429h; 1.929h]$:
   $y(x) = \max(0; 2.0139h - 7.180h \times 10^{-2}x + 5.875h \times 10^{-4}x^2 + 9.553h \times 10^{-7}x^3)$

5.8 Appendix B : Mesh Resolution

The quality of the meshes used in this study are evaluated in terms of wall-coordinates Δx^+, Δy^+ and Δx^+. The Δy^+ values along x/h are plotted in **Figures 5.17**, **5.18** and **5.19**. The average and maximum values are reported in for all the coordinates are reported in **Tables 5.6**, **5.7** and **5.8**.

5.9 Appendix C : Computational Time

The simulations were run using Niagara, a distributed memory cluster from the Research Alliance of Canada. Niagara consists of 2024 nodes, each with 40 Intel Skylake cores (2.4 Ghz)

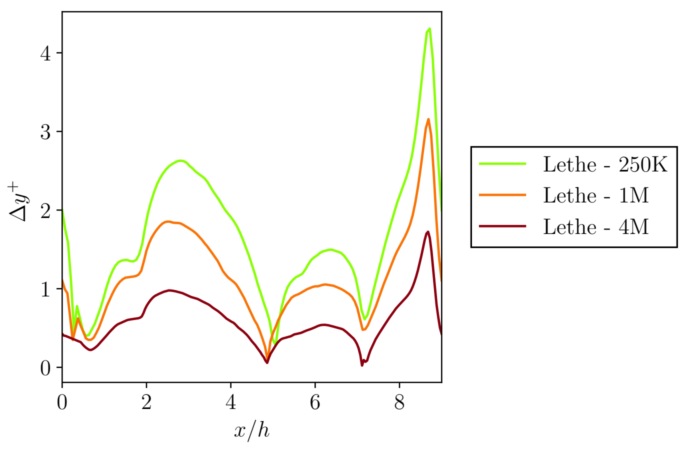

Figure 5.17 Distribution of Δy^+ along the lower wall for the different meshes used at Re = 5600. This was calculated for an average time of 800 s and a time step of 0.025.

Table 5.6 Maximum and average $\Delta x^+, \Delta y^+$ and Δz^+ for the different meshes at Re = 5600. This was calculated for an average time of 800 s and a time step of 0.025.

Mesh	Δx^+_{avg}	Δx^+_{max}	Δy^+_{avg}	Δy^+_{max}	Δz^+_{avg}	Δz^+_{max}
250K	2.96	8.85	1.66	4.30	4.20	12.52
1M	2.40	7.49	1.16	3.15	2.26	7.06
4M	1.39	4.67	0.61	1.73	1.51	5.06

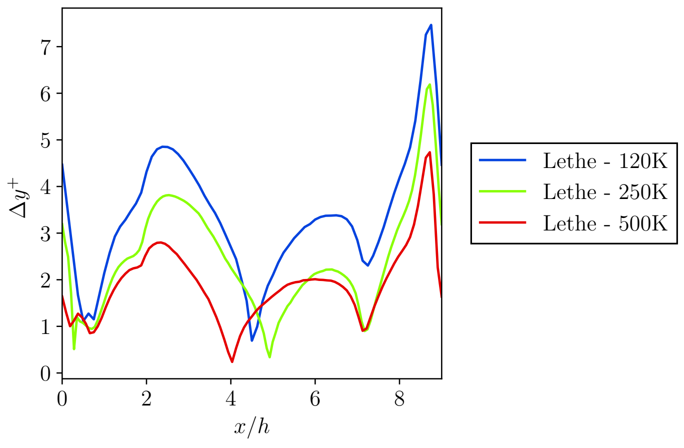

Figure 5.18 Distribution of Δy^+ along the lower wall for the different meshes used at Re = 10 600. This was calculated for an average time of 800 s and a time step of 0.025.

Table 5.7 Maximum and average $\Delta x^+, \Delta y^+$ and Δz^+ for the different meshes at Re = 10 600. This was calculated for an average time of 800 s and a time step of 0.025.

Mesh	Δx^+_{avg}	Δx^+_{max}	Δy^+_{avg}	Δy^+_{max}	Δz^+_{avg}	Δz^+_{max}
120K	6.83	17.69	3.34	7.46	6.41	16.60
250K	4.37	12.71	2.44	6.18	6.18	17.98
500K	4.84	13.95	1.89	4.73	4.15	11.97

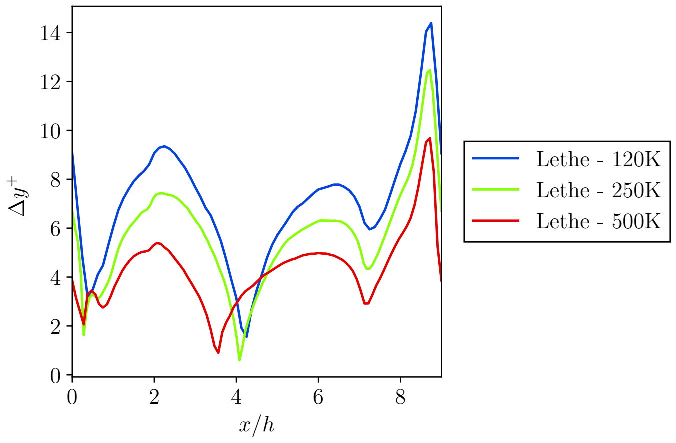

Figure 5.19 Distribution of Δy^+ along the lower wall for the different meshes used at Re = 37 000. This was calculated for an average time of 800 s and a time step of 0.025.

Table 5.8 Maximum and average $\Delta x^+, \Delta y^+$ and Δz^+ for the different meshes at Re = 37 000. This was calculated for an average time of 800 s and a time step of 0.025.

Mesh	Δx^+_{avg}	Δx^+_{max}	Δy^+_{avg}	Δy^+_{max}	Δz^+_{avg}	Δz^+_{max}
120K	14.46	34.09	7.06	14.38	13.57	31.98
250K	10.21	25.60	5.73	12.46	14.44	36.21
500K	11.11	28.51	4.35	9.67	9.54	24.48

with 202 GB of RAM per node. The computational times in core hours for all simulations considered in this study are presented in **Table 5.9**.

Table 5.9 Computational times of all simulations performed.

Mesh	**Time step** [s]	**Time average** [s]	**Time [core hour]**
Baseline, time step and time averaging simulations for Re = 5600			
Coarse			3647
Regular	0.1	1000	16 800
Fine			116 667
Coarse			5180
Regular	0.05	1000	24 667
Fine			174 000
Coarse			6933
Regular	0.025	1000	36 667
Fine			188 000
Coarse			9400
Regular	0.0125	1000	39 933
Fine			167 333
Simulations for Re = 10 600			
Very coarse			3061
Coarse	0.025	800	5493
Intermediate			15 307
Simulations for Re = 37 000			
Very coarse			3173
Coarse	0.025	800	5440
Intermediate			15 253

[a]**Note:** (cells, DoFs) of each mesh: very coarse (∼120K, 500K), coarse (∼250K, 1.1M), intermediate (∼500K, 2.4M), regular (∼1M, 4.5M) and fine (∼4M, 16M).

CHAPTER 6 ARTICLE 2 : A MATRIX-FREE STABILIZED SOLVER FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS

Laura Prieto Saavedra, Peter Munch, Bruno Blais.

Published in the Journal of Computational Physics, volume 538, on the 16th of June 2025. Available at `https://doi.org/10.1016/j.jcp.2025.114186`. *Part of the content was presented at the ECCOMAS Conference in Lisbon in June 2024 and the* ***deal.II*** *users and developers workshop in Fort Collins in August 2024.*

Declaration: The author of this thesis has contributed to the writing of the original draft, the visualization, the validation, the software, the methodology, the investigation, the formal analysis, the data curation and the conceptualization.

Abstract: We present an efficient solver for the incompressible Navier–Stokes equations implemented in a matrix-free fashion. It uses a higher-order continuous Galerkin finite element method for the space discretization and leverages a stabilized formulation that includes both the SUPG and PSPG terms. We solve the non-linear problem in a fully coupled way, using a Newton-Krylov method, which is preconditioned by a monolithic geometric multigrid solver. To evaluate its efficiency in terms of time to solution and scalability on modern high-performance computers, we use a manufactured solution, a steady flow around a sphere with Reynolds number Re = 150 and the Taylor–Green vortex benchmark at Re = 1 600. The results indicate that the solver is robust and scales for both steady-state and transient problems. We compare the matrix-free solver to a matrix-based version and show it exhibits lower memory requirements, better scalability, and significant speedups (10–100× for higher-order elements). Moreover, we demonstrate that a matrix-free implementation is highly efficient when using higher-order elements, which provide higher accuracy at a lower number of degrees of freedom for complex steady problems. To the best of our knowledge, this work is the first that uses a matrix-free monolithic geometric multigrid preconditioner to solve the stabilized Navier–Stokes equations. All implementations are available via the open-source software `Lethe`.

6.1 Introduction

The simulation of flows using computational fluid dynamics **(CFD)** has advanced greatly in recent decades and is now an essential tool for several industries ranging from aerospace design to process engineering. Despite this, simulation of flow problems (e.g., rotating flows, external flows) with a Reynolds number Re $>1\,000$ remains a challenge even with modern high-performance computers (e.g., in the aerospace industry [155]). One of the reasons for this is that most numerical algorithms were not originally designed to effectively utilize the hardware resources available today. For instance, when using the classical finite element method for such problems, a high resolution in terms of mesh, number of unknowns, and time steps is needed, which leads to large data structures (e.g., a sparse matrix) that need to be stored in the main memory and later accessed when performing computations. This leads to memory-bandwidth-bounded algorithms on distributed systems. The main idea of a matrix-free solver is to eliminate this bottleneck by modifying the algorithm and taking advantage of the abundance of floating-point operations in modern processors. In this work, we present such an algorithm for the incompressible Navier–Stokes equations.

The idea of a matrix-free solver is not new and can trace its roots back to the high-order spectral element method in the 1980s [88], where they realized that for higher-order polynomial expansions and localized basis functions, computational work takes place at the element level. This challenged the idea of generating sparse matrices and storing them as a global operator, and led to the development of operators on the element level. Later, this method gained popularity because of its geometric flexibility, high accuracy, and improved convergence properties. Specifically in the context of CFD, these methods became popular in that decade with the work of Paul Fischer and colleagues from the `NEK5000` project [89–93]. Subsequently, these methods started to become popular in the context of general-purpose finite element method **(FEM)** [156] and to be implemented in libraries such as `Nektar++` [157] and `deal.II` [81], where they aimed to use these strategies to solve more general problems. In recent years, they have become more popular due to new hardware capabilities, such as GPUs, where the matrix-free approach is particularly suitable; e.g., the implementations in the `MFEM` finite element library [83] or the `libCEED` library [80]. In addition to the method used to implement the operator, various solution strategies can be applied to solve the incompressible Navier–Stokes equations.

There are two main solution strategies for the discretized problem of the incompressible Navier–Stokes equations that can be found in the literature: i) splitting or segregated methods, which decouple the pressure and the velocity fields, and ii) fully coupled or monolithic methods for which the global system of equations is solved for both the velocity and the pressure at the same time; for general reviews on both strategies see [30, 60, 61]. The first strategy generally involves a Helmholtz-like equation for an (intermediate) velocity and the solution of a Poisson-like problem for the pressure to project the intermediate velocity onto a space of divergence-free velocity field (pressure projections; see, e.g., [63, 158, 159]). Due to the simplicity of these methods, involving the solution of rather simple systems of linear equations, they are known to be efficient and are as a result the main strategy used in several open source specialized CFD software, for example `Neko` [37], `NEK5000` [65], `Nektar++` [39], `NekRS` [38] and `ExaDG` [40]. However, challenges regarding boundary conditions, and stability and accuracy in time are known [30, 64]. The second strategy usually involves the solution of a system of nonlinear equations and leads to a corresponding Jacobian of saddle-point form. This can be more challenging from a nonlinear solver and a linear algebra perspective, but allows larger time steps, potentially amortizing additional costs compared to splitting methods. The development of efficient solvers for saddle-point problems is an active research field (see [60] for a comprehensive review). In this work, we consider the stabilized version of the incompressible Navier–Stokes equation, which no longer leads to a saddle-point problem, and decide to solve the problem monolithically.

In general, the efficient solution of the monolithic problem requires the solution of systems of linear equations using a preconditioned Krylov method (e.g., the generalized minimal residual method, **GMRES**). There are different preconditioner strategies that can be classified into two main groups: i) block preconditioners and ii) (monolithic) multilevel methods, which include domain decomposition and multigrid methods [60]. We use a monolithic geometric multigrid method **(GMG)** as a preconditioner for the GMRES solver. Monolithic multigrid methods have been widely considered for the solution of the Stokes equations, both as preconditioners and solvers. An example is the work by Kohl et al. [97], where a monolithic GMG solver was developed for the discretization of the Stokes equations, using pressure-stabilizing/Petrov-Galerkin stabilization, with a particular focus on performance for higher-order FEM and considering a matrix-free implementation. They showed good parallel scalability and computational efficiency up to 147K processes and systems with more than 3.6×10^{12} unknowns using both stabilized equal order and Taylor-Hood elements.

Voronin et al. [101] developed a low-order GMG preconditioner with Vanka and Braess-Sarazin as smoothers in a similar manner and obtained satisfactory results; this work was recently extended to high-order discretizations and *hp*-multigrid in [102]. The results showed better performance and reduced setup and solve times for higher orders and unstructured problems. The same group presented similar ideas but using a monolithic algebraic multigrid **(AMG)** in [71]. Another work that considers Taylor-Hood elements for the Stokes and generalized Stokes equations is that by Jodlbauer et al. [103]. They implemented a matrix-free monolithic GMG solver using Chebyshev-Jacobi smoothers and obtained satisfactory results. They indicated that the method can serve as a basis for developing a method with similar ideas for Navier–Stokes problems, but more work needs to be done to ensure robustness for convection-dominated flows.

Contrary to the research for the Stokes equations, in the case of the Navier–Stokes equations that consider stabilization techniques, the focus has been mainly on block preconditioners. A complete review on this type of preconditioners can be found in the book by Elman et al. [29]. Cyr et al. [68] compared three block preconditioners (Pressure-Convection Diffusion, **PCD**; Least Squares Commutators, **LSC**; and semi-implicit method for pressure linked equations consistent, **SIMPLEC**) with additive Schwarz domain decomposition methods and monolithic AMG methods with incomplete LU factorization **(ILU)** as smoother. They considered different steady-state and transient benchmarks, and in general, they obtained good scalability properties for monolithic AMG, LSC, and PCD, but not for the domain decomposition method and the SIMPLEC preconditioner. They also observed that the LSC preconditioner was not robust with respect to stabilization and that the monolithic AMG required specialized "heavy" smoothers.

Monolithic GMG preconditioners have also been studied in the context of magnetohydrodynamics equations. Ohm et al. [160] developed physics-specialized block smoothers and tested the algorithm with different applications using stabilization and demonstrated superiority over traditional AMG. Adler et al. [104] considered a formulation without stabilization and compared their multigrid implementation with a segregated method; they observed that the multigrid was more efficient and robust and was capable of solving steady-state and transient cases up to 170 million degrees of freedom. However, they stated that the main limitation of the work was the low order ($\mathbf{P}_2$ for the velocity, P_1 for the pressure and lowest-order Nédélec elements for the magnetic field) and the lack of stabilization of the discretization they used. To the best of our knowledge, our work presents the first study that aims to use a matrix-free GMG method to solve the incompressible Navier–Stokes equations with stabilization. In this paper, we show how such a solver can be used to simulate challenging steady-state and transient problems. In particular, we consider a flow around a sphere at Re = 150 and the turbulent Taylor–Green vortex problem at Re = 1600; in both of these problems the element Peclet number indicates that stabilization is required.

The article is organized as follows: in Section 6.2 we present the continuous Galerkin discretization along with the stabilization techniques. In Section 6.3 we present implementation details on the matrix-free implementation of the solver in the open source computational fluid dynamics software `Lethe`, which previously only supported matrix-based algorithms, along with the information on the geometric multigrid used as a preconditioner for the linear solver. In Section 6.4, we present the results of three numerical tests: a manufactured solution, a steady-state flow around a sphere, and the Taylor–Green vortex benchmark. These results allow us to evaluate the numerical accuracy and performance of the solver. Finally, in Section 6.5 we summarize our conclusions and point to future work on matrix-free methods for the incompressible Navier–Stokes equations.

6.2 Governing Equations and Discretization

We aim to solve the incompressible Navier–Stokes equations in a d-dimensional domain Ω with boundary $\partial\Omega$:

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \text{ on } \Omega \times [0,t], \\ \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} - \boldsymbol{f} &= 0 \text{ on } \Omega \times [0,t]. \end{aligned} \tag{6.1}$$

This is a coupled non-linear problem for the velocity $\boldsymbol{u}(\boldsymbol{x},t)$ and the pressure $P(\boldsymbol{x},t)$. The parameter ν is the kinematic viscosity, $p^* = P/\rho$ is the reduced pressure with density ρ and $\boldsymbol{f}$ is a source term often referred to as the body force vector. The well-posedness of the problem is guaranteed by providing appropriate boundary conditions in $\partial\Omega$, and initial conditions in the case of a transient problem. For steady-state problems, $\partial_t \boldsymbol{u} = 0$.

6.2.1 Variational Form and Spatial Discretization

We use the finite element method to discretize the Navier–Stokes equations (Eq. 6.1) in space. For this, the domain Ω is divided into k cells Ω_k with a mesh size h, e.g., hexahedral-shaped cells in 3D. We multiply by test functions, a scalar test function q for the continuity equation and a vector test function $\boldsymbol{v}$ for the momentum equation. Then, the pressure and diffusive terms are integrated by parts leading to the final variational formulation: find $(\boldsymbol{u}, p) \in \mathcal{U} \times \mathcal{P}$ such that for all $(\boldsymbol{v}, q) \in \mathcal{V} \times \mathcal{Q}$:

$$\begin{aligned} F(\boldsymbol{u}, p) := {} & (q, \nabla \cdot \boldsymbol{u})_\Omega + (\boldsymbol{v}, \partial_t \boldsymbol{u})_\Omega + (\boldsymbol{v}, (\boldsymbol{u} \cdot \nabla)\boldsymbol{u})_\Omega - (\boldsymbol{v}, \boldsymbol{f})_\Omega \\ & + \nu(\nabla \boldsymbol{v}, \nabla \boldsymbol{u})_\Omega - \langle \boldsymbol{v}\boldsymbol{n}, \nu \nabla \boldsymbol{u} \rangle_{\partial\Omega} \\ & - (\nabla \cdot \boldsymbol{v}, p^*)_\Omega + \langle \boldsymbol{n} \cdot \boldsymbol{v}, p^* \rangle_{\partial\Omega} = 0, \end{aligned} \tag{6.2}$$

where $\boldsymbol{n}$ is the normal outward-pointing vector on the boundary $\partial\Omega$ and $\boldsymbol{v}\boldsymbol{n}$ is a dyadic product. For simplicity, we use the notation $(a,b)_\Omega = \int_\Omega a \cdot b \; \mathrm{d}\Omega$. The boundary terms are discarded, since we assume zero stress conditions on $\partial\Omega$. The solution and test function spaces are defined as follows:

$$
\begin{aligned}
&\mathcal{U} := \{\boldsymbol{u}_h \in H^1(\Omega)^d, \boldsymbol{u}_h = \boldsymbol{u}_0 \text{ on } \partial\Omega\},\\
&\mathcal{V} := \{\boldsymbol{v}_h \in H^1(\Omega)^d, \boldsymbol{v}_h = 0 \text{ on } \partial\Omega\},\\
&\mathcal{P} := \{p^* \in L^2(\Omega)\},\\
&\mathcal{Q} := \{q \in L^2(\Omega)\},
\end{aligned}
$$

where $H^1(\Omega)^d$ is the Sobolev space containing functions with square-integrable first derivatives and $L^2(\Omega)^d$ is the space with square-integrable functions. The solution and test functions are discretized in space using a finite element space of continuous piecewise Lagrange polynomials $(\boldsymbol{\phi}, \psi)$ of degree p (Q_p):

$$
\boldsymbol{u}_h(\boldsymbol{x}, t) = \sum_j \boldsymbol{\phi}_j(\boldsymbol{x})\boldsymbol{U}_j(t),\ p_h^*(\boldsymbol{x}, t) = \sum_j \psi_j(\boldsymbol{x})\boldsymbol{P}_j(t), \tag{6.3a}
$$

$$
\boldsymbol{v}_h(\boldsymbol{x}, t) = \sum_i \boldsymbol{\phi}_i(\boldsymbol{x})\boldsymbol{V}_i(t),\ q_h(\boldsymbol{x}, t) = \sum_i \psi_i(\boldsymbol{x})\boldsymbol{Q}_i(t). \tag{6.3b}
$$

One of the main advantages of the finite element method is the possibility of using high-order elements, since we can simply increase the degree p of the polynomial defined on a cell without having to explicitly change the discretization. In this publication, we consider higher-order elements to be of degree $p \geq 2$ following the definition provided by [34]. To deal with the non-linearity due to the advection term, we use Newton's method with line search (considering a stepsize α), which consists of computing a sequence of Newton steps that successively apply the following two steps:

1. Solve: $\boldsymbol{F'}(\boldsymbol{U}^n, \boldsymbol{P}^n)[\delta\boldsymbol{U}^n, \delta\boldsymbol{P}^n]^\top = -F(\boldsymbol{U}^n, \boldsymbol{P}^n)$.
2. Compute update: $\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \alpha\delta\boldsymbol{U}^n$ and $\boldsymbol{P}^{n+1} = \boldsymbol{P}^n + \alpha\delta\boldsymbol{P}^n$.

The first step leads to the solution of a linearized system in each non-linear iteration, where the system matrix corresponds to the Jacobian of the weak form and the right-hand side corresponds to the negative of its residual form. In matrix-vector notation, the resulting linear system has the following form:

$$
\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{F'}} \begin{bmatrix} \delta\boldsymbol{U} \\ \delta\boldsymbol{P} \end{bmatrix} = -\underbrace{\begin{bmatrix} \boldsymbol{R}_u \\ \boldsymbol{R}_p \end{bmatrix}}_{\boldsymbol{R}}, \tag{6.4a}
$$

where the terms of the Jacobian are found by means of a Gateaux derivative. The block matrix $\boldsymbol{A}$ comprises the transient, convective, and diffusive terms, $\boldsymbol{B}$ includes the gradient term, and its transpose $\boldsymbol{B}^T$ the divergence term; for an element k, the block matrices are as follows for an entry (i,j):

$$\boldsymbol{A}^k_{i,j} = (\boldsymbol{\phi}_i, \partial_t \boldsymbol{\phi}_j + (\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi_j}\cdot\nabla)\boldsymbol{u})_{\Omega_k} + \nu(\nabla\boldsymbol{\phi}_i, \nabla\boldsymbol{\phi}_j)_{\Omega_k}, \tag{6.4b}$$

$$\boldsymbol{B}^k_{i,j} = (\nabla\cdot\boldsymbol{\phi}_i, \psi_j)_{\Omega_k}. \tag{6.4c}$$

This discretization in space has three limitations: i) it requires to fulfill the Ladyzhenskaya-Babuska-Brezzi **(LBB)** condition [161], which restricts the available elements or element combinations that can be used, ii) it is unstable when increasing the Reynolds number as it leads to oscillations in the solution, and iii) the linear system that needs to be solved is a saddle-point problem that is difficult to solve numerically using common iterative methods. We use stabilization techniques to overcome these limitations and solve the problem in a fully-coupled monolithic way.

6.2.2 Stabilization Techniques

We add two stabilization terms to the weak formulation F (Eq. 6.2): the Streamline-Upwind/Petrov-Galerkin **(SUPG)** and the Pressure-Stabilizing/ Petrov-Galerkin **(PSPG)** terms [149]:

$$\begin{aligned}\hat{F}(\boldsymbol{u},p) :&= F(\boldsymbol{u},p) + \underbrace{\sum_k(\tau\nabla q, \partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p^* - \nu\Delta\boldsymbol{u} - \boldsymbol{f})_{\Omega_k}}_{\text{PSPG term}} \\ &+ \underbrace{\sum_k(\tau(\boldsymbol{u}\cdot\nabla)\boldsymbol{v}, \partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p^* - \nu\Delta\boldsymbol{u} - \boldsymbol{f})_{\Omega_k}}_{\text{SUPG term}} = 0.\end{aligned} \tag{6.5}$$

Both terms consist of the strong residual of the momentum equation, multiplied by some test function terms and a stabilization parameter τ. For τ, we extended the definition of Tezduyar et al. [149] to high order by including the element order p:

$$\tau = \left[\left(\frac{1}{\Delta t}\right)^2 + \left(\frac{2\|\boldsymbol{u}\|p}{h_{\text{conv}}}\right)^2 + 9\left(\frac{4\nu p^2}{h^2_{\text{diff}}}\right)^2\right]^{-1/2}, \tag{6.6}$$

where $h_{\text{conv}} = h_{\text{diff}} = h$ is the element size, which in 3D is equal to $h = (6h_k/\pi)^{1/3}$, where h_k is the volume of a cell. These types of stabilization techniques have been proven to be consistent formulations that result in solutions with minimal loss of accuracy [42, 149]. Recently, the periodic hills turbulent benchmark was simulated using this formulation and accurate results were obtained for average velocities and Reynolds stresses [162]. PSPG allow us to use equal-order elements for the pressure and the velocity, and SUPG reduces oscillations in the velocity field when simulating flows with high Reynolds numbers. The stabilization terms modify the linear system presented in Eq. 6.4 as follows:

$$\underbrace{\begin{bmatrix} \widehat{\boldsymbol{A}} & \widehat{\boldsymbol{B}} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}}_{\widehat{\boldsymbol{F}'}} \begin{bmatrix} \delta \boldsymbol{U} \\ \delta \boldsymbol{P} \end{bmatrix} = - \underbrace{\begin{bmatrix} \widehat{\boldsymbol{R}}_u \\ \widehat{\boldsymbol{R}}_p \end{bmatrix}}_{\widehat{\boldsymbol{R}}}, \tag{6.7a}$$

where each block, evaluated at the linearization point $(\hat{\boldsymbol{u}}, \hat{p}^*)$, is now given by:

$$\begin{aligned} \widehat{\boldsymbol{A}}^k_{i,j} &= \boldsymbol{A}^k_{i,j} + (\tau(\hat{\boldsymbol{u}} \cdot \nabla)\boldsymbol{\phi}_i, \partial_t \boldsymbol{\phi}_j + (\hat{\boldsymbol{u}} \cdot \nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j \cdot \nabla)\hat{\boldsymbol{u}} - \nu \Delta \boldsymbol{\phi}_j)_{\Omega_k} \\ &\qquad + (\tau(\boldsymbol{\phi}_j \cdot \nabla)\boldsymbol{\phi}_i, \partial_t \hat{\boldsymbol{u}} + (\hat{\boldsymbol{u}} \cdot \nabla)\boldsymbol{u} + \nabla \hat{p}^* - \nu \Delta \hat{\boldsymbol{u}} - \boldsymbol{f})_{\Omega_k}, &(6.7\text{b}) \\ \widehat{\boldsymbol{B}}^k_{i,j} &= \boldsymbol{B}^k_{i,j} + (\tau(\hat{\boldsymbol{u}} \cdot \nabla)\boldsymbol{\phi}_i, \nabla \psi_j)_{\Omega_k}, &(6.7\text{c}) \\ \boldsymbol{C}^k_{i,j} &= \boldsymbol{B}^k_{j,i} + (\tau \nabla \psi_i, \partial_t \boldsymbol{\phi}_j + (\hat{\boldsymbol{u}} \cdot \nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j \cdot \nabla)\hat{\boldsymbol{u}} - \nu \Delta \boldsymbol{\phi}_j)_{\Omega_k}, &(6.7\text{d}) \\ \boldsymbol{D}^k_{i,j} &= (\tau \nabla \psi_i, \nabla \psi_j). &(6.7\text{e}) \end{aligned}$$

As can be seen, several terms are added to each of the blocks. Most importantly, the $\boldsymbol{D}$ block replaces the zero block encountered without stabilization. We are also interested in solving steady-state simulation cases; in this case, all terms that include ∂_t are not present, and the first term of the stabilization parameter defined in Eq. 6.6 vanishes; however, the general block structure (Eq. 6.7) is preserved. Although Galerkin/Least Squares **(GLS)** stabilization is not considered in this study, this type of stabilization could be used instead and would lead to a linear system with the same structure; similar results and conclusions in terms of performance are expected. We note that for Q_1Q_1 and Cartesian meshes the term $\Delta \boldsymbol{u}$ present in the strong residual evaluates to zero. However, for deformed meshes, due to mixed derivatives and higher-order elements this is not the case. Nevertheless, we recover the right order of accuracy. We defer to future work the investigation of other approaches to consistently account for this term.

6.2.3 Time Discretization

For transient problems, we use a second-order implicit backward-differentiation time-stepping scheme. For a constant time-step, the time derivative of the solution takes the following form:

$$\partial_t \boldsymbol{u}^{n+1} \approx \frac{1}{\Delta t}\left(\frac{3}{2}\boldsymbol{u}^{n+1} - 2\boldsymbol{u}^{n} + \frac{1}{2}\boldsymbol{u}^{n-1}\right), \tag{6.8a}$$

and the one of the solution increment in Newton's method the following form:

$$\partial_t \delta\boldsymbol{u}^{n+1} \approx \frac{1}{\Delta t}\left(\frac{3}{2}\delta\boldsymbol{u}^{n+1}\right). \tag{6.8b}$$

It is an implicit multistep scheme that needs solution vectors from the last two timesteps ($\boldsymbol{u}^n$, $\boldsymbol{u}^{n-1}$) and is A-stable when constant time step is used [76]. However, the implementation used in Lethe allows for variable time-step following the procedure described by [76].

6.3 Numerical Implementation

The solver is implemented in `Lethe` [105], an open source computational fluid dynamics software.[1] This software is written using the `deal.II` finite element library [109].[2] It uses `p4est` for mesh support [107], MPI for parallelization, and `Trilinos` for matrix-based linear algebra [108]. Until now, `Lethe` has supported only matrix-based algorithms to solve stabilized incompressible Navier–Stokes equations using a continuous Galerkin finite element approach [105]. This solver has been used to simulate a variety of physical phenomena such as non-Newtonian flow [163], particle-laden flow [164] and flow in agitated vessels [165]. In this publication, we extend the support of `Lethe` to matrix-free algorithms. The reason for this is based on the observation that, for our applications, high-order elements with $p \geq 2$ can greatly improve accuracy with a reduced number of degrees of freedom **(DoFs)** compared to linear elements; see [166, 167]. However, the use of these orders leads to an increase in the cost of assembly and application of the Jacobian, which for practical applications requires a considerable amount of computational resources. Having a matrix-free solver allows us not only to use higher-order elements for our applications, but also to take full advantage of the high-performance computing architectures available today.

6.3.1 Matrix Free Approach

The system of linear equations (Eq. 6.7) is solved by performing matrix vector multiplications repeatedly within an iterative method. The idea of matrix-free multiplication relies on avoiding the storage of the matrix by defining an operator that computes the effect of the

[1]`https://github.com/chaos-polymtl/lethe`
[2]`https://github.com/dealii/dealii`

matrix-vector product onto a vector by computing matrix entries. In this work, this operator is implemented using the cell-wise quadrature approach, which requires the redefinition of the matrix from the global level to the element level. The operator $\widehat{\mathcal{F}}'$ allows us to rewrite the product of the Jacobian matrix and a vector ($\boldsymbol{v} = \widehat{\boldsymbol{F}'}\boldsymbol{u}$) as a loop over cells:

$$\boldsymbol{v} = \widehat{\mathcal{F}}'(\boldsymbol{u}) = \sum_k \boldsymbol{R}_k^\top \widehat{\boldsymbol{F}}_k' \boldsymbol{R}_k \boldsymbol{u}.$$

This expression can be read from right to left: the cell-local values of cell k are gathered from $\boldsymbol{u}$ through $\boldsymbol{u}_k = \boldsymbol{R}_k\boldsymbol{u}$, then the local element matrix $\widehat{\boldsymbol{F}}_k'$ is applied to obtain $\boldsymbol{v}_k$ and, in the end, the contribution is scattered to the global degrees of freedom obtaining vector $\boldsymbol{v} = \boldsymbol{R}_k^\top \boldsymbol{v}_k$. This product is implemented by using what is called the cell-quadrature approach, which decomposes the Jacobian matrix into the product of three matrices when performing its multiplication with a vector $\boldsymbol{u}_k$ as follows:

$$\widehat{\boldsymbol{F}'}_k \boldsymbol{u}_k = \boldsymbol{\mathcal{S}}_k^\top \boldsymbol{\mathcal{Q}}_k \boldsymbol{\mathcal{S}}_k \boldsymbol{u}_k.$$

It is important to note that the operator computes all the blocks of the matrix $\boldsymbol{F}_k'$ on a cell and quadrature basis, which means that it includes the computation of the contributions of all the blocks of the matrix, i.e. $\widehat{\boldsymbol{A}}_k$, $\widehat{\boldsymbol{B}}_k$, $\boldsymbol{C}_k$ and $\boldsymbol{D}_k$, to the cell. To better understand the definition of $\boldsymbol{\mathcal{S}}_k$ and $\boldsymbol{\mathcal{Q}}_k$, we look at only one component of one of the block matrices in the operator: the diffusive term ($\nu(\nabla \boldsymbol{v}, \nabla \boldsymbol{u})$; Eq. 6.4); we discretize it to obtain the following form:

$$\begin{aligned}\nu(\nabla \boldsymbol{v}_h, \nabla \boldsymbol{u}_h)_\Omega &= \nu \sum_k (\nabla \hat{\boldsymbol{v}}_h, \nabla \hat{\boldsymbol{u}}_h)_{\Omega_k} \\ &= \nu \sum_k \left(\hat{\boldsymbol{V}}_i^k\right)^\top \int_{\Omega_k} \nabla \hat{\boldsymbol{\phi}}_i^\top \nabla \hat{\boldsymbol{\phi}}_j \hat{\boldsymbol{U}}_j^k \mathrm{d}\boldsymbol{x},\end{aligned}$$

where ∇ represents the gradient in the physical space and $\hat{\square}$ are vectors at the element level. We now use the mapping to go from the physical space $\boldsymbol{x}$ to the reference space $\boldsymbol{\xi}$ and use numerical quadrature to approximate the integral:

$$\begin{aligned}\int_{\Omega_k} \nabla \hat{\boldsymbol{\phi}}_i^\top \nabla \hat{\boldsymbol{\phi}}_j \hat{\boldsymbol{U}}_j^k \mathrm{d}\boldsymbol{x} &= \int_{\Omega_k} \left(\boldsymbol{J}_k^{-\top} \nabla_{\boldsymbol{\xi}} \hat{\boldsymbol{\phi}}_i\right)^\top \left(\boldsymbol{J}_k^{-\top} \nabla_{\boldsymbol{\xi}} \hat{\boldsymbol{\phi}}_j\right) \hat{\boldsymbol{U}}_j^k \left(\det \boldsymbol{J}_k\right) \mathrm{d}\boldsymbol{\xi} \\ &= \sum_q \underbrace{(\nabla_{\boldsymbol{\xi}_q} \hat{\boldsymbol{\phi}}_i)^\top}_{\boldsymbol{\mathcal{S}}_k^\top} \underbrace{\boldsymbol{J}_k^{-1} \omega_q (\det \boldsymbol{J}_k) \boldsymbol{J}_k^{-\top}}_{\boldsymbol{\mathcal{Q}}_k} \underbrace{(\nabla_{\boldsymbol{\xi}_q} \hat{\boldsymbol{\phi}}_j)}_{\boldsymbol{\mathcal{S}}_k} \hat{\boldsymbol{U}}_j^k.\end{aligned}$$

This equation can again be read as a sequence of three vector-matrix multiplications going from right to left as: i) transform the vector values on the local degrees of freedom to a vector of gradients in quadrature points ($\boldsymbol{\mathcal{S}}_k$), ii) multiply the gradients by the integration weights and Jacobian information ($\boldsymbol{\mathcal{Q}}_k$), and iii) apply the transposed of the gradient to have a vector of Laplacian values on the cell degrees of freedom ($\boldsymbol{\mathcal{S}}_k^T$). This decomposition allows to reuse the output of $\boldsymbol{\mathcal{S}}_k$ for all other terms of $\widehat{\boldsymbol{A}}_k$, $\widehat{\boldsymbol{B}}_k$, $\boldsymbol{C}_k$ and $\boldsymbol{D}_k$. The same procedure applies also to the residual evaluation. The implementation in `Lethe` uses the matrix-free algorithms implemented in `deal.II` [81, 109], which also achieve additional optimizations by using sum factorization techniques for tensor-product elements and vectorization. The former takes advantage of tensor-product elements for which ansatz and test functions can be decomposed as a tensor product of one-dimensional functions. The latter performs matrix-free multiplication by $\widehat{\boldsymbol{F}}_k'$ for several cell batches if the same SIMD instruction is used. One of the challenges that arises due to the usage of a matrix-free algorithm, apart from implementation details, is the need for an efficient, potentially matrix-free, linear solver.

6.3.2 Linear Solver Preconditioner: Geometric Multigrid

The choice of a preconditioner for the linear solver is essential, since the overall performance of the solver and its parallel scalability greatly depend on the efficiency of the solution of the linear system. We use GMRES as solver and precondition it using GMG [112, 113, 168]. This algorithm constructs a hierarchy of discretizations based on the geometrical refinement of the grid and considers all cells in their most refined state for each of the levels. The main motivation for the use of this preconditioner is that there is no need to build the whole system matrix (at any step of the algorithm), which is the case in classical preconditioners such as ILU [69] or AMG [70].

It is important to note that we always compute the exact Jacobian of the problem. However, there are other techniques available in the literature, such as the Jacobian-free Newton-Krylov method **(JFNK)**, that could also be used (for a comprehensive review see [169], for a recent publication that uses matrix-free FEM and JFNK see [170]).

There are four main components of a geometric multigrid preconditioner: i) a coarse-grid solver, ii) matrix-free level operators, iii) smoothers for each level, and iv) intergrid operators (prolongation and restriction). In `Lethe`, we incorporated the implementation of the GMG of `deal.II` [112, 113]. In **Algorithm 2**, a pseudocode of the multigrid v-cycle and its components is presented. The matrix-free level operators are built using the procedure presented in Section 6.3.1. The other components of the preconditioner are described in the following subsections.

Algorithm 2 Pseudocode for one v-cycle of the GMG preconditioner with l levels used to solve $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$. It uses level operators $\mathcal{A}^{(l)}$, intergrid operators (restriction $R_l^{(l-1)}$ and prolongation $P^l_{(l-1)}$), smoothers and a coarse-grid solver.

```
function GMG($l$, $\mathcal{A}^{(l)}, \boldsymbol{x}^{(l)}$, $\boldsymbol{b}^{(l)}$)
    if $l = 0$ then
        $\boldsymbol{x}^{(0)} \leftarrow$ CoarseGridSolver($\mathcal{A}^{(0)}, \boldsymbol{x}^{(0)}, \boldsymbol{b}^{(0)}$)        // Solve coarse grid
    else
        $\boldsymbol{x}^{(l)} \leftarrow$ Smoother($\mathcal{A}^{(l)}, \boldsymbol{x}^{(l)}, \boldsymbol{b}^{(l)}$)        // Presmooth
        $\boldsymbol{r}^{(l)} \leftarrow \boldsymbol{b}^{(l)} - \mathcal{A}^{(l)}\ \boldsymbol{x}^{(l)}$        // Compute residual
        $\boldsymbol{b}^{(l-1)} \leftarrow R_l^{(l-1)}\ \boldsymbol{r}^{(l)}$        // Restrict
        $\boldsymbol{x}^{(l-1)} \leftarrow$ GMG ($l-1$, $\mathcal{A}^{(l-1)}$, 0, $\boldsymbol{b}^{(l-1)}$)
        $\boldsymbol{x}^{(l)} \leftarrow \boldsymbol{x}^{(l)} + P^l_{(l-1)}\ \boldsymbol{x}^{(l-1)}$        // Prolongate
        $\boldsymbol{x}^{(l)} \leftarrow$ Smoother($\mathcal{A}^{(l)}, \boldsymbol{x}^{(l)}, \boldsymbol{b}^{(l)}$)        // Postsmooth
    return $\boldsymbol{x}^{(l)}$
```

Coarse-Grid Solver

Three options are considered for the coarse grid solver: i) a direct solver as implemented in the Amesos2 package in Trilinos [118], ii) a GMRES iterative method preconditioned by a p-multigrid (**PMG**; [171]) algorithm with a direct solver as its coarse-grid solver and iii) a single v-cycle of PMG solver with direct solver as its coarse-grid solver. The latter is a multigrid algorithm where the hierarchy of discretizations is obtained by reducing the order of the polynomial degree by one in order to obtain the next multigrid level; as such, all levels have the same mesh but different polynomial orders. This PMG algorithm was also incorporated in `Lethe` by taking advantage of the flexible multigrid implementation in `deal.II` [112].

Smoothers

For the pre- and post-smoothing steps, we consider a relaxation scheme for each level: $\boldsymbol{x}_{n+1}^{(l)} = \boldsymbol{x}_n^{(l)} + \omega \boldsymbol{P}^{-1}\boldsymbol{r}_n^{(l)}$, where $\boldsymbol{r}_n^{(l)}$ is the residual, defined as $\boldsymbol{r}_n^{(l)} = \boldsymbol{b}^{(l)} - \mathcal{A}^{(l)}\boldsymbol{x}_n^{(l)}$ with the level operator $\mathcal{A}^{(l)}$; and $\boldsymbol{P}$ is a preconditioner. The smoother is a critical component of the multigrid algorithm, as it often controls convergence by determining number of iterations or multigrid cycles needed [112,115]. In this work, two preconditioners for the smoother are considered: i) a diagonal matrix corresponding to the diagonal of each matrix-free level operator , which we call "inverse diagonal"(abbreviated in the figures as **ID**), and ii) an Additive Schwarz Method (**ASM**). The former has a very low setup and application cost and has been proven to be robust for Poisson and Stokes problems [112]. The latter is more expensive, as it consists of approximate solves on sub-blocks of the level operator with all degrees of freedom of a cell, but might lead to fewer smoother iterations. In this work, we use a naive implementation of ASM whose setup consists of two steps. In the first step, we assemble the sparse system matrices and, in the second step, we extract from it the blocks and perform LU decomposition on these. Since we are computing the sparse system matrix as intermediate quantity, the memory consumption of our ASM implementation is high and comparable to that of the matrix-based code. This implementation could be further improved by using ideas presented for example in [115, 172–174]. Also, the type of patches could be varied: due to good results, we use cell-centric patches, while vertex-star-based patches are more common in the context of the solution of non-stabilized Stokes problems due to their optimally at higher polynomial degrees [101, 102, 174, 175]. We defer their investigation to future work.

The relaxation parameter ω is computed via $\omega = 2/(\lambda_{\min}(\boldsymbol{P}^{-1}\mathcal{A}) + \lambda_{\max}(\boldsymbol{P}^{-1}\mathcal{A}))$, which is optimal for symmetric positive definite matrices [114]. For the estimation of the maximum eigenvalue $\lambda_{\max}$ we use 20 iterations of the power iteration method. Then, the minimum eigenvalue is set to $\lambda_{\min} = \lambda_{\max}/\psi$, where values of ψ between 2 and 20 are common in literature [115]. For more details on the eigenvalue estimation algorithm the reader is referred to [116]. The estimation of the eigenvalues for each level operator is crucial since the stabilization terms, introduced in Eq. 6.5, depend on the size of the element and it changes for each level.

Intergrid Operators

The transfer operators are also implemented in a matrix-free fashion in `deal.II` as detailed in the publication by Munch et al. [112]; apart from the transfer of the defect $\boldsymbol{x}^{(l)}$ and the residuals $\boldsymbol{r}^{(l)}$, they are also required to interpolate the solutions of previous transient iterations, as required for the time-stepping scheme in the case of transient simulations ($\boldsymbol{u}^n$ and $\boldsymbol{u}^{n-1}$ in Eq. 6.8), and the linearization point ($\hat{\boldsymbol{u}}$ and $\hat{p}^*$ in Eq. 6.7) to all the multigrid levels.

6.4 Numerical Tests

In this section, we consider three cases to evaluate the performance of the solver. The first two cases are steady-state cases: the first one considers internal flow in a simple geometry, while the second one considers external flow in a complex geometry with a high Reynolds number. The third case is a transient turbulent benchmark, the Taylor–Green vortex, commonly used to test higher-order methods. These tests allow us to demonstrate the computational performance and accuracy of the solver. We also compare the results with a matrix-based implementation, which assembles the system matrix in a classical fashion, considers AMG or ILU as preconditioners for the linear solver, and is also available in `Lethe`. In this work, we only consider 3D meshes with hexahedral-shaped cells; however, the implementation can also be used in 2D and could be used with simplices. All tests were run on the Niagara distributed memory cluster of the Digital Research Alliance of Canada. Niagara consists of 2 024 nodes, each with 40 Intel Skylake cores (2.4 GHz) with 202 GB of RAM per node. We use AVX512 instructions for vectorization over 8 cells/doubles (512 bits). A summary of the solver parameters used for each case is presented in **Table 6.1**. We point out that we use 5 or 2 smoothing sweeps in the case of inverse diagonal or ASM. For details on how these values were chosen, the reader is referred to Appendix A in Section 6.7.

6.4.1 Manufactured Solution

The Method of Manufactured Solutions (**MMS**) allows us to verify the implementation of the solver by manufacturing an analytical problem with an exact solution [176]. It consists on defining $\boldsymbol{u}$ and p^* as follows (see **Figure 6.1** for the velocity magnitude on the domain):

$$\begin{pmatrix} u_x \\ u_y \\ u_z \\ p^* \end{pmatrix} = \begin{pmatrix} \sin^2(\pi x)\cos(\pi y)\sin(\pi y)\cos(\pi z)\sin(\pi z) \\ \cos(\pi x)\sin(\pi x)\sin^2(\pi y)\cos(\pi z)\sin(\pi z) \\ -2\cos(\pi x)\sin(\pi x)\cos(\pi y)\sin(\pi y)\sin^2(\pi z) \\ \sin(\pi x)\sin(\pi y)\sin(\pi z) \end{pmatrix},$$

and inserting them into the Navier–Stokes equations to find the appropriate source term $\boldsymbol{f}$. This source term is considered to simulate the steady-state problem, using Q_pQ_p elements with $p = 1, 2, 3$ on a domain $\Omega = (-1, 1)^3$ with zero Dirichlet boundary conditions everywhere. The domain is subdivided in 2^ℓ segments in each direction. This problem considers an element Peclet number $\mathrm{Pe_h} = h$ which means that there is no effect of the stabilization terms aside from ensuring that the Q_pQ_p space is stable. All refinement cases considered are presented in **Table 6.2**. The non-linear solver starts with a zero initial guess.

A convergence analysis was carried out and the results are presented in **Figure 6.2**. The results demonstrate that both the velocity and the pressure converge spatially to the correct order for all finite element discretizations used, that is, $O(\Delta h^{p+1})$ for the velocity and $O(\Delta h^p)$ for the pressure, with exception of the pressure order of convergence obtained using Q_1Q_1 elements. This result has already been obtained in other works that consider the stabilized formulation for the incompressible Navier–Stokes [105].

This MMS problem considers a low element Peclet number ($\mathrm{Pe_h}$) and a structured mesh,

Table 6.1 Summary of the solver parameters used for each of the numerical tests. It includes the parameters of the Newton non-linear solver and the GMRES linear solver preconditioned by geometric multigrid (GMG), algebraic multigrid (AMG) or incomplete LU (ILU). GMG considers two different smoothers and three coarse-grid solvers depending on the problem. The AMG and ILU preconditioners are only used for matrix-based reference computations and use the implemetations from Trilinos ML [119] and Trilinos Ifpack [120], respectively. For more detailed parameters for the AMG and ILU preconditioners refer to Appendix B in Section 6.8.

Component	**Parameter**	**MMS**	**Sphere**	**TGV**
Newton solver	Absolute tolerance	10^{-8}	10^{-5}	10^{-5}
GMRES linear solver	Relative residual	10^{-4}	10^{-4}	10^{-4}
	Absolute residual	10^{-10}	10^{-7}	10^{-7}
GMG preconditioner	**Smoother**			
	Inverse diagonal (ID)			
	Number of iterations	5	5	5
	Additive Schwarz method (ASM)			
	Number of iterations	2	2	–
	Coarse-grid solver			
	Direct solver	✓	–	✓
	ID: GMRES preconditioned by PMG with direct solver			
	Relative residual	–	10^{-2}	–
	ASM: PMG with direct solver	–	✓	–
AMG preconditioner	**Smoother** *Incomplete LU*			
	Fill	0	0	–
	Absolute tolerance	10^{-12}	10^{-9}	–
	Relative tolerance	1	1	–
	Number of iterations	2	2	–
ILU preconditioner	Fill	–	–	0
	Absolute tolerance	–	–	10^{-10}
	Relative tolerance	–	–	1

however, in practical problems, higher Peclet numbers and unstructured meshes are often encountered. Due to this, we also verify the solver using a Taylor vortex problem in 2D for different unstructured meshes considering a Reynolds number $\mathrm{Re} = 10\,000$ ($\mathrm{Pe_h} \in [491, 1963]$) and an inviscid case with $\mathrm{Re} \to \infty$ (as proposed by Ham and Iaccarino [177]). Since no

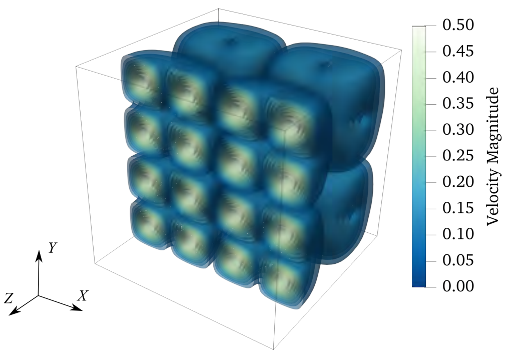

Figure 6.1 Isocontours of the velocity magnitude in the domain with a closer view to a xy-plane considering $\ell = 6$ and Q_2Q_2 elements for the manufactured solution problem.

scalability of the solver is studied for the Taylor vortex case, the results are presented in Appendix D in Section 6.10. The results show that the solver recovers the right order of convergence for the velocity, and that this order is preserved even for inviscid flows with highly unstructured meshes.

The MMS problem also allows us to study the scalability of the solver. We conducted a strong scaling study using up to 2 560 computing cores (64 nodes); its results are presented in **Figure 6.3**. The figure shows the results for the matrix-free solver using the two proposed smoothers for the GMG preconditioner (inverse diagonal and ASM) and the matrix-based reference solver using AMG. The inverse-diagonal smoother was not robust for the Q_1Q_1 cases, which is the reason why only ASM results are shown. What particularly stands out is that the matrix-free solver requires less time to solution than the matrix-based solver for all degree elements regardless of the smoother used for the GMG preconditioner. In the case of the ASM smoother, the matrix-free approach is 1.3-1.9$\times$ faster than the matrix-based approach for Q_1Q_1. The difference increases in the case of higher order elements where the matrix-free solver is 3.9-8.6$\times$ faster for Q_2Q_2 and 4.5-14$\times$ faster for Q_3Q_3. The inverse-diagonal smoother only converges for the higher-order cases, where it is 13.1-35.9$\times$ faster for Q_2Q_2 and 24.4-96.2$\times$ faster for Q_3Q_3. The speedup increases with increasing problem size and is related to the poor parallel-scalability behavior of the matrix-based solver. The results show that the matrix-free solver scales better than the matrix-based solver for all cases. In fact, in the literature, several results indicate that AMG preconditioners do not scale optimally and its scalability deteriorates as the number of nodes and element order p increase [178,179]. In addition, the results also give an indication of the memory requirements of both solvers, since several simulations using $1, 2$, and 4 nodes run out of memory when using the matrix-based solver or the matrix-free solver with the ASM smoother, and therefore cannot be shown in the plot. As expected, the inverse-diagonal smoother requires less memory

Table 6.2 Summary of the refinement level, number of cells and number of degrees of freedom (DoFs) for all the simulation cases considered for the manufactured solution problem. To generate the mesh, a coarse mesh consisting of a single cell is created and then refined ℓ times. The number of cells increases as $2^{3\ell}$.

Refinement ℓ	**No. Cells**	**No. DoFs**		
	$(1 \times 2^{3\ell})$	$\boldsymbol{Q_1Q_1}$	$\boldsymbol{Q_2Q_2}$	$\boldsymbol{Q_3Q_3}$
4	4K	20K	144K	470K
5	33K	144K	1M	3.6M
6	262K	1M	8M	28M
7	2M	8M	68M	228M
8	17M	68M	540M	1B
9	134M	540M		

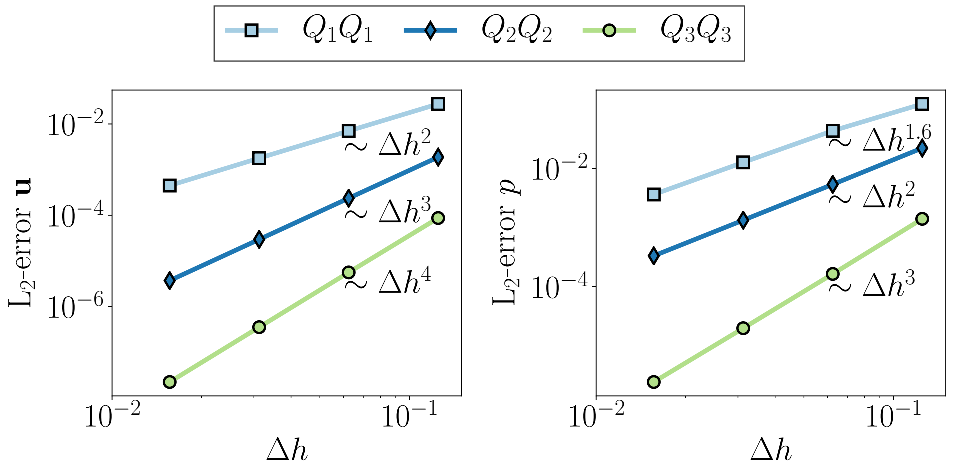

Figure 6.2 Order of convergence for velocity-pressure discretizations with $p = 1, 2$, and 3 and cases with refinement $4 \leq \ell \leq 7$ for the manufactured solution problem.

than the ASM smoother, which is more robust but has similar memory requirements to the matrix-based solver.

While in the previous results the wallclock time of the total simulation is presented, **Figure 6.4** reports the time spent for the Q_2Q_2 cases broken down into the setup of the preconditioner[3], the solution of the linear system, and other components of the simulation, such

[3]For GMG, the setup cost includes the creation of the multigrid levels, the setup of the coarse-grid solver, the setup and initialization of the smoother for each level, the setup of the matrix-free level operators, and the transfer of relevant data between levels. For AMG, the cost is given by the Trilinos implementation with the same components but with level matrices computed through the Galerkin operator instead of rediscretization as in the matrix-free context.

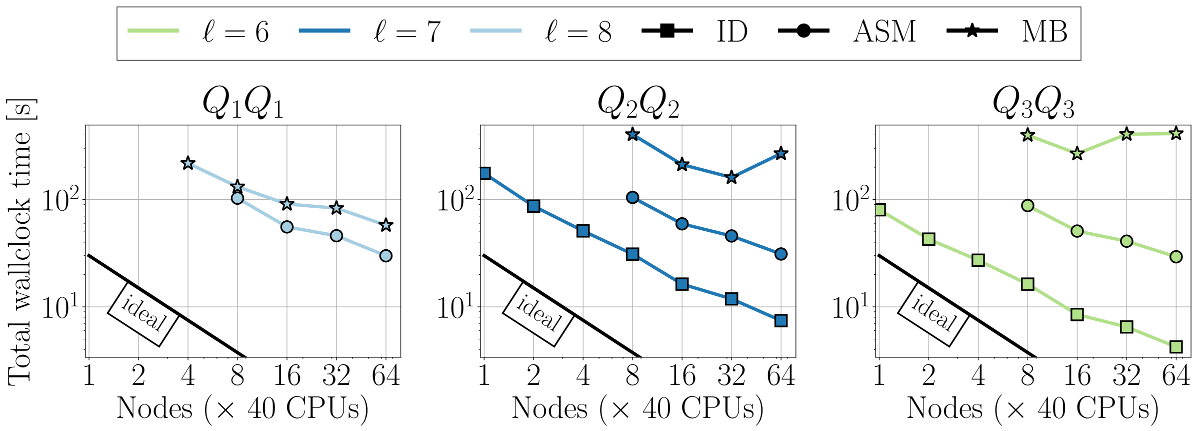

Figure 6.3 Strong scaling study considering $\ell = 8$ for Q_1Q_1, $\ell = 7$ for Q_2Q_2 and $\ell = 6$ for Q_3Q_3 for the manufactured solution problem, comparing inverse diagonal (ID) and ASM as smoother of GMG, as well as AMG for the matrix-based (MB) solver.

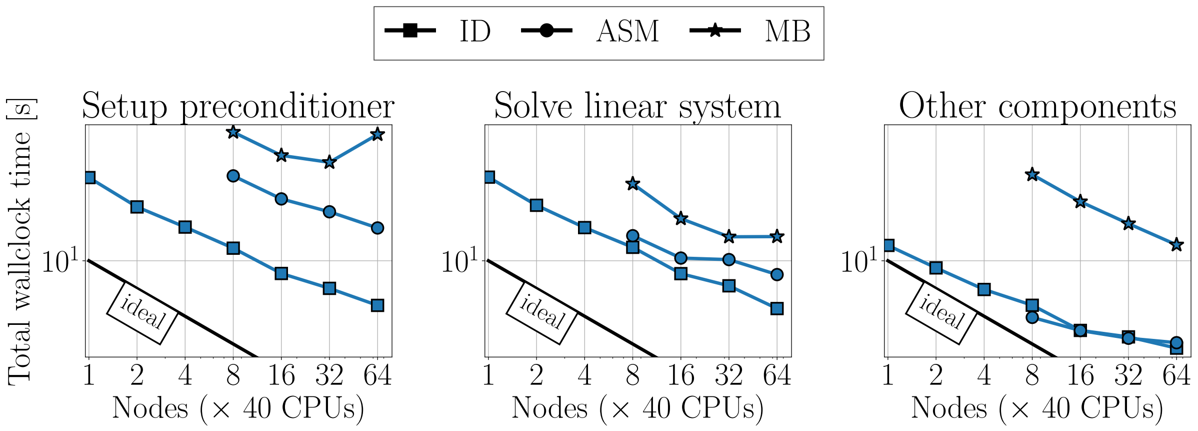

Figure 6.4 Detailed timings for the setup of the preconditioner, time to solution of the linearized problem and other components (assembly of the Jacobian, assembly of the residual, mesh generation and postprocessing) for the strong scaling study for the manufactured solution problem using Q_2Q_2 and $\ell = 7$ (see Figure 6.3 for total times).

as mesh creation, Jacobian assembly, residual assembly, and postprocessing. We observe that the setup of the preconditioner dominates the total cost, in terms of time, of the matrix-based solver and also determines its scalability. In the case of the matrix-free solver, both the setup and the solution of the linear system have a very similar cost for the inverse-diagonal smoother, while the cost of setting up the preconditioner is larger in the case of the ASM smoother. Furthermore, the results show that the matrix-based solver spends more time than the matrix-free solver in the other parts of the code: as an example, in the simulation performed using 8 nodes, the matrix-free solver only spends 5.1% of the total time in these other parts, while this time represents 19.4% in the case of the matrix-based simulation. The reason for this is that the latter spends a considerable amount of time assembling the Jacobian and the residual. In fact, 13.7% corresponds to the Jacobian assembly and 5.1% to the assembly of the residual vector. This shows the advantage of the matrix-free approach, since it assembles the residual efficiently and avoids completely the assembly of the Jacobian.

To analyze the weak scalability of the matrix-free solver, several strong scaling tests were performed using both smoothers. The results are presented in **Figure 6.5**. As expected, the time to solution increases with an increasing number of refinements ℓ and the solver scales in the weak sense for all cases and both smoothers. The weak scalability can be observed by taking a look at times needed for the same order with increasing refinement using 1, 8 and 64 nodes. For example, in the case of Q_2Q_2 elements and the inverse-diagonal smoother (see the arrows in Figure 6.5), the time needed for 1 node ($\ell = 6$) is 22 seconds, for 8 nodes ($\ell = 7$) 31 seconds, and with 64 nodes ($\ell = 8$) 29 seconds; this indicates a weak efficiency of approximately 78% when increasing the number of processes by a factor of 64.[4]

A summary of the iterations needed for the non-linear and linear solver is presented in **Table 6.3** for the matrix-free cases in order to analyze closely the behavior of the non-linear and linear solvers. The results show that the inverse-diagonal smoother has to perform for all cases more linear iterations per Newton step than the ASM smoother. For both smoothers, the number of iterations slightly increases as the number of refinements ℓ increases; for ASM only up to 1 iteration for $p \geq 2$, while for inverse diagonal up to 3.5 for Q_2Q_2 and 4.3 for Q_3Q_3. In the case of linear elements, the increase is larger; however, the ASM converges for all cases, while no convergence is obtained when using the inverse diagonal already for lower refinements. Similar observations can also made for higher polynomial degrees. Appendix C in Section 6.9 shows results for Q_4Q_4. They indicate that ASM seems to be p-robust.

Table 6.4 presents the iterations of the matrix-based cases presented in the strong scaling study. The results showed that more linear solver iterations per Newton iterations are needed for all the cases compared to the matrix-free iteration count. Moreover, the number of iterations varies with the number of nodes used for the computation, with a difference of 5.2 iterations for the Q_1Q_1, 0.2 iterations for Q_2Q_2 and 1.2 iterations for Q_3Q_3.

[4]While this gives an indication of the weak scalability of the solver, it is important to keep in mind that there is always an error deviation when running simulations on clusters, due to, e.g., operating system overhead or parallel jobs. As a reference, we run the case with $\ell = 8$ and Q_3Q_3 elements (17M cells and 1B DoFs) 10 times and obtained a standard deviation of 1% on the total wallclock time.

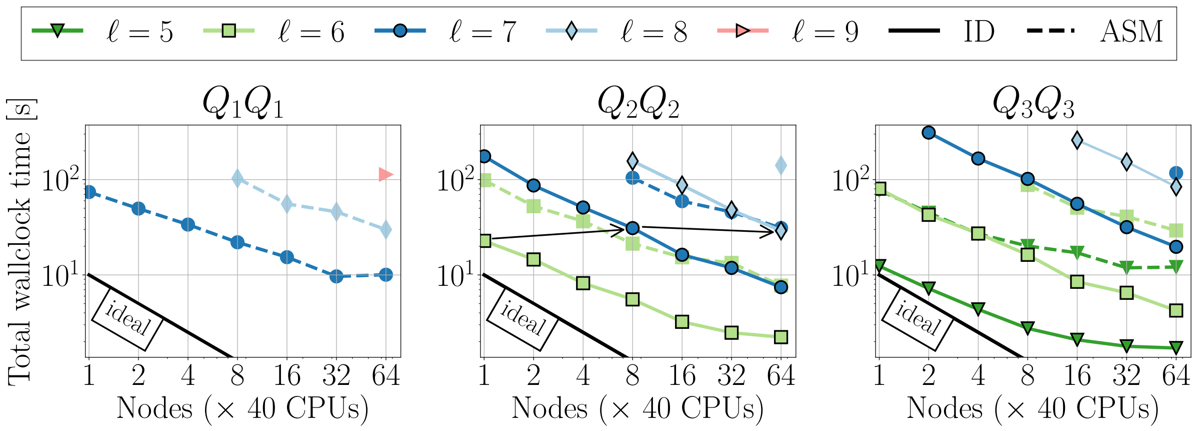

Figure 6.5 Several strong scaling simulations for the manufactured solution problem considering $7 \leq \ell \leq 9$ for Q_1Q_1, $6 \leq \ell \leq 8$ for Q_2Q_2 and $5 \leq \ell \leq 8$ for Q_3Q_3. As an example, the arrows indicate a weak scaling simulation with 212K DoFs per core and Q_2Q_2 elements.

Table 6.3 Summary of Newton iterations (N_N) and average linear solver iterations per Newton iteration ($\overline{N}_L$) needed to solve the manufactured solution cases with $4 \leq \ell \leq 9$ using two smoothers: inverse diagonal (ID) and additive Schwarz method (ASM). If "$-$" the simulation did not converge, if "$\star$" the simulation run out of memory.

	Q_1Q_1				Q_2Q_2				Q_3Q_3			
	ID		**ASM**		**ID**		**ASM**		**ID**		**ASM**	
ℓ	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$
4	3	7.0	3	4.3	3	7.0	3	4.7	3	7.7	3	5.7
5	–	–	3	5.0	3	7.7	3	5.0	3	8.3	3	6.0
6	–	–	3	5.3	3	7.7	3	5.3	3	9.0	3	6.7
7	–	–	3	6.0	3	8.7	3	5.3	3	9.3	3	6.7
8	–	–	3	6.0	2	10.5	3	5.7	2	12.0	$\star$	$\star$
9	–	–	2	7.5								

Table 6.4 Summary of Newton iterations (N_N) and average linear solver iterations per Newton iteration ($\overline{N}_L$) needed to solve the the manufactured solution cases with $6 \leq \ell \leq 8$ using the matrix-based version of the code for the strong scaling analysis.

Q_1Q_1			Q_2Q_2			Q_3Q_3		
ℓ	N_N	$\overline{N}_L$	ℓ	N_N	$\overline{N}_L$	ℓ	N_N	$\overline{N}_L$
6	3	13.8 ± 5.2	7	3	12.8 ± 0.2	8	3	13.5 ± 1.2

6.4.2 Flow Around a Sphere

This section considers a three-dimensional steady-state flow around a sphere with a high Reynolds number Re = 150. It allows us to show high-order capabilities of the solver in a problem that includes common challenges encountered when simulating external flows. In order to reach the desired Reynolds number, we ramp up the Reynolds number by performing six simulations with $\mathrm{Re} = 10, 19, 33, 55, 80, 104$ before proceeding to solve the case with $\mathrm{Re} = 150$ using the latest solution as an initial condition. A pseudotransient approach could be used instead; however, for this specific case, we found that a basic implementation requires more linear iterations and, therefore, leads to an increase in the total time of the simulation (even including the cost of the previous ramp steps). For other problems with a higher Reynolds number, the pseudo-transient approach might be more suitable. The domain for this case, along with all relevant geometric parameters, is shown in **Figure 6.6**. It consists of a flow with fixed entrance velocity $U_\infty = 1$ in the x direction, no-slip boundary conditions on the sphere and slip boundary conditions on the wall.

Several cases are considered with an increasing number of refinements ℓ; the corresponding number of cells and DoFs are presented in **Table 6.5**. In this case, the coarse grid consists of 1 024 cells; since we consider higher-order elements, the coarse grid is too expensive to solve with a direct solver (e.g., 121 244 DoFs when using Q_3Q_3 elements), therefore, we use a GMRES preconditioned by PMG with a direct solver for the inverse-diagonal smoother, and a PMG with a direct solver for the ASM smoother. All the Q_1Q_1 simulations have an element Peclet number in the range $\mathrm{Pe_h} \in [0.06, 6]$ for the smallest cell size and $\mathrm{Pe_h} \in [1.4, 93.4]$ for the largest cell size; therefore, this problem requires stabilization. The relevant quantity in this benchmark is the drag coefficient, which has two contributions, the friction coefficient C_τ and the pressure coefficient C_p. The coefficient is computed using the shear stresses (frictional forces F_τ) and the normal stresses (pressure forces F_p) as follows:

$$C_D = C_p + C_\tau = \frac{F_p}{\frac{1}{2}\rho U_\infty^2 \left(\frac{\pi}{4}D^2\right)} + \frac{F_\tau}{\frac{1}{2}\rho U_\infty^2 \left(\frac{\pi}{4}D^2\right)}.$$

The value of the drag coefficient for this problem can be obtained by fitting experimental data in the literature and corresponds to $C_D = 0.889$ [180]; all simulations performed obtained a drag coefficient that converges towards that value (not shown). The results can be better analyzed when the two contributions to the drag coefficient are plotted separately against the number of DoFs and the total wallclock time (see **Figure 6.7**).

The results for the pressure drag coefficient suggest that increasing the order of the finite

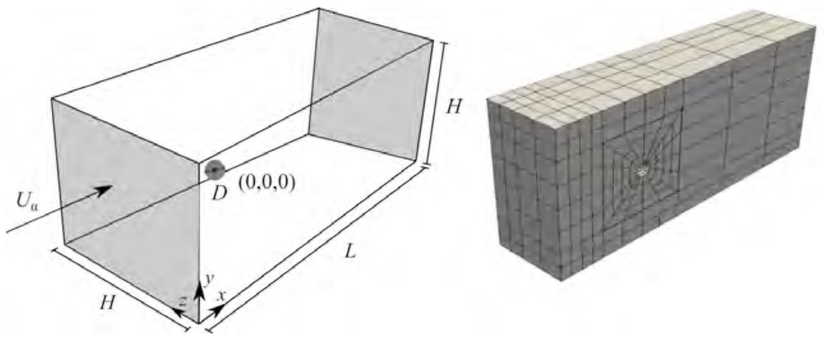

Figure 6.6 Three-dimensional geometry with $D = 1$, $H = 10$, $L = 22$ and $U_\infty = 1$, and clip of the coarse grid ($\ell = 0$) used for the flow around a sphere problem. The block-structured mesh consists of hexahedral-shaped cells distributed in twelve blocks. In addition a spherical manifold is located around the sphere to better capture its shape. For more details about the geometry and the mesh refer to Appendix E in Section 6.11.

Table 6.5 Summary of the refinement level, number of cells and number of degrees of freedom (DoFs) for all the simulation cases considered for the flow around a sphere problem. The initial mesh has 1,024 cells and this number increases with the refinement ℓ as $1\,024\ \times 2^{3\ell}$.

Refinement ℓ	**No. Cells**	**No. DoFs**		
	($1\,024 \times 2^{3\ell}$)	Q_1Q_1	Q_2Q_2	Q_3Q_3
1	8K	-	280K	926K
2	66K	280K	2.1M	7.2M
3	524K	2.1M	17M	57M
4	4M	17M	135M	455M
5	34M	135M	-	-

element does not increase the accuracy of the solution nor reduce the total wallclock time, regardless of the smoother used. However, in the case of the friction coefficient, it can be observed that the Q_1Q_1 cases considered never achieve the same accuracy as the simulation cases that consider $p = 2$ or 3; in fact, if we take a look at the simulation for Q_1Q_1 with $\ell = 5$ (135M DoFs) we obtain the same accuracy as the simulation for Q_2Q_2 using $\ell = 2$ (2.1M DoFs). This is impressive considering that it only has 1.6% of the number of DoFs of Q_1Q_1 and runs 54.7× faster in the case of the inverse-diagonal smoother and 68.5× faster in the case of the ASM smoother. In the case of Q_3Q_3, we also obtain better accuracy for both inverse diagonal and ASM, however, the gain in terms of computational time changes according to the smoother. In the case of ASM, the time for the Q_3Q_3 simulations is similar to the one for the Q_2Q_2 simulations; however, it has more DoFs and once the mesh is further refined $\ell = 4$, it quickly becomes more expensive. In the case of the inverse diagonal, the

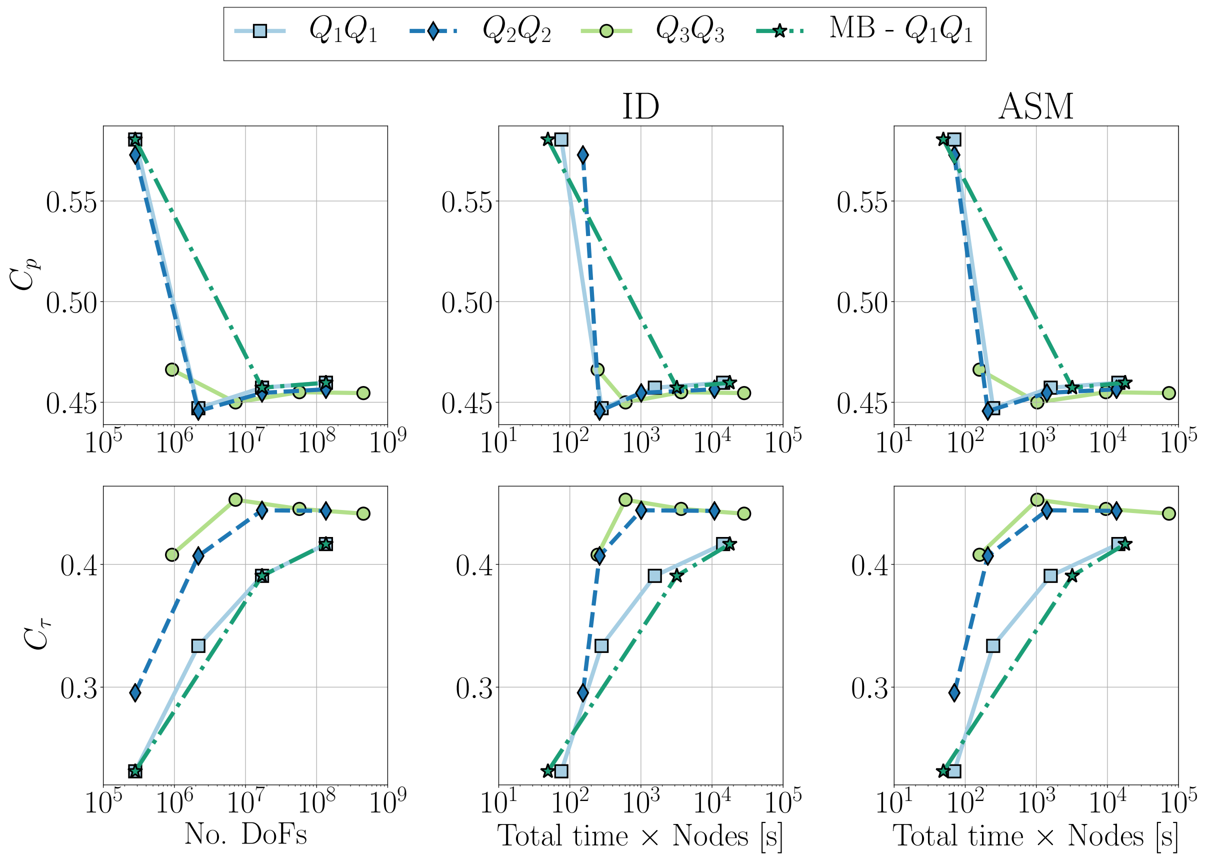

Figure 6.7 Results for the pressure drag coefficient (C_p) and the friction drag coefficient (C_τ) for different velocity-pressure discretizations against number of DoFs and total wallclock time using two smoothers within the GMG: inverse diagonal (ID) and additive Schwarz method (ASM).

simulations for Q_3Q_3 are more competitive in terms of time. This result enforces previous results obtained by Barbeau et al. [167] using a matrix-based solver for a similar setup but with a moving sphere.

In addition to matrix-free simulations, we run the Q_1Q_1 cases using the matrix-based solver. There are no results for the case with $\ell = 5$, due to the lack of convergence of the linear solver. This shows that this linear solver is not robust enough for this case; in fact, simulations with higher-order elements for these cases face similar issues when using the matrix-based solver. Despite this, the results obtained show the same accuracy as the Q_1Q_1 matrix-free results and are only faster than the matrix-free results for the case considering $\ell = 2$ (280K DoFs).

To further analyze the behavior of the preconditioners, a summary of the iterations needed for the nonlinear and linear solvers is presented in **Table 6.6** for the matrix-free results

Table 6.6 Summary of average Newton iterations ($\overline{N}_N$) per ramp step and average linear solver iterations per Newton iteration ($\overline{N}_L$), needed to solve the flow around a sphere cases with $1 \leq \ell \leq 5$ using two smoothers: inverse diagonal (ID) and additive Schwarz method (ASM). If "$-$" the simulation did not converge.

	$\boldsymbol{Q_1Q_1}$				$\boldsymbol{Q_2Q_2}$				$\boldsymbol{Q_3Q_3}$			
	ID		**ASM**		**ID**		**ASM**		**ID**		**ASM**	
$\boldsymbol{\ell}$	$\overline{N}_N$	$\overline{N}_L$	$\overline{N}_N$	$\overline{N}_L$	$\overline{N}_N$	$\overline{N}_L$	$\overline{N}_N$	$\overline{N}_L$	$\overline{N}_N$	$\overline{N}_L$	$\overline{N}_N$	$\overline{N}_L$
1					4.8	39.8	4.8	10.5	3.7	44.5	3.5	8.4
2	5.3	37.0	5.3	16.9	3.5	46.4	3.5	12.4	2.7	62.5	2.7	10.2
3	3.7	45.5	3.7	19.3	2.7	56.0	2.7	15.4	2.7	56.0	2.7	10.9
4	2.8	45.6	2.8	20.5	2.5	55.5	2.5	15.7	2.2	55.5	2.2	11.8
5	2.3	47.1	2.5	22.7								

Table 6.7 Summary of average Newton iterations ($\overline{N}_N$) per ramp step and average linear solver iterations per Newton iteration ($\overline{N}_L$), needed to solve the flow around a sphere cases with $2 \leq \ell \leq 5$ and $p = 1$ using the matrix-based version of the code for the strong scaling analysis. If "$-$" the simulation did not converge.

	$\boldsymbol{Q_1Q_1}$	
$\boldsymbol{\ell}$	$\overline{N}_N$	$\overline{N}_L$
2	3.5	15.4
3	$-$	$-$
4	2.7	23.5
5	2.5	32.0

Table 6.8 Summary of Newton iterations (N_N) and average linear solver iterations per Newton iteration ($\overline{N}_L$) for two ramp steps (Re = 10 and Re = 150) needed to solve the flow around a sphere cases with $2 \leq \ell \leq 5$ and $p = 1$. If "$-$" the simulation did not converge.

	ID				**ASM**				**MB**			
	Re =10		Re =150		Re =10		Re =150		Re =10		Re =150	
$\boldsymbol{\ell}$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$	N_N	$\overline{N}_L$
2	5	18.2	6	50.8	5	11.0	6	21.3	4	10.0	3	29.0
3	4	19.5	3	71.3	4	12.5	3	28	$-$	$-$	$-$	$-$
4	4	22.8	3	62.7	4	15.8	3	23	4	21.3	2	58.5
5	3	29.0	2	60.5	4	16.8	2	25.5	4	24.0	2	84.0

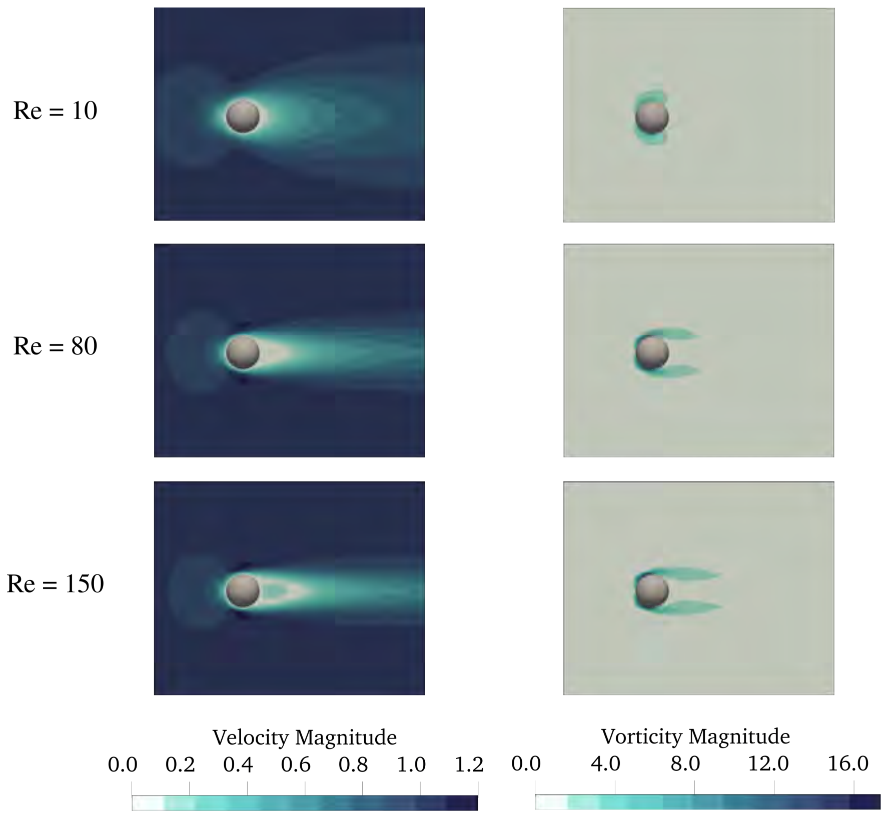

Figure 6.8 Zoom-in of the velocity and vorticity profiles on a xy-plane plane for the flow around a sphere for Re $= 10, 80$ and 150 obtained using Q_2Q_2 elements and $\ell = 2$.

and in **Table 6.7** for the matrix-based results. As in the previous steady-state case, the inverse-diagonal smoother has a larger increase in iteration count for all element degrees and all cases than the ASM smoother. However, for this case, the variation is larger and with a difference of up to 10.1 iterations for Q_1Q_1, 16.2 iterations for Q_2Q_2, and 18 iterations for Q_3Q_3. A similar situation is observed for the Q_1Q_1 cases of the matrix-based solver, where the difference in iteration counts is up to 16.6 iterations. The ASM smoother, on the contrary, is more robust than the inverse-diagonal smoother, but of course at a higher computational expense in terms of memory and resources. This behavior is consistently observed for the different ramp steps taken to solve the problem using different Reynolds numbers, as can be seen in **Table 6.8**; note that the number of linear iterations increases with increasing Reynolds number, which agrees with the physical phenomena observed in **Figure 6.8**.

In terms of total number of linear iterations, the inverse-diagonal smoother has a higher number than the ASM for all cases. When performing the experiments, we realized that the results for the inverse diagonal depend more on the quality of the coarse-grid solution than the results for the ASM. This is the reason for choosing a GMRES preconditioned by PMG, since we perform more iterations at the coarse-grid level in contrast to one PMG v-cycle for the ASM. As a consequence the cases using the inverse-diagonal smoother spent more time on the coarse-grid solution than the cases using ASM. For example, for the case with Q_2Q_2 elements and $\ell = 2$, the matrix-free solver with the inverse diagonal spends 23% of the time solving the coarse grid, while the ASM only spends 0.8%. Additional experiments in which the number of smoothing steps was increased (not shown here), show that the results using the inverse-diagonal smoother are inconclusive as the number of linear iterations increases with increasing number of smoothing steps which should not be the case. All of this would lead us to conclude that the most robust option for these types of steady-state external flow problems is a GMG preconditioner with the ASM smoother.

6.4.3 Taylor–Green Vortex

The last example is a canonical problem in fluid dynamics and is often used as a transient benchmark for high-order methods: Taylor–Green vortex (**TGV**) [34]. It is useful to demonstrate the ability of the solver to accurately simulate vortex dynamics and the energy cascade in turbulent flows. In this case, the domain $\Omega = (-\pi, \pi)^3$ has periodic boundary conditions in all directions and a structured hexahedral mesh. We study the flow with a Reynolds number $\mathrm{Re} = 1600$ and a fixed Courant–Friedrichs–Lewy (**CFL**) number of 1.[5] The initial conditions are:

$$\begin{pmatrix} u_x \\ u_y \\ u_z \\ p^* \end{pmatrix} = \begin{pmatrix} \sin(x)\cos(y)\cos(z) \\ -\cos(x)\sin(y)\cos(z) \\ 0 \\ \frac{1}{16}\left(\cos(2x) + \cos(2y)\right)\left(\cos(2z) + 2\right) \end{pmatrix}.$$

We compute the energy decay using i) the kinetic energy E_k and ii) the enstrophy ξ as follows:

$$E_k = \frac{\rho}{\Omega}\int \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2} \to \epsilon(E_k) = -\frac{\mathrm{d}E_k}{\mathrm{d}t}, \quad \xi = \frac{\rho}{\Omega}\int \frac{\boldsymbol{\omega}\cdot\boldsymbol{\omega}}{2} \to \epsilon(\xi) = \frac{2\mu}{\rho}\xi,$$

where ω is the vorticity defined as $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$. The energy decay is i) the time derivative of the total kinetic energy or ii) the volumetric integral of the enstrophy. We plot both quantities against the experimental results. This allows us not only to validate our solver, but also to observe the numerical dissipation through the difference of the two measures of

[5]The CFL definition considered in this study is CFL$= (\|\boldsymbol{u}\|\Delta t/h)p^m$ with $m = 1$, where p corresponds to the degree of the finite element. In the literature, values for m between 1 and 2 are common for higher-order continuous Galerkin and discontinuous Galerkin methods [61, 111]. After several accuracy tests, we chose $m = 1$.

the energy dissipation. The matrix-based version of the solver has already been used for this benchmark using $p = 1, 2, 3$, where excellent agreement with the reference results of [34] were obtained (see [105]). In this work, we performed simulations using a finer mesh of 384^3 cells for all element orders and the results are presented in **Figure 6.9**. Results with almost no numerical dissipation are observed for Q_2Q_2 and Q_3Q_3. What particularly stands out is that these kinds of simulations can now be performed using a desktop computer or a small cluster without massive computational requirements. In **Figure 6.10**, an example of the isocontours of the magnitude of the velocity profile are presented.

We also study the scalability of the solver using the cases in **Table 6.9**. For this, we analyze the average time per time step in the simulation interval (6-10 s) and plot it against the number of nodes in **Figure 6.11**. We experimented with the inverse diagonal and the ASM smoother and found that both were robust, with the ASM obtaining less linear iterations per Newton step (2.7 for $\ell = 7$ and Q_1Q_1, 2.6 for $\ell = 6$ and Q_2Q_2, and 2.7 for $\ell = 5$ and Q_3Q_3; for inverse-diagonal results see **Table 6.10**); however, since the ASM smoother is more expensive, we decided to only show inverse-diagonal results for this benchmark. The matrix-free solver is 2.6-4.2× faster than the matrix-based solver for Q_1Q_1, 7.6-16.4× faster for Q_2Q_2, and 11.1-32.3× faster for Q_3Q_3. The matrix-free solver has, in general, a good scalability; the results obtained with 32 and 64 nodes for Q_2Q_2 and Q_3Q_3 elements reach the scaling limit and are the points where the lowest speedups are obtained. The matrix-based approach also shows good scalability for all element orders, in contrast to the behavior of the solver for steady-state cases; in this case, ILU is used as preconditioner. In terms of memory requirements, the results agree with what was observed for the steady-state cases, the matrix-free solver has less memory requirements than the matrix-based solver which allow us to run the Q_2Q_2 and Q_3Q_3 problems using 1 and 2 nodes.

To analyze in detail the time spent per time step, **Figure 6.12** presents the time spent for the Q_2Q_2 cases broken down into the setup of the preconditioner (GMG or ILU), the solution of the linear system, and other parts of the simulations. It is important to mention that for this problem we reuse the preconditioner across Newton iterations for both the matrix-based and the matrix-free solver, which means that we only setup the preconditioner once per transient iteration. The results show that the matrix-free solver cost in terms of time is dominated by the setup of the preconditioner followed by the solution of the linear system and the other components. In contrast, the matrix-based solver is dominated by the cost of the other components, specifically the cost of the assembly of the Jacobian and the residual.[6] In the

[6]We note that using an inexact Newton method, that does not assemble the Jacobian in every Newton iteration but possibly increases the nonlinear and linear iterations, would be an option to reduce the setup costs of the matrix-based approach.

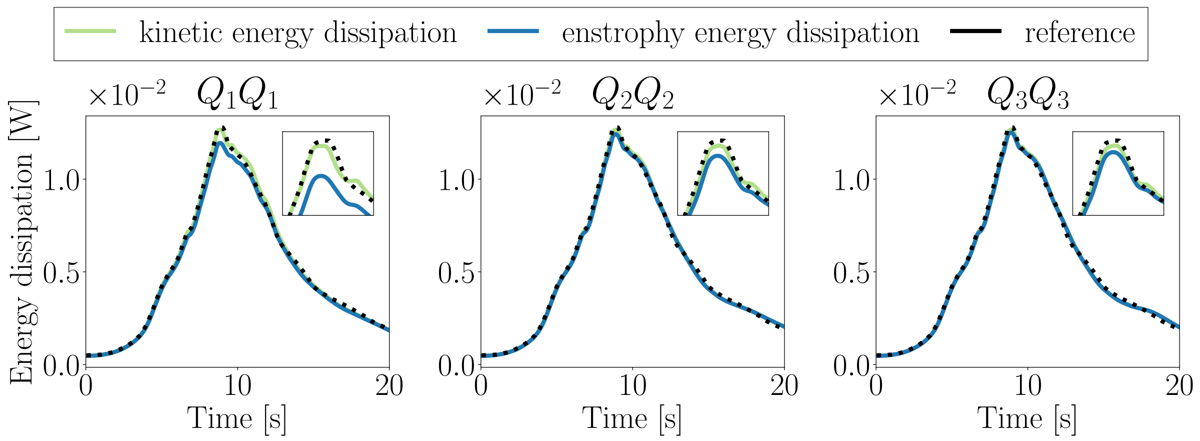

Figure 6.9 Energy dissipation comparison for a Taylor–Green vortex simulation using a 384^3 mesh for Q_1Q_1, 192^3 mesh for Q_2Q_2 and 128^3 mesh for Q_3Q_3, all with a total number of 228M degrees of freedom. The results are compared to the reference results of Wang (2013) [34].

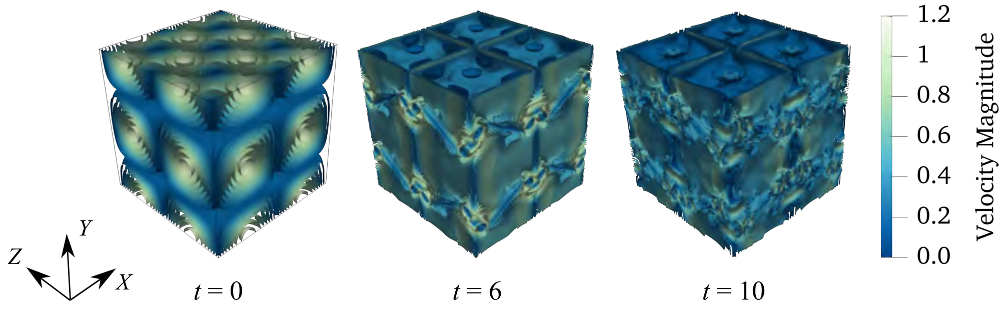

Figure 6.10 Isocontours of the velocity magnitude at different times for the Taylor–Green vortex problem obtained using Q_1Q_1 elements and $\ell = 7$.

case of the simulation performed with 8 nodes, the other components represent 53% of the time per time step for the matrix-based solver, while for the matrix-free solver, this value is only 11%.

Weak scalability is also studied by running several strong-scaling simulations with up to 2 560 computing cores (64 nodes); the results are presented in **Figure 6.13** for the total wallclock time. As in the MMS case, the time to solution increases with an increasing number of refinements ℓ and the solver scales in the weak sense for all cases. The arrows in Figure 6.13 highlight the case of Q_2Q_2 elements with 262K cells per node; the results show that the time increases as the refinement ℓ increases, which makes sense since more transient iterations are needed (80 for $\ell = 6$, 167 for $\ell = 7$, and 365 for $\ell = 8$). The same plot is shown for the

Table 6.9 Summary of the refinement level, number of cells and number of degrees of freedom (DoFs) for all the simulation cases considered for the Taylor–Green vortex problem. All the Q_1Q_1 simulations have an element Peclet number in the range $\mathrm{Pe_h} \in [12, 47]$, and therefore, require SUPG and PSPG stabilization.

Refinement ℓ	**No. Cells**	**No. DoFs**		
	$(1 \times 2^{3\ell})$	Q_1Q_1	Q_2Q_2	Q_3Q_3
5	33K			3.6M
6	262K		8M	28M
7	2M	8M	68M	228M
8	17M	68M	540M	
9	134M	540M		

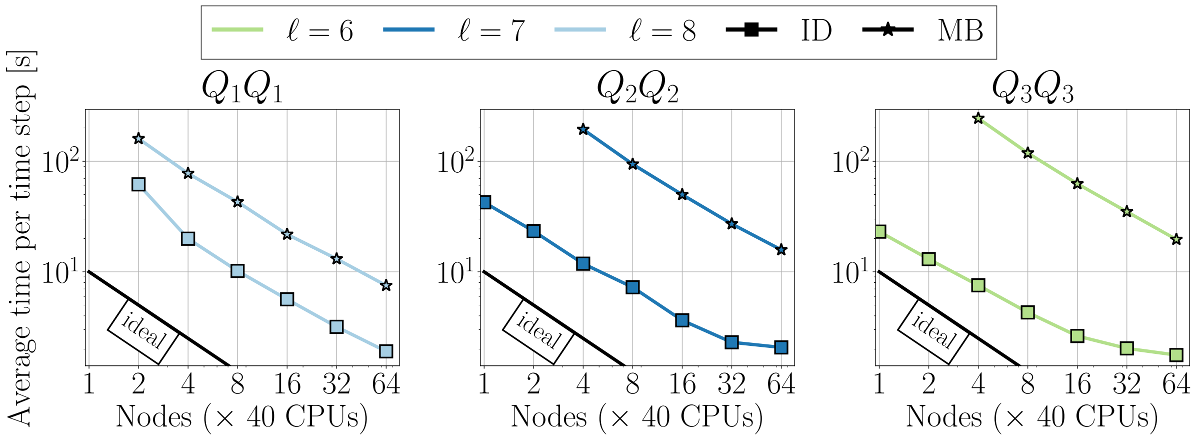

Figure 6.11 Strong scaling study considering $\ell = 8$ for Q_1Q_1, $\ell = 7$ for Q_2Q_2 and $\ell = 6$ for Q_3Q_3 for the Taylor–Green vortex problem in the time interval $6 \leq t \leq 10$.

average time per time step in **Figure 6.14**. The results for the Q_2Q_2 cases show the opposite behavior, the average time per time step decreases with increasing refinement ℓ (see arrows in Figure 6.14). The reason for this is that the linear iterations per Newton step decrease as we increase the refinement (4.7 for $\ell = 6$, 3.7 for $\ell = 7$, and 3.2 for $\ell = 8$). These results indicate that to improve performance we could increase the CFL and perform less transient iterations with more linear iterations per Newton step for larger refinements. However, one should be careful when choosing the CFL as the accuracy of the results should not be compromised.

A summary of the iterations needed for the non-linear and linear solver is presented in Table 6.10 for the matrix-free solver and in **Table 6.11** for the matrix-based solver. The results show that both solvers perform the same amount of transient iterations; however,

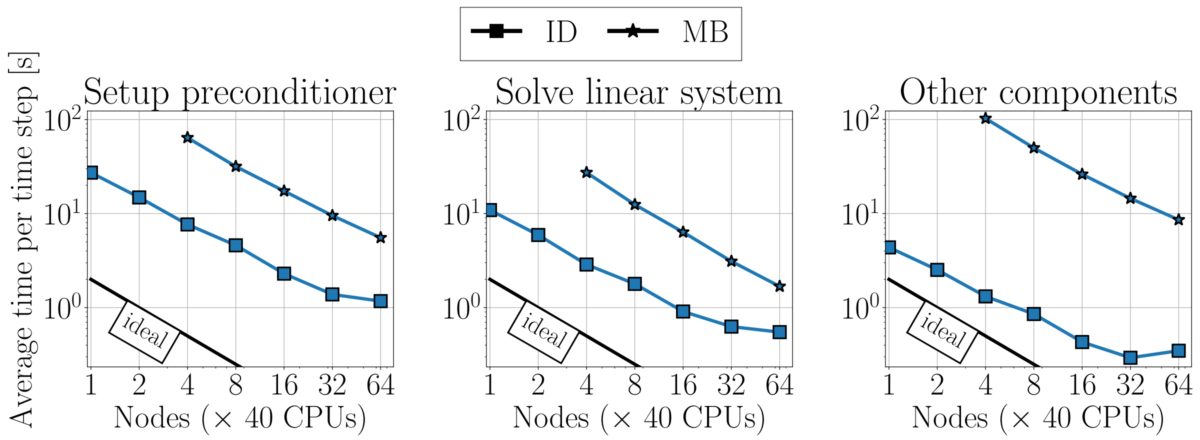

Figure 6.12 Detailed timings for the setup of the preconditioner, time to solution of the linearized problem and other components (mesh creation, Jacobian assembly, residual assembly, and computation of enstrophy and kinetic energy) for the strong scaling study for the Taylor–Green vortex problem using Q_2Q_2, $\ell = 7$ and a the time interval $6 \leq t \leq 10$ (see Figure 6.11 for total times).

Table 6.10 Summary of transient iterations (N_T), average Newton iterations ($\overline{N}_N$) per transient iteration and average linear solver iterations per Newton iteration ($\overline{N}_L$), needed to solve the TGV cases with $5 \leq \ell \leq 6$ using the inverse diagonal (ID) smoother.

	$\boldsymbol{Q_1Q_1}$			$\boldsymbol{Q_2Q_2}$			$\boldsymbol{Q_3Q_3}$		
$\boldsymbol{\ell}$	N_T	$\overline{N}_N$	$\overline{N}_L$	N_T	$\overline{N}_N$	$\overline{N}_L$	N_T	$\overline{N}_N$	$\overline{N}_L$
5							68	5.7	4.6
6				80	4.7	4.0	129	4.0	4.6
7	77	4.1	3.9	167	3.1	3.7	264	2.3	4.4
8	165	2.6	3.2	345	1.8	3.2			
9	360	1.8	2.4						

Table 6.11 Summary of transient iterations (N_T), average Newton iterations ($\overline{N}_N$) per transient iteration and average linear solver iterations per Newton iteration ($\overline{N}_L$), needed to solve the TGV cases with $5 \leq \ell \leq 6$ using the matrix-based solver.

$\boldsymbol{Q_1Q_1}$				$\boldsymbol{Q_2Q_2}$				$\boldsymbol{Q_3Q_3}$			
$\boldsymbol{\ell}$	N_T	$\overline{N}_N$	$\overline{N}_L$	$\boldsymbol{\ell}$	N_T	$\overline{N}_N$	$\overline{N}_L$	$\boldsymbol{\ell}$	N_T	$\overline{N}_N$	$\overline{N}_L$
8	165	2.0	42.7 ± 4.6	7	167	2.0	45.1 ± 5.2	6	129	2.4	41.2 ± 5.8

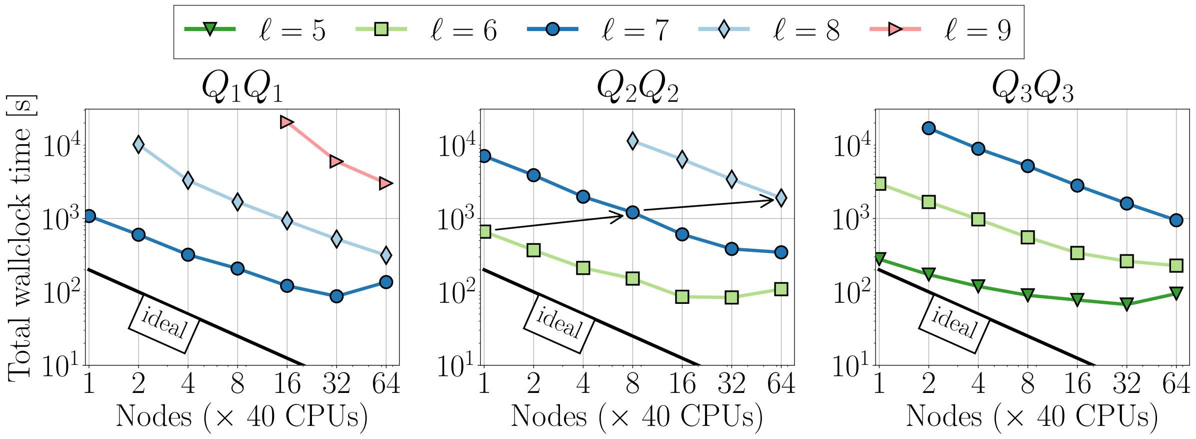

Figure 6.13 Total time for several strong scaling simulations for the Taylor–Green vortex problem considering $7 \leq \ell \leq 9$ for Q_1Q_1, $6 \leq \ell \leq 8$ for Q_2Q_2 and $5 \leq \ell \leq 8$ for Q_3Q_3 in the time interval $6 \leq t \leq 10$. The arrows show how to observe the weak scalability in this plot.

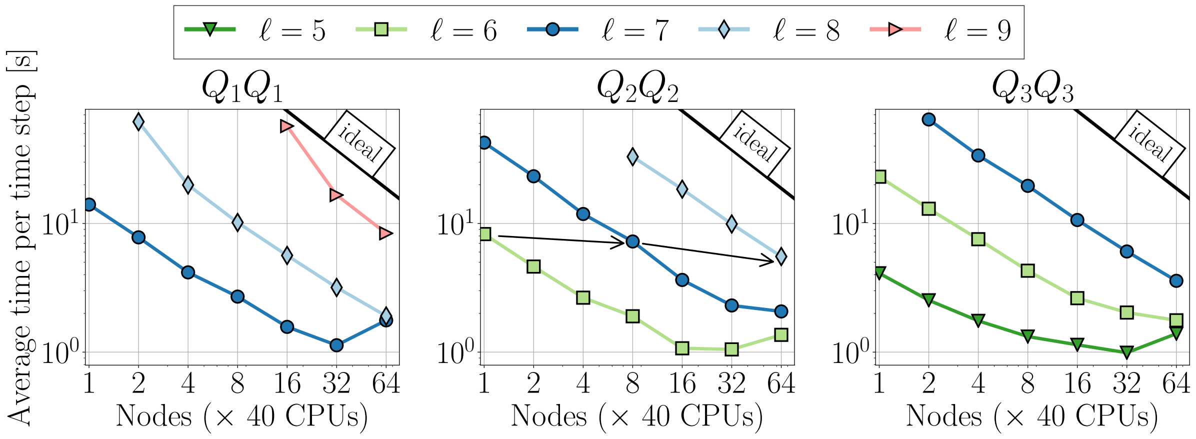

Figure 6.14 Average time per time step for several strong scaling simulations for the Taylor–Green vortex problem considering $7 \leq \ell \leq 9$ for Q_1Q_1, $6 \leq \ell \leq 8$ for Q_2Q_2 and $5 \leq \ell \leq 8$ for Q_3Q_3 in the time interval $6 \leq t \leq 10$. The arrows show how to observe the weak scalability in this plot.

small differences can be observed for the average number of Newton iterations and linear iterations. Most notably, the GMG preconditioner always requires a small number of linear iterations per Newton step (2–5), while the ILU preconditioner needs more than 40 iterations for all the element orders studied. However, one needs to keep in mind that GMG performs in total 10 smoothing steps per iteration and level. One can conclude that the monolithic GMG preconditioner shows excellent behavior for transient problems, and it is robust with a simple inverse-diagonal smoother. For a recent publication that uses the same benchmark to test a matrix-free finite-volume solver with a pressure-projection resolution technique instead, see the work by Knaus [181].

6.5 Conclusions and Further Research

We have presented a stabilized solver for the incompressible Navier–Stokes equations implemented in a matrix-free fashion in the context of continuous Galerkin finite element methods. The solver uses a monolithic geometric multigrid (GMG) preconditioner for a Newton–Krylov solver. We presented experimental results for two steady-state problems and one transient problem, and consider two types of smoothers for the GMG preconditioner: an inverse diagonal and a cell-centric additive Schwarz method (ASM).

For steady-state problems, the solver scales for up to 2 560 computing cores and one billion DoFs. The results indicate that the ASM smoother is more robust than the inverse-diagonal smoother for all cases considered, however, with a significant higher memory requirement. Compared against its matrix-based version with an AMG preconditioner, the matrix-free solver showed lower times to solution (1.6× faster for Q_1Q_1 with ASM, 10–100× faster for higher-order elements with inverse diagonal) and a better scalability. For the *flow around a sphere* benchmark with a high Reynolds number (Re = 150), we demonstrated that high-order elements in the context of continuous Galerkin can achieve higher accuracy when computing the friction drag coefficient with a lower number of degrees of freedom. This conclusion could only be obtained via the matrix-free solver, since its matrix-based version - along with its linear solver – is not robust and is computationally too expensive for higher order elements.

For transient turbulent problems, the results indicate good scalability for problems with up to 2 560 computing cores and half a billion DoFs. In this case, the linear solver was robust for all cases, even with a simple inverse-diagonal smoother. Compared to the matrix-based version with an ILU preconditioner, the matrix-free solver was significantly faster even for linear elements (3.4× for Q_1Q_1 elements, 10-30× faster for higher-order elements). The results show that due to the high computational efficiency and the lower memory requirements of the solver, one can obtain fully converged simulations of the Taylor–Green vortex benchmark using a desktop computer or a small cluster.

To summarize, the solver presented in this study, along with its GMG preconditioner, allows us to simulate steady-state and transient problems with a competitive time-to-solution and an efficient use of computational resources, especially when using higher order elements. Further research directions include the extension of the monolithic GMG solver for locally refined meshes in the context of more challenging engineering applications, a detailed comparison of the solver to a pressure-projection implementation approach, and the development of a more efficient implementation of the ASM smoother. Latter also includes the investigation of different patch types, in particular, of vertex-star patches, which are the basis of Vanka smoothers.

6.6 Acknowledgments

The authors acknowledge collaboration with the `deal.II` community. The authors also acknowledge technical support and computing time provided by the Digital Research Alliance of Canada and Calcul Québec. BB acknowledges financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) through the RGPIN-2020-04510 Discovery Grant and the funding from the Multiphysics Multiphase Intensification Automatization Workbench (MMIAOW) Canadian Research Chair Level 2 in computer-assisted design and scale-up of alternative energy vectors for sustainable chemical processes (CRC-2022-00340). PM acknowledges the funding by the Swedish Research Council (VR) under grant 2021-04620 and the strategic funding from the IT department of Uppsala University.

6.7 Appendix A : Smoother Iterations

In order to choose the number of smoother iterations used in this work, we consider the flow around a sphere case with a fixed refinement. We perform several experiments with different number of smoothing iterations (SI) between 1 and 7, elements Q_pQ_p of order $p = 1, 2, 3$, and the two smoothers: inverse diagonal (ID) and additive Schwarz method (ASM). The results are presented in **Tables 6.12** and **6.13**. For the Newton iterations the same results as in Table 6.6 are obtained (not shown). Using one or two smoother iterations for the ASM smoother is already enough in terms of time to solution and linear iteration count. Even though with more smoothing steps the number of linear iterations further decreases, the accumulated number of expensive smoothing steps increases, leading to an overall increase in time to solution. For the inverse diagonal smoother, more smoother iterations are needed to obtain the lowest time to solution, which is related to costs of other parts of multigrid solver, which are not negligible compared to the extremely cheap smoother. Therefore, we choose 2 smoothing iterations for the ASM smoother and 5 for the ID smoother.

Table 6.12 Summary of total time in seconds multiplied by the number of nodes (T), average linear solver iterations per Newton iteration ($\overline{N}_L$), and average smoothing steps per level per Newton iteration ($\overline{N}_S = 2 \cdot \text{SI} \cdot \overline{N}_L$), needed to solve the flow around a sphere case with two global refinements with different number of smoothing iterations ($1 \leq \text{SI} \leq 7$) using the inverse diagonal (ID) as smoother. If "$-$" the simulation reached the maximum number of iterations set to 200.

	$\boldsymbol{Q_1Q_1}$			$\boldsymbol{Q_2Q_2}$			$\boldsymbol{Q_3Q_3}$		
SI	T	$\overline{N}_L$	$\overline{N}_S$	T	$\overline{N}_L$	$\overline{N}_S$	T	$\overline{N}_L$	$\overline{N}_S$
1	–	–	–	–	–	–	–	–	–
2	87	82.1	341.3	–	–	–	–	–	–
3	81	57.8	361.3	329	70.4	438.9	–	–	–
4	71	43.8	365.8	273	55.2	391.2	670	77.2	641.0
5	72	37.0	386.9	255	46.4	483.8	620	62.5	649.4
6	74	32.5	408.8	263	38.6	484.4	610	53.1	663.8
7	70	28.0	412.6	260	34.1	499.5	604	45.8	669.4

Table 6.13 Summary of total time in seconds multiplied by the number of nodes (T), average linear solver iterations per Newton iteration ($\overline{N}_L$), and average smoothing steps per level per Newton iteration ($\overline{N}_S = 2 \cdot \text{SI} \cdot \overline{N}_L$), needed to solve the flow around a sphere case with two global refinements with different number of smoothing iterations ($1 \leq \text{SI} \leq 7$) using the additive Schwarz method (ASM) as a smoother. If "$-$" the simulation reached the maximum number of iterations set to 200.

	$\boldsymbol{Q_1Q_1}$			$\boldsymbol{Q_2Q_2}$			$\boldsymbol{Q_3Q_3}$		
SI	T	$\overline{N}_L$	$\overline{N}_S$	T	$\overline{N}_L$	$\overline{N}_S$	T	$\overline{N}_L$	$\overline{N}_S$
1	73	31.3	65.7	234	21.5	45.3	1152	18.1	38.1
2	68	16.9	71.8	232	12.4	53.7	1188	10.2	44.8
3	72	12.2	79.3	244	9.0	60.3	1240	7.4	50.6
4	72	9.9	87.3	249	7.4	67.4	1260	6.1	56.5
5	72	8.3	93.4	245	6.3	73.3	1268	5.4	63.8
6	72	7.3	99.0	259	5.7	80.0	1892	4.9	70.5
7	75	6.7	107.2	268	5.3	88.0	1884	4.4	76.1

6.8 Appendix B : AMG and ILU Parameters

A complete overview of the parameters used by the Trilinos ML AMG preconditioner and Trilinos Ifpack ILU preconditioner (version 13.4.1) is presented in Figures 6.15 and 6.16 respectively. This set of parameters has been successfully used by the matrix-based code for a wide range of applications; see [106] for some examples.

```
Teuchos::ParameterList parameter_list;
ML_Epetra::SetDefaults("NSSA", parameter_list);
parameter_list.set("aggregation: type", "Uncoupled");
// General parameters
parameter_list.set("aggregation: block scaling", true);
parameter_list.set("smoother: type", "ILU");
parameter_list.set("coarse: type", "ILU");
parameter_list.set("initialize random seed", true);
parameter_list.set("smoother:sweeps", 2);
parameter_list.set("cycle applications", 1);
parameter_list.set("prec type", "MGV");
parameter_list.set("smoother: Chebyshev alpha", 10.);
parameter_list.set("smoother: ifpack overlap", 1);
parameter_list.set("aggregation: threshold", 1e-4);
parameter_list.set("coarse: max size", 2000);
// Null space
parameter_list.set("null space: type", "pre-computed");
parameter_list.set("null space: dimension", 4);
parameter_list.set("null space: vectors", constant_modes);
// Smoother parameters
parameter_ml.set("smoother: ifpack level-of-fill", 0);
parameter_ml.set("smoother: ifpack absolute threshold", 1e-12);
parameter_ml.set("smoother: ifpack relative threshold", 1.0);
// Coarse parameters
parameter_ml.set("coarse: ifpack level-of-fill", 0);
parameter_ml.set("coarse: ifpack absolute threshold", 1e-12);
parameter_ml.set("coarse: ifpack relative threshold", 1.0);
```

Figure 6.15 Input parameters of the AMG preconditioner as implemented in Trilinos ML [119].

```
Teuchos::ParameterList parameter_list;
parameter_list.set("fact: level-of-fill", 0);
parameter_list.set("fact: absolute threshold", 10e-10);
parameter_list.set("fact: relative threshold", 1);
parameter_list.set("schwarz: combine mode", "Add");
```

Figure 6.16 Input parameters of the ILU preconditioner as implemented in Trilinos Ifpack [120].

6.9 Appendix C : Manufactured Solution using Q_4Q_4 Elements

The publication focuses on polynomial orders $p = 1, 2, 3$ since in practice, when using continuous Galerkin methods, these are the orders most commonly used. However, our matrix-free implementation can also be used with higher orders such as Q_4Q_4 elements. In what follows, we use this capability for the manufactured solution problem. We consider three refinement levels ($4 \leq l \leq 6$) cases and run the simulations to check the order of convergence as well as the number of nonlinear and linear iterations. All the results are presented in **Table 6.14**. The results show that the convergence is preserved even when using an order $p = 4$, obtaining an order of convergence of $O(h^5)$ for the velocity and $O(h^4)$ for the pressure. In terms of number of iterations, the results show similar results as those obtained for polynomial orders $p < 4$ (see Table 6.3). The ASM smoother preserves the same number of linear iterations per Newton iteration as in the Q_3Q_3 cases and the iterations do not increase as much as in the ID case. On the other hand, the inverse diagonal smoother is less robust, and the number of iterations increases.

6.10 Appendix D : Additional Verification via a Taylor Vortex in 2D

We simulate a Taylor Vortex in 2D in a domain $\Omega = (0, 2\pi)^2$ with periodic boundary conditions, where the velocity and pressure are defined as follows:

$$\begin{pmatrix} u_x \\ u_u \\ p^* \end{pmatrix} = \begin{pmatrix} \cos(x)\sin(y)e^{-2\nu t} \\ -\sin(x)\cos(y)e^{-2\nu t} \\ \frac{1}{4}(\cos(2x) + \cos(2y))e^{-4\nu t} \end{pmatrix}. \tag{6.9}$$

This problem allows us to verify the solver for structured and unstructured meshes considering a more challenging problem. We consider four unstructured meshes. The first three are generated by transforming each of the points of the cartesian mesh (x, y) as:

$$x' = x + \beta \sin(2\pi y) \tag{6.10}$$

$$y' = y + \beta \sin(2\pi y) \tag{6.11}$$

where β is set to $0.15, 0.45$ and 0.75. The last unstructured mesh is generated randomly using Gmsh with its BAMG algorithm. In Figure 6.17, we present the results for elements Q_1Q_1 and Q_3Q_3 as these are the lowest and largest orders considered in the manuscript. The results are presented for a Reynolds number $\text{Re} = 10\,000$ ($\text{Pe}_\text{h} \in [491, 1963]$) and for the inviscid case ($\text{Pe}_\text{h} \to \infty$) (as proposed by Ham and Iaccarino [177]). The results show that the correct order of accuracy is recovered for the velocity, that is $\mathcal{O}(\Delta h^{p+1})$, i.e., second order for Q_1Q_1 elements and fourth order for Q_3Q_3 elements. Moreover, the order of accuracy is preserved even for highly unstructured meshes.

Table 6.14 Summary of the refinement level, number of cells, number of degrees of freedom (DoFs), L$_2$ error for the velocity and the pressure, Newton iterations (N_N), and average linear solver iterations per Newton iteration ($\overline{N}_L$) for the three simulation cases considered for the extension of the manufactured solution problem using Q_4Q_4 elements. To generate the mesh, a coarse mesh consisting of a single cell is created and then refined ℓ times. The number of cells increases with $2^{3\ell}$.

Refinement $\boldsymbol{\ell}$	**No. Cells** ($1 \times 2^{3\ell}$)	**No. DoFs**	**L$_2$error $\boldsymbol{u}$**	**L$_2$error p**	**ID**		**ASM**	
					N_N	$\overline{N}_L$	N_N	$\overline{N}_L$
4	4K	1M	2.97×10^{-6}	1.04×10^{-4}	3	11.7	3	5.7
5	33K	8M	9.26×10^{-8}	6.33×10^{-6}	3	13.0	3	6.0
6	262K	67M	2.89×10^{-9}	3.94×10^{-7}	3	14.0	3	6.7

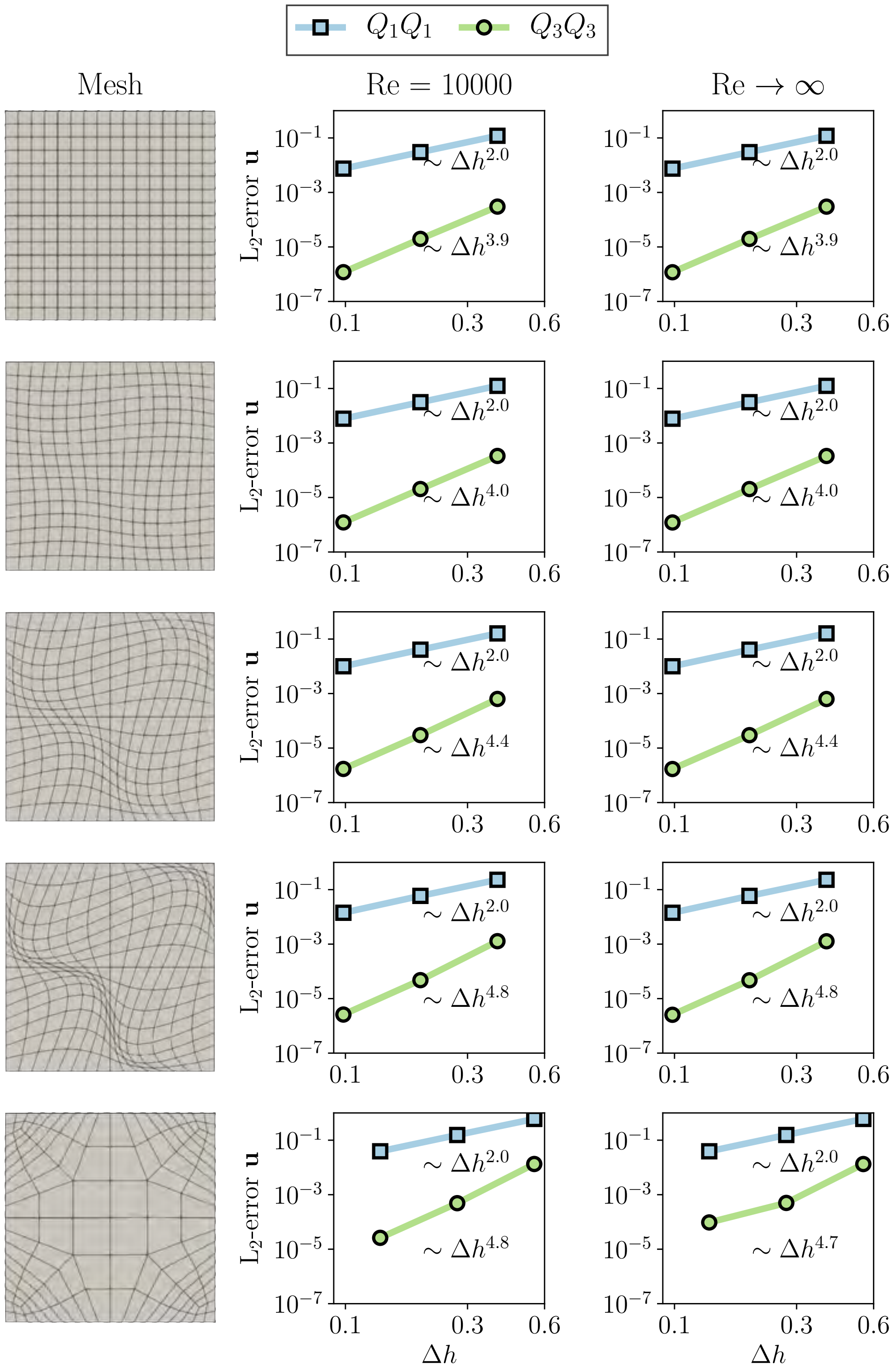

Figure 6.17 Order of convergence for velocity-pressure discretizations Q_1Q_1 and Q_3Q_3 for the Taylor-Vortex in 2D problem.

6.11 Appendix E : Details for the Flow Around a Sphere mesh

The domain is divided into 12 blocks of different sizes. Each block is divided using hexahedral-shaped cells of different sizes. Figure 6.18 shows the dimensions of each of these blocks. All blocks, with the exception of block 12, which contains the sphere, are Cartesian blocks. The largest cells are located downstream of the sphere and the smallest cells are located in the block where the sphere is located. The geometry and mesh files can be found in the Lethe GitHub repository (`https://github.com/chaos-polymtl/lethe/tree/master/examples/incompressible-flow/3d-flow-around-sphere`).

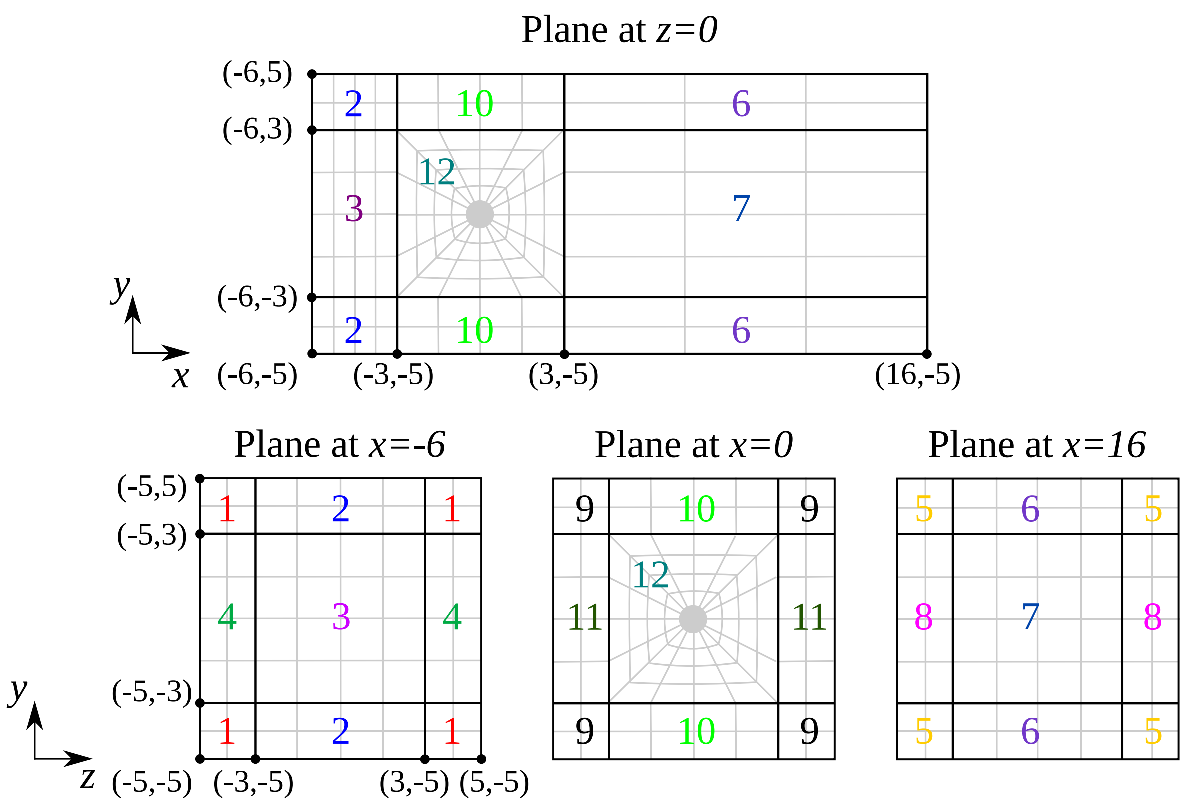

Figure 6.18 Schematic representing all the blocks needed to create the block-structured mesh used for the flow around a sphere case.

CHAPTER 7 ARTICLE 3 : AN IMPLICIT LARGE-EDDY SIMULATION STUDY OF THE TURBULENT TAYLOR–COUETTE FLOW WITH AN INNER ROTATING CYLINDER

Laura Prieto Saavedra, Joël Archambault, Peter Munch, Bruno Blais.

Accepted in the Journal of Turbulence, on 20 June 2025. Available at `https://doi.org/10.1080/14685248.2025.2524336`. *Part of the content was presented at the ECCOMAS Conference in Lisbon in June 2024.*

Declaration: The author of this thesis has contributed to the writing of the original draft, the visualization, the validation, the software, the methodology, the investigation, the formal analysis, the data curation and the conceptualization.

Abstract: The Taylor–Couette flow case is a turbulent benchmark that assesses the capability of numerical methods to simulate problems with curved boundaries and boundary layers. In this study, we consider the case with a rotating inner cylinder and a stationary outer cylinder at a Reynolds number Re = 4000. This allows us to assess the accuracy of our continuous Galerkin finite element solver, which uses an implicit Large-Eddy Simulation (LES) approach and SUPG/PSPG stabilization techniques. We perform numerical experiments using different polynomial orders $p = 1, 2, 3$ with up to 6M cells and 716M degrees of freedom. We compare enstrophy and kinetic energy profiles, along with vorticity and Q criterion distributions. Moreover, we compute the numerical dissipation of the implicit LES approach using the energy equation. The results show that the prediction of enstrophy is more accurate with increasing order p and refinement of the mesh. The energy balance analysis allows us to show that high-order elements can obtain less numerical dissipation with a lower number of degrees of freedom and with fewer computational resources. The work raises the question of what is an acceptable amount of numerical dissipation and provides valuable data for future users of this benchmark.

7.1 Introduction

Understanding rotating flows is critical for the design of devices with applications in different domains, such as, wind turbines [182, 183], jet engines [184], rotating mixers [185], and rotating reactors [186, 187]. One of the most investigated rotating flow problems in fluid mechanics is known as the Taylor–Couette flow. It consists of the flow between two rotating coaxial cylinders and is relevant for several reasons: i) it is possible to study it with high precision experimentally due to its simple geometry and symmetry, ii) its curved boundaries allow the study of boundary layers and their interaction with the bulk flow, as well as iii) it is a mathematically well defined problem described by the Navier–Stokes equations.

The Taylor–Couette flow has been widely studied experimentally. A comprehensive review can be found in the review article of Grossmann et al. [188]. For independent rotating cylinders, experiments were conducted in the early 30s by Wendt et al. [189]. Andereck et al. [190] conducted experiments that allowed building a phase diagram that showed the different flow structures that could be obtained depending on the Reynolds number of the inner and outer cylinders. The flow structures led to different *phase* zones, which were classified according to the physical phenomena observed, e.g., Couette flow physics, Taylor-vortices, modulated waves, spiral turbulence, among others. The experimental work of Lathrop et al. [191, 192] and Lewis et al. [193] is also commonly cited in the literature for the Taylor–Couette flow with pure inner-cylinder rotation.

Numerical simulations of this benchmark started to be conducted thereafter. Direct numerical simulations **(DNS)** were conducted, for example, by Dong [194], who used a Fourier series in the axial direction and spectral elements otherwise to compute DNS data between $\mathrm{Re} = 1000$ and $\mathrm{Re} = 8000$. Pirrò & Quadrio [195] conducted simulations at $\mathrm{Re} = 10\,500$ using a mixed spatial discretization consisting of spectral schemes in the homogeneous directions and a fourth-order finite difference scheme in the radial direction. They compared velocity fluctuations with experimental data and observed large-scale vortices and wall turbulent cycles as a result of near-wall shear. In the work of Ostilla-Mónico et al. [196], they performed DNS simulations using a second-order finite difference code with fractional time-stepping at a Reynolds number of $\mathrm{Re} = 100\,000$. They studied the effect of the size of the computational domain in the statistics (low- and high-order moments of the flow) and experimented with "computational boxes" to reduce computational requirements.

While DNS simulations seem to be the focus, there are also some works that used large-eddy simulations **(LES)** to study the Taylor–Couette flow. The work of Poncet el al. [197], considers LES simulations of this benchmark for six different combinations of rotational and axial Reynolds numbers. Cheng et al. [198] also performed wall-resolved LES simulations for the case where the inner cylinder is in rotation and the outer cylinder is stationary. They used a stretched-vortex subgrid-scale model, but the flow was resolved close to the wall for the Reynolds number range $\mathrm{Re} = 100\,000 - 1\,000\,000$. The first work showing results using an implicit LES strategy is that by Bazilevs & Akkerman [199], where the benchmark was used to evaluate the convergence characteristics of an isogeometric analysis method along with a residual-based variational multiscale formulation for the flow at $\mathrm{Re} = 8000$. The mean velocities, the mean angular momentum, and the torque were compared to the DNS results by Dong [194]. They highlighted that the use of NURBS (non-uniform rational B-splines) allows one to exactly represent circular geometries, which contributed to obtaining good accuracy for all statistical data (ensemble-averaged mean azimuthal velocity, angular momentum, and azimuthal velocity fluctuations as functions of the radial coordinate). They also emphasized that the good performance of the methodology is highly dependent on the decision of not using explicit eddy viscosities, since eddy-viscosity-based modeling approaches have problems when dealing with a flow dominated by rotation. Aydinbakar et al. [200] also studied the flow using a space-time variational multiscale method with space-time isogeometric discretization as well as discussed the importance of a reasonable computational cost and an accurate description of the curved walls.

Although most of the literature on this benchmark focused on comparing azimuthal velocity, angular momentum, and torque profiles, with experimental or DNS data, Wang & Jourdan [201] proposed a different perspective to study this benchmark. The authors argue that comparing average quantities is time consuming and requires significant computational resources due to the long averaging time needed (once a statistically-stationary state is reached, around 250 non-dimensional units according to [199] or 500 second units according to [201]), which could explain why this benchmark has not been used extensively for the assessment of computational fluid dynamics software. Therefore, they define a benchmark, similar to the traditional Taylor–Green vortex, that allows one to assess LES/DNS high-order numerical methods for turbulent flows within the context of the more complex Taylor–Couette flow.

Traditionally, the Taylor-Green vortex problem has been used as a benchmark because of its simplicity but also its ability to assess the behavior of the energy dissipation through the kinetic energy and enstrophy of the simulated flow. However, it lacks certain characteristics that the Taylor–Couette flow problem does have: curved boundaries and boundary layers. Wang & Jourdan [201] defined a Taylor–Couette benchmark at $\mathrm{Re} = 4000$ and provided DNS kinetic energy and enstrophy values for further comparison. They used a high-order flux reconstruction method, an implicit LES approach, and an explicit total variation diminishing Runge-Kutta scheme for time integration.

Following their ideas, in this work we aim to extend the benchmark by using the notion of energy balance to quantify the amount of artificial dissipation in LES; we also assess a continuous Galerkin computational fluid dynamics software (`Lethe`) that uses a stabilized formulation of the Navier–Stokes equations. Furthermore, we aim at providing further insight into the effect of the polynomial order of the finite element and the computational cost of the simulation in terms of resources and linear and nonlinear iterations, and we raise questions regarding what an acceptable amount of artificial dissipation should be in a practical LES simulation.

The article is organized as follows: In Section 7.2 we present the benchmark and discuss its key features. In Section 7.3, we present the setup of our experiments including the numerical discretization and stabilization techniques. In Section 7.4, we present the results of the simulations performed and discuss the results. Finally, we summarize our conclusions in Section 7.5.

7.2 Taylor–Couette Benchmark

The Taylor–Couette flow, as proposed by Wang & Jourdan [201], consists of the flow between two concentric cylinders, an inner cylinder with radius $r_\mathrm{i} = 0.5$ and an outer cylinder with radius $r_\mathrm{o} = 1$; the former rotates with angular velocity ω_i, while the latter is stationary. A schematic of the geometry can be seen in **Figure 7.1**. The length of the cylinders is given by $L = \pi$, the difference in the radius is equal to $d = r_\mathrm{o} - r_\mathrm{i}$, and we can define the aspect ratio as $\Gamma = L/d = 2\pi$.

7.2.1 Physics

We define the Reynolds number with respect to the inner cylinder as follows:

$$\mathrm{Re}_\mathrm{i} = \frac{U_\mathrm{i} d}{\nu} = \frac{\omega_\mathrm{i} r_\mathrm{i} d}{\nu}, \tag{7.1}$$

where ν is the kinematic viscosity and U_i is the inner-cylinder wall velocity. In addition to the Reynolds number, a common dimensionless quantity used when characterizing the Taylor–Couette flow is the Taylor number. It is defined as:

$$\mathrm{Ta} = \frac{(1+\eta)^6}{64\eta^4}\mathrm{Re}^2 = \frac{(1+\eta)^4}{64\eta^2}\frac{d^2\left(r_\mathrm{i}+r_\mathrm{o}\right)^2\omega^2}{\nu^2}, \tag{7.2}$$

where $\eta = r_\mathrm{i}/r_\mathrm{o}$ is the radius ratio of the inner and outer cylinders. The critical value is given by $\mathrm{Ta}_\mathrm{crit} \approx 1700$. The simulations that consider $\mathrm{Ta} < \mathrm{Ta}_\mathrm{crit}$ are laminar, and an analytical solution can be obtained. As soon as $\mathrm{Ta} > \mathrm{Ta}_\mathrm{crit}$ a transition to turbulence is observed and several physical phenomena arise. At low Taylor numbers ($\mathrm{Ta} \leq 10^6$), a system of coherent toroidal vortices or Taylor vortices can be observed with a circular loci center located in the axial direction. Around $\mathrm{Ta} \approx 3 \times 10^6$, the loci of the centers of the vortices become traveling waves, which also makes the flow periodic in the azimuthal direction and gives rise to complex

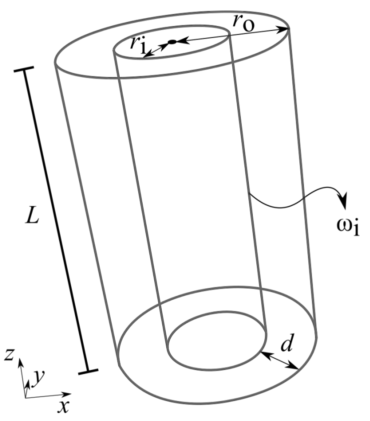

Figure 7.1 Geometry used for the Taylor–Couette flow with r_i radius of the inner cylinder, r_o radius of the outer cylinder, d the difference between the radius, L the length of the cylinders and ω_i the angular velocity of the inner cylinder.

boundary layers at the inner and outer walls of the cylinders. As the Taylor number increases further (e.g., for $\eta = 0.71$, $\text{Ta} \geq 3 \times 10^8$), the boundary layers undergo a shear instability that leads to a fully turbulent flow [188, 202].

The case considered in this work considers a Reynolds number $\text{Re}_\text{i} = 4000$ and a Taylor number of $\text{Ta} = 4.56 \times 10^7$. Wang & Jourdan [201] propose to simulate this flow for a total of 33.5 seconds. They highlighted four key moments related to the enstrophy history profile for this benchmark: 1) $t_1 = 22.058$, the moment where the first peak is encountered, 2) $t_2 = 24.016$, the moment of the first valley, 3) $t_3 = 27.2$ the moment for the second peak, and 4) $t_4 = 33.5$ the end time of the simulation. Before t_1 the enstrophy grows, indicating a strong vortex stretching near the cylinder walls and in the bulk of the flow. Strong symmetric patterns are also present, indicating laminar flow. Between t_1 and t_2, the enstrophy decreases and there is a breakdown of the vortex sheets into smaller eddies; however, the flow is still symmetric. After t_2, we start to see the transition to turbulence as the symmetry is broken and smaller eddies are obtained in the flow, even within the boundary layers, which increase the enstrophy until t_3. After this, the flow becomes more turbulent and irregular. In **Figure 7.2**, three dimensional visualizations of the vorticity magnitude for the isocontours of the Q criterion are shown. These visualizations were obtained with results from our solver and allow one to identify all the stages previously described.

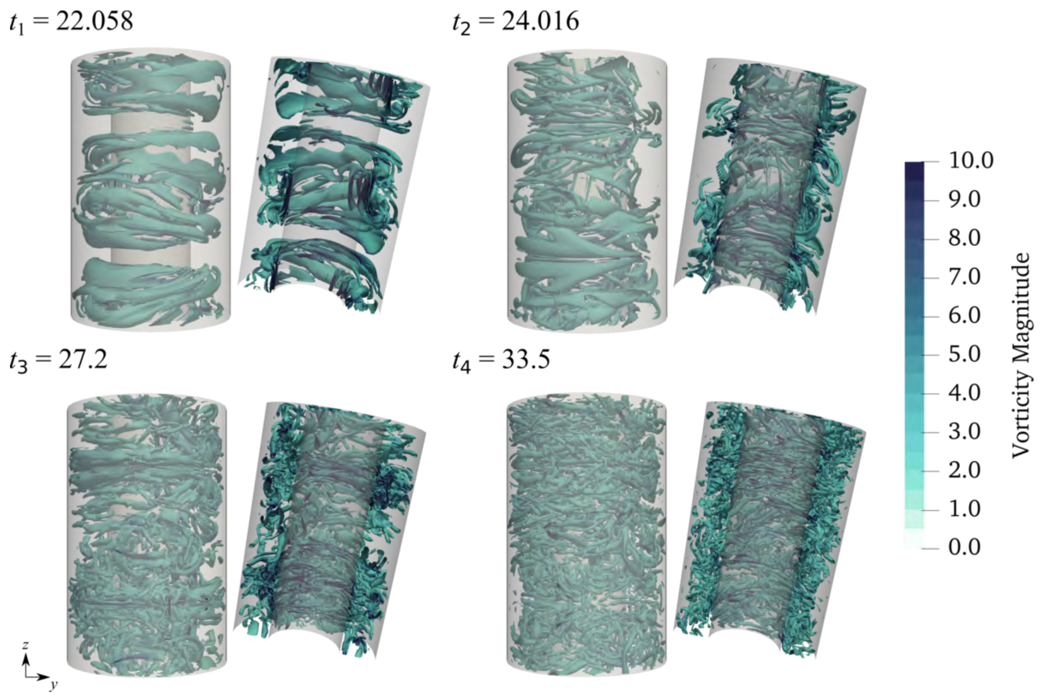

Figure 7.2 Three dimensional visualizations of the vorticity magnitude in the isocontours of the positive Q criterion values at different time steps of the Taylor–Couette simulation at Re = 4000. The visualizations are obtained using second order elements and a mesh consisiting of 942K cells.

7.2.2 Governing Equations

The Taylor–Couette flow can be described by the incompressible Navier–Stokes equations in a domain Ω with boundary $\Gamma = \partial\Omega$:

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \text{ on } \Omega \times [0,t], \\ \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} &= 0 \text{ on } \Omega \times [0,t]. \end{aligned} \tag{7.3}$$

where $\boldsymbol{u}$ is the velocity, $p^* = P/\rho$ is the reduced pressure with density ρ, and ν is the kinematic viscosity. We consider periodic boundary conditions in the axial direction z, no-slip boundary conditions on the outer cylinder wall and a fixed Dirichlet boundary condition on the inner cylinder, where we set the velocity vector to $\boldsymbol{u} = [-y, x, 0]^\top$, which is equivalent to an angular velocity of $1\,\mathrm{rad\,s^{-1}}$. The initial condition is:

$$\begin{pmatrix} u_r \\ u_\theta \\ u_z \\ p^* \end{pmatrix} = \begin{pmatrix} \epsilon U_\mathrm{i} \cos\theta \sin\left(\frac{(r-r_\mathrm{i})\pi}{r_\mathrm{i}}\right) \sin\left(\frac{z}{d}\right) \\ Ar + \frac{B}{r} + \epsilon U_\mathrm{i} \sin\theta \sin\left(\frac{(r-r_\mathrm{i})\pi}{r_\mathrm{i}}\right) \sin\left(\frac{z}{d}\right) \\ 0 \\ \frac{1}{2}A^2r^2 + 2AB\ln r - \frac{B^2}{2r^2} + \frac{1}{2}\left(\epsilon U_\mathrm{i}\right)^2 \cos 2\theta \sin\left(\frac{2(r-r_\mathrm{i})\pi}{r_\mathrm{i}}\right) \sin\left(\frac{2z}{d}\right) \end{pmatrix} \tag{7.4}$$

where ϵ is a coefficient set to 0.1, r is the radial coordinate, θ the azimuthal coordinate, and z the axial coordinate. The coefficients A and B are defined as follows:

$$A = -\frac{\omega_\mathrm{i}\eta^2}{1-\eta^2}, \quad B = \frac{\omega_\mathrm{i} r_i^2}{1-\eta^2}. \tag{7.5}$$

7.2.3 Balance of Energy

Unlike benchmarks in which homogeneous isotropic turbulence is encountered, e.g., the Taylor-Green vortex benchmark with periodic boundary conditions, for the Taylor–Couette flow the kinetic energy rate cannot be directly compared to the enstrophy since other relevant terms come into play [21, 34]. In order to understand all of these terms, one can study the rate of change of the kinetic energy in the system. The kinetic energy rate equation can be obtained as the inner product of the velocity $\boldsymbol{u}$ and the equation of motion for an incompressible flow as follows [202]:

$$\begin{aligned} \int_\Omega \frac{\partial}{\partial t}\left(\frac{1}{2}\rho u^2\right) \mathrm{d}\Omega = &- \int_\Omega \left(\nabla \cdot \frac{1}{2}\rho u^2 \boldsymbol{u}\right) \mathrm{d}\Omega \\ &- \int_\Omega \left(\nabla \cdot P\boldsymbol{u}\right) \mathrm{d}\Omega - \int_\Omega P\left(-\nabla \cdot \boldsymbol{u}\right) \mathrm{d}\Omega \\ &- \int_\Omega \left(\nabla \cdot [\boldsymbol{\tau} \cdot \boldsymbol{u}]\right) \mathrm{d}\Omega - \int_\Omega \left(-\boldsymbol{\tau} : \nabla \boldsymbol{u}\right) \mathrm{d}\Omega \end{aligned} \tag{7.6}$$

where u^2 is the square of the magnitude of the velocity vector $\boldsymbol{u}$, P is the pressure and $\boldsymbol{\tau} = -\mu\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^\top\right)$ is the deviatoric stress tensor with dynamic viscosity μ. The first term on the right-hand side represents the rate at which the kinetic energy is convected across the domain; since we have periodic boundary conditions, it is equal to zero for this problem. The divergence theorem can be applied to the second term on the right-hand side:

$$-\int_\Omega \nabla \cdot (P\boldsymbol{u})\,\mathrm{d}\Omega = = -\int_\Gamma \boldsymbol{n} \cdot [P\boldsymbol{u}]\,\mathrm{d}\Gamma, \tag{7.7}$$

and to the fourth term as well:

$$-\int_{\Omega} (\nabla \cdot [\boldsymbol{\tau} \cdot \boldsymbol{u}])\,\mathrm{d}\Omega = -\int_{\Gamma} \boldsymbol{n} \cdot [\boldsymbol{\tau} \cdot \boldsymbol{u}]\,\mathrm{d}\Gamma, \tag{7.8}$$

to obtain:

$$\begin{aligned}\int_{\Omega} \frac{\partial}{\partial t}\left(\frac{1}{2}\rho u^2\right)\mathrm{d}\Omega =& \cancelto{0}{-\int_{\Omega}\left(\nabla \cdot \frac{1}{2}\rho u^2 \boldsymbol{u}\right)\mathrm{d}\Omega} \\ &- \int_{\Gamma} \boldsymbol{n} \cdot [P\boldsymbol{u}]\,\mathrm{d}\Gamma - \cancelto{0}{\int_{\Omega} P\,(-\nabla \cdot \boldsymbol{u})\,\mathrm{d}\Omega} \\ &- \int_{\Gamma} \boldsymbol{n} \cdot [\boldsymbol{\tau} \cdot \boldsymbol{u}]\,\mathrm{d}\Gamma - \int_{\Omega} (-\boldsymbol{\tau} : \nabla \boldsymbol{u})\,\mathrm{d}\Omega.\end{aligned} \tag{7.9}$$

The final balance is then given by the following equation:

$$\int_{\Omega} \frac{\partial}{\partial t}\left(\frac{1}{2}\rho u^2\right)\mathrm{d}\Omega = -\int_{\Gamma} (\boldsymbol{n} \cdot [\boldsymbol{\tau} \cdot \boldsymbol{u}])\,\mathrm{d}\Gamma - \int_{\Omega} (-\boldsymbol{\tau} : \nabla \boldsymbol{u})\,\mathrm{d}\Omega \tag{7.10}$$

The first term corresponds to the power applied to the wall of the inner cylinder calculated through the deviatoric stress tensor $\boldsymbol{\tau}$ contracted with the velocity at the wall, while the second term corresponds to the viscous dissipation within the domain. Following the idea first presented by Schranner et al. [203] and then used also in other works, e.g., [204], and considering an incompressible flow with $\rho = 1$, we rewrite the balance of energy in terms of kinetic energy divided by density as follows:

$$-\epsilon = \int_{\Omega} \frac{\partial}{\partial t}\left(\frac{1}{2}u^2\right)\mathrm{d}\Omega + \int_{\Gamma} (\boldsymbol{n} \cdot [\boldsymbol{\tau}^* \cdot \boldsymbol{u}])\,\mathrm{d}\Gamma + \int_{\Omega} (-\boldsymbol{\tau}^* : \nabla \boldsymbol{u})\,\mathrm{d}\Omega \tag{7.11}$$

where the residual ϵ is a positive value that corresponds to the source of numerical dissipation that causes the kinetic energy to be lost over time, and $\tau^* = -\nu\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\top\right)$. The individual values of the three terms: kinetic energy time derivative, the power input due to the torque on the inner cylinder, and the viscous dissipation, can be obtained as post-processed quantities of the simulation. The value of the numerical dissipation is normalized with respect to the maximum value of the viscous dissipation, to finally obtain a dimensionless quantity ϵ_{n}. Studying this value allows us to quantify the numerical dissipation of the implicit LES approach used in this work and allows us to have a specific metric that can be used to compare different simulation cases for the Taylor–Couette benchmark.

7.3 Setup of the Simulation

In this section, the CFD solver used in this study, along with the mesh and the methodology are presented.

7.3.1 Stabilized Continuous Finite Element Solver

The simulations are carried out using `Lethe` [105, 106], an open source computational fluid dynamics software. This software is based on the `deal.II` finite element library [82], which in turn uses `p4est` for mesh support [107] and MPI for parallelization. This publication uses the matrix-free fluid solver of `Lethe` [205], which has been proven to scale well for both steady-state and transient challenging problems at high Reynolds number, including the Taylor-Green vortex benchmark. The software solves the incompressible Navier–Stokes equations in the domain Ω as presented in Equation 7.3 for velocity and pressure in a coupled way using a continuous Galerkin approach. It also takes into account the Streamline-Upwind/Petrov-Galerkin **(SUPG)** and the Pressure-Stabilizing/ Petrov-Galerkin (**(PSPG)**) stabilization techniques, allowing to use the same element order for the velocity and the pressure and avoiding oscillations in the solution. The stabilization terms are added to the weak formulation of equation 7.3 and are defined as follows:

$$\begin{aligned}
&\underbrace{(q, \nabla \cdot \boldsymbol{u})_\Omega + (\boldsymbol{v}, \partial_t \boldsymbol{u})_\Omega + (\boldsymbol{v}, (\boldsymbol{u} \cdot \nabla)\boldsymbol{u})_\Omega - (\boldsymbol{v}, \boldsymbol{f})_\Omega + \nu(\nabla \boldsymbol{v}, \nabla \boldsymbol{u})_\Omega - (\nabla \cdot \boldsymbol{v}, p^*)_\Omega}_{\text{Weak formulation}} \\
&+ \underbrace{\sum_k (\tau \nabla q, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} - \boldsymbol{f})_{\Omega_k}}_{\text{PSPG term}} \\
&+ \underbrace{\sum_k (\tau (\boldsymbol{u} \cdot \nabla)\boldsymbol{v}, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} - \boldsymbol{f})_{\Omega_k}}_{\text{SUPG term}} = 0.
\end{aligned} \tag{7.12}$$

The terms consider the residual of the momentum equation, multiplied by a term that considers a stabilization parameter τ and the test functions ($\boldsymbol{v}$ or q). For τ, we extend the definition of Tezduyar et al. [149] to obtain a stabilization parameter that includes the polynomial order of the finite element p:

$$\tau = \left[\left(\frac{1}{\Delta t} \right)^2 + \left(\frac{2\|\boldsymbol{u}\| p}{h_{\text{conv}}} \right)^2 + 9 \left(\frac{4\nu p^2}{h_{\text{diff}}^2} \right)^2 \right]^{-1/2}, \tag{7.13}$$

where $h_{\text{conv}} = h_{\text{diff}} = h$ is the element size, which in 3D is equal to $h = (6h_k/\pi)^{1/3}$, where

h_k is the volume of a cell. For the time discretization, we use a second-order backward differentiation method which is A-stable [76] and use a constant CFL number of 1, where the CFL is defined as CFL $= (\|\boldsymbol{u}\|\Delta t/h)p$. We use a Newton-Krylov approach to solve the non-linear problem and use a GMRES method with a monolithic geometric multigrid as a preconditioner for the linear system. A detailed description of the matrix-free solver implementation can be found in [205]; more details on the geometric multigrid preconditioner can be found in [112].

7.3.2 Mesh

The mesh that we use in this work is uniform in the axial and azimuthal directions and includes one local refinement in the radial direction next to the wall boundaries of the cylinder. We use hexahedral-shaped elements to discretize the domain and we attach a high-order cylindrical manifold that allows an accurate description of the curved boundaries using isoparametric Q elements. The smallest mesh considered for the experiments has 154K cells, while the largest has 6M cells. **Figure 7.3** shows a sample mesh with 5670 cells and two different views highlighting the local refinement next to the walls. For more information on how hanging nodes are managed, we refer the reader to the procedure introduced by Bangerth & Kayser-Herold [206]. In the original benchmark, cells are clustered in the radial direction by specifying a growth ratio of 1.5 near the wall; in our case, one refinement next to the walls was enough to obtain satisfactory results (see Appendix A in Section 7.7 for the comparison of the results using two meshes: one with one refinement next to the walls and the other one with two refinements).

7.3.3 Methodology

Following the procedure specified by Wang & Jourdan [201] and the extensions proposed in this work, the benchmark consists of the following:

1. Use the initial condition presented in equation 7.4 and run the experiments until $t = 33.5\,\mathrm{s}$.

2. Compare the enstrophy and kinetic energy history with the reference DNS result ($p = 5$, 102K cells and 110M DoFs) provided by Wang & Jourdan [201].

3. Compare the distribution of the vorticity magnitude ($\nabla \times \boldsymbol{u}$) and the Q criterion at $t = 24.016$ among the different experiments and with the reference DNS results.

4. Compare the history of numerical dissipation ϵ_n among the different experiments.

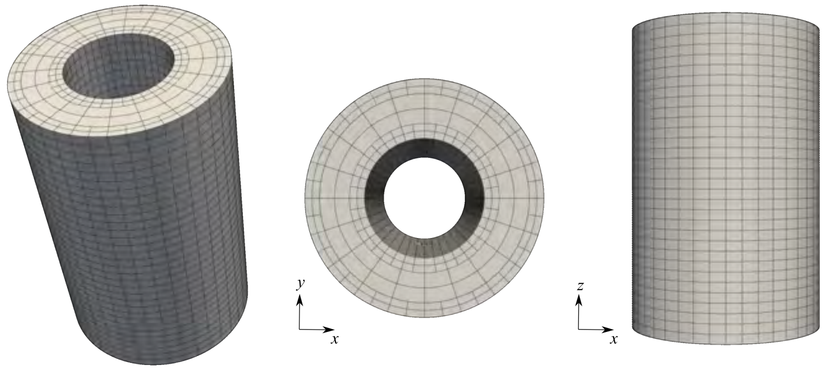

Figure 7.3 Sample of the Taylor–Couette mesh used for the simulations. Note the additional refinement next to the walls.

5. Evaluate performance in terms of computational time as well as linear and non-linear iterations.

All tests were run on the Niagara distributed memory cluster from the Digital Research Alliance of Canada. Niagara consists of 2024 nodes, each with 40 Intel Skylake cores (2.4 GHz) with 202 GB of RAM per node. We use AVX512 instructions for vectorization over 8 cells/doubles (512 bits). We use between 4 and 32 nodes, depending on memory requirements.

7.4 Results and Discussion

We consider nine experiments with different number of global refinements, i.e., number of cells, and element orders Q_pQ_p, where p is the polynomial degree of the underlying element. A summary of all the cases considered is presented in **Table 7.1**.

Table 7.1 Summary of the number of cells and number of degrees of freedom (DoFs) for all the simulation cases considered for the Taylor–Couette flow with Re = 4000.

Global refinement	**No. Cells**	**No. DoFs**		
$\boldsymbol{\ell}$		$\boldsymbol{Q_1Q_1}$	$\boldsymbol{Q_2Q_2}$	$\boldsymbol{Q_3Q_3}$
4	154K	765K	5M	18M
5	942K	4M	33M	108M
6	6M	27M	215M	716M

7.4.1 History of Enstrophy and Kinetic Energy

The results of the enstrophy profiles are presented in **Figure 7.4**. We observe the ability of the solver to achieve a more accurate enstrophy with increasing order of the elements p and refinement of the mesh. For the Q_1Q_1 cases, the results approach the reference data as we increase the mesh resolution. Taking a look at the first peak, which occurs at $t_1 = 22.058$, we observe a delayed dissipation for the coarsest mesh due to the inability to capture small eddies. For the Q_2Q_2 elements, the results for the coarse mesh are considerably improved and the two finer meshes obtain very similar results to those obtained by Wang & Jourdan [201]. As we further increase the order of the elements to Q_3Q_3, even the case with the coarsest mesh gives good results when compared to the reference results; all the meshes capture the first and the second peak. There is a difference observed for the finest Q_3Q_3 simulation after the second enstrophy peak; however, note that after $t_3 = 27.2$, no similarity between results is expected in the history of enstrophy as the flow quickly becomes fully turbulent.

The results for the kinetic energy are presented in **Figure 7.5**. All cases converge to the reference solution, with the coarsest Q_1Q_1 being the only exception. A similar result was obtained by Wang & Jourdan [201] for their coarsest simulation ($p = 2$ with 2.8M cells and a total number of 13.8M DoFs). We can conclude that while the kinetic energy is a useful quantity, enstrophy encounters a more significant change in terms of order of magnitude during the 33.5 second interval. This makes enstrophy a better quantity that can be used to compare results of different simulations computed using different solvers.

7.4.2 Vorticity and Q Criterion Distributions

To complete the benchmark as proposed by Wang & Jourdan [201], we compare in **Figure 7.6** the vorticity profiles obtained for a zy-plane at $t = t_2$ for three different simulations (the finest Q_1Q_1 simulation, the medium Q_2Q_2 simulation, and the coarsest Q_3Q_3 simulation) against the finest Q_3Q_3 simulation, which is considered the one with the highest resolution in this work. One can observe strong symmetric patterns throughout the domain and higher vorticity values near the inner wall for all simulations, as described previously in the physics section. Moreover, one can observe large structures that are captured by all the simulations and showcase low vorticity values. Focusing on the highest values of the vorticity, one can also identify smaller vortices both near to the inner wall and in the bulk of the flow.

In addition to the vorticity, we also present in **Figure 7.7** the Q criterion (as defined in [21]) and present it for the same plane and the same set of simulations. This allows us to focus solely on the vortices with higher vorticity, that is, the smallest structures captured by

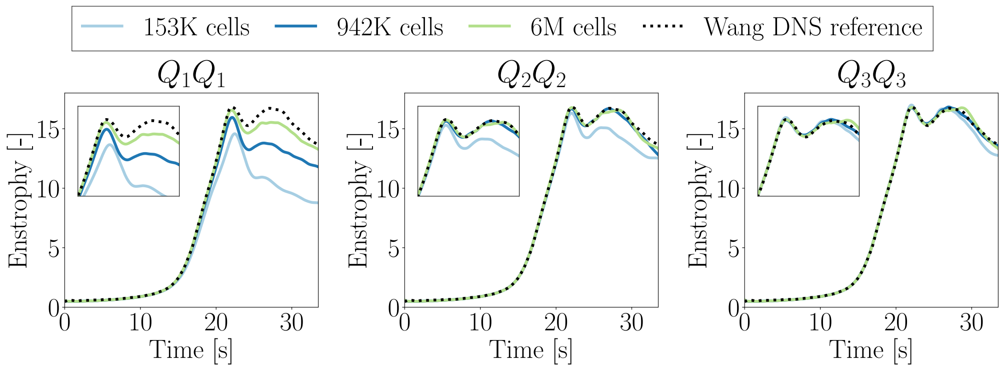

Figure 7.4 Enstrophy over time for the Taylor–Couette flow at Re = 4000 using different cases with Q_pQ_p elements with $p = 1, 2$ and 3, and different mesh sizes.

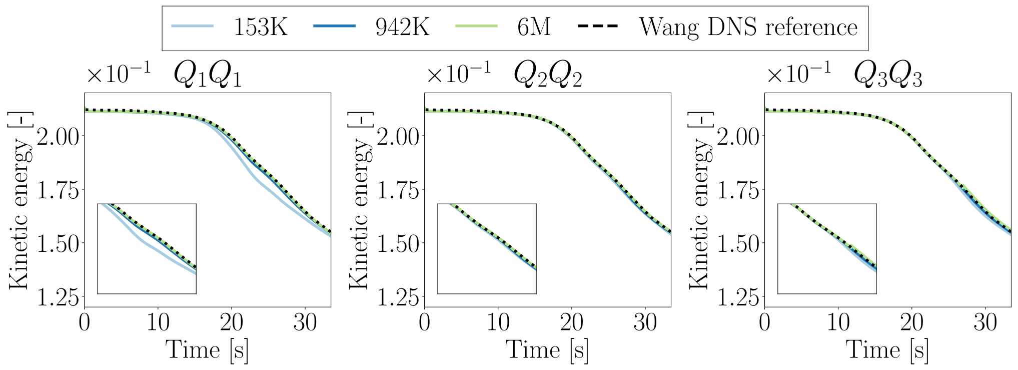

Figure 7.5 Kinetic energy over time for the Taylor–Couette flow at Re = 4000 using different cases with Q_pQ_p elements with $p = 1, 2$ and 3, and different mesh sizes.

the different simulations. To quickly see the differences among them, we have highlighted three different regions in the figure around $z = 2.4$. For the *region 1*, the Q_1Q_1 simulation manages to obtain a similar number of vortices placed at the same locations as the finest Q_3Q_3 simulation, while the other two simulations seem to have less vortices at different locations. In the case of the *region 2*, the Q_1Q_1 simulation does not resolve very small vortices near to the rolls, on the other hand, the other two simulations seem to better capture those vortices. Finally, in the *region 3*, one can observe that the Q_1Q_1 simulation fails to capture the smaller vortices, while the other two manage to resolve them, with the Q_2Q_2 simulation being the one that most resembles the results of the finest Q_3Q_3 simulation.

Although these two-dimensional profiles are indeed useful in identifying the turbulent structures, we do not believe that they are appropriate to compare the influence of the order of the element directly. The reason for this is that high-order rendering is still a challenge

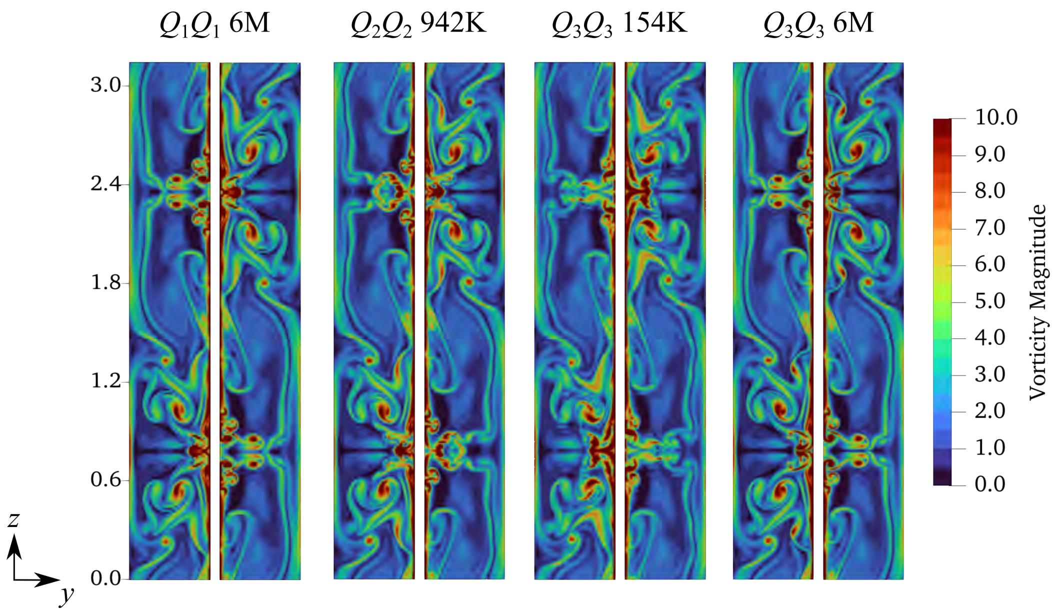

Figure 7.6 Comparison of the vorticity profile for four different simulations defined by polynomial order and total number of cells for the Taylor–Couette flow at Re $= 4000$ at $t_2 = 24.016\,$s in a zy-plane.

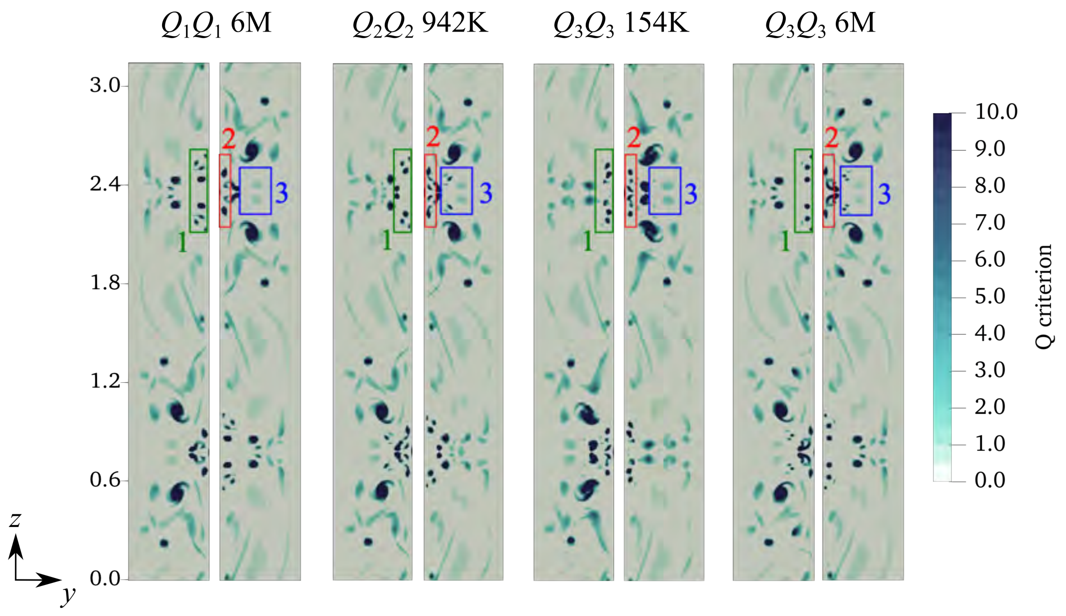

Figure 7.7 Comparison of the Q criterion profile for four different simulations defined by polynomial order and total number of cells for the Taylor–Couette flow at Re $= 4000$ at $t_2 = 24.016\,$s in a zy-plane. There are three highlighted regions at around $z = 2.4$ for comparison purposes.

and even though we use the high-order capabilities provided by `Paraview` and available in `deal.II`, there are still many questions on whether this rendering is accurate, especially when extracting planes from a three-dimensional object. Due to this and in order to provide a more rigorous comparison, we decided to analyze the simulations closely using the energy balance presented in Section 7.3.

7.4.3 Numerical Dissipation

The normalized numerical dissipation ϵ_n of each of the experiments is plotted over time in **Figure 7.8**. As expected, the numerical dissipation decreases as we refine the mesh for all the element orders. From all simulations, the medium and fine simulations for Q_3Q_3 elements obtain the smallest dissipation of all cases. We can see that the difference between the dissipation between the cases that use the same order varies considerably, with Q_1Q_1 having the largest difference. It is important to note that all simulations, except the coarsest ones for Q_1Q_1 and Q_2Q_2, obtain some negative values of the order of 10^{-3} for the numerical dissipation that cannot be clearly seen in Figure 7.8. These small negative values are considered to be noise attributed to the error of the method used to calculate numerical dissipation during the post-processing stages. Similar conclusions were drawn by Cadieux et al. [204] who used the same methodology to quantify the numerical dissipation for turbulent separation bubble flow. Moreover, they demonstrated that the method to quantify the numerical dissipation did not have a large influence on the results, as long as it was of the same order as the solver that produced the solution.

The superposed enstrophy (dotted curve in Figure 7.8) allows us to analyze the moment where the energy starts being dissipated by the different schemes and its relation to the physics encountered at the key times of the enstrophy profile. For the Q_1Q_1 simulation with the coarse mesh the dissipation starts increasing at ∼10 s, while it starts at ∼15 s for Q_2Q_2 and at ∼18 s for Q_3Q_3 for the coarser mesh. This mismatch is a direct representation of the role of the order of the element and how dissipation occurs earlier for lower order elements even when keeping the same number of DoFs (see fine Q_1Q_1 case with 27M DoFs and compare it to medium Q_2Q_2 case with 33M DoFs). This energy balance analysis also allows us to see that even though in the previous profiles the finest simulation using Q_1Q_1 looked very similar to the finest simulation using Q_3Q_3, the energy being dissipated varies significantly, explaining the differences in the smaller vortices observed in the two dimensional profiles.

The results show that the numerical dissipation reduces significantly when using higher-order elements. The question that remains is whether it is worth it to use high-order elements for this benchmark in terms of computational resources. To answer this question we decided to

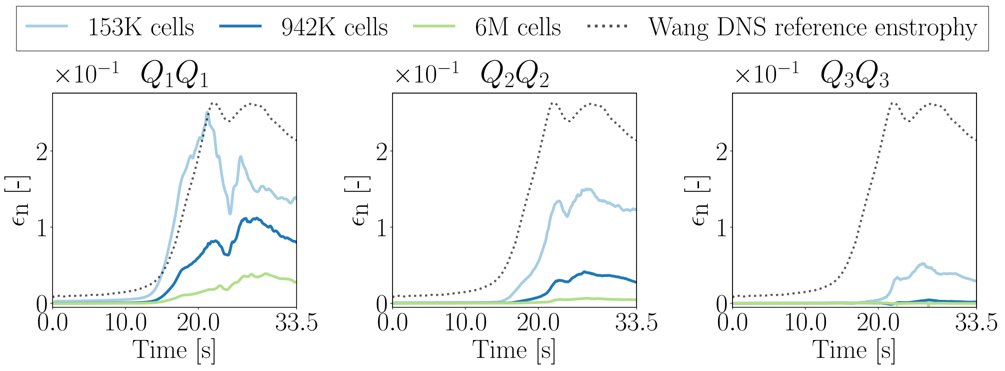

Figure 7.8 Normalized numerical dissipation against time for the Taylor–Couette flow at Re = 4000 for all the cases considered. The reference enstrophy is rescaled to the same order of magnitude and shown in the same plot for analysis purposes.

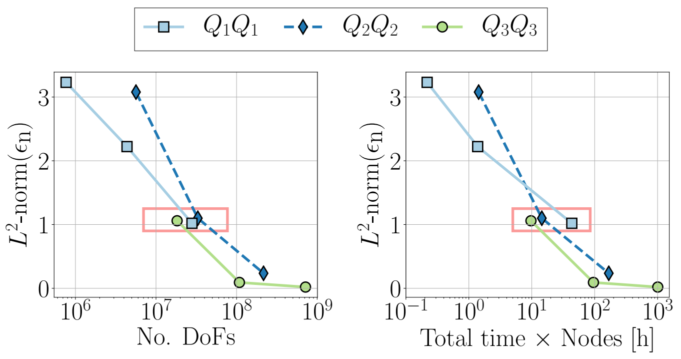

Figure 7.9 L^2-norm of the energy balance against the number of degrees of freedom and the computational time for all the simulation cases. Results for a L^2-norm of 1 highlighted within a rectangle.

look at the L^2-norm of the energy balance and plot it against the cost (in terms of number of degrees of freedom and computational time) for all simulation cases, as shown in **Figure 7.9**. In this case, the smaller the value, the smaller the included artificial dissipation. The results show that the Q_1Q_1 simulations never reach the minimal L^2 norm achieved by the higher-order simulations. On the other hand, if we use the finest mesh with Q_3Q_3 elements to get the smallest numerical dissipation, it becomes quickly more expensive than all the other cases. A case in between would be the case where the L^2 norm is approximately one (see rectangles in Figure 7.9), where the Q_3Q_3 simulation with the coarsest mesh is cheaper in terms of DoFs and time multiplied by the number of nodes than the Q_1Q_1 with the finest mesh and the Q_2Q_2 with the medium mesh. If we recall the enstrophy profiles (see Figure 7.4), the coarsest Q_3Q_3 case already obtained an enstrophy profile close to the reference results by Wang &

Jourdan [201].

7.4.4 Computational Time and Iteration Counts

For the sake of completeness, we present three additional metrics that can give some hints on performance: the number of transient iterations, the number of Newton iterations needed for the nonlinear solver, and the number of linear iterations needed for the GMRES solver preconditioned by a geometric multigrid preconditioner. Each of them is reported for all the cases considered in **Table 7.2**. The results show that because of the constant CFL condition, the transient iterations increase as the mesh is refined. In terms of non-linear iterations, all simulation cases perform one Newton iteration per transient iteration, with the exception of the coarse cases for Q_1Q_1 and Q_2Q_2, which need 1.5 and 1.4 iterations, respectively. Moreover, the linear iterations per Newton step decrease as the mesh is further refined. The results are analyzed taking into account that the multigrid preconditioner is setup at every transient iteration and it is the most expensive part of the algorithm ($\sim 50\%$ of the total time). With this in mind, one possibility to reduce computational time is to reduce the setup costs of the preconditioner by spending more time solving the problem, i.e., more Newton steps per transient iteration and possibly more linear iterations per Newton step. To achieve this, one could consider to use a higher CFL number.

Increasing the CFL is something that is not commonly done in practice. There are several reasons for this, including the stability constraints of time-stepping schemes and the effect on actual physical results. However, since we use an unconditionally stable implicit time-stepping scheme, we can increase the CFL to understand its effect on physics and computational performance. For this, we chose the medium simulation case for Q_2Q_2 elements

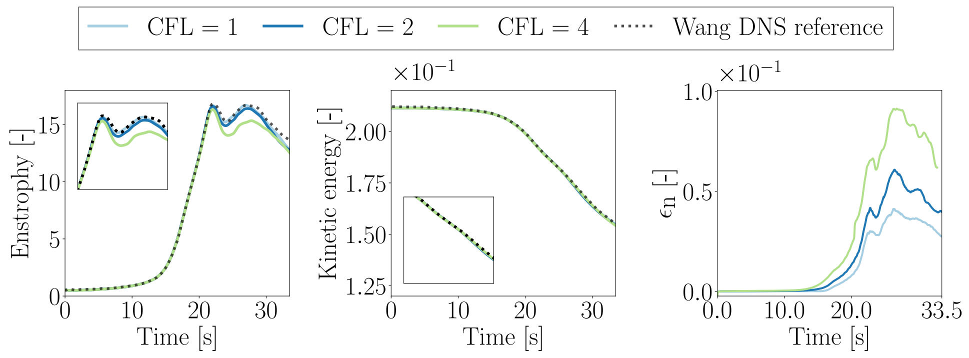

Figure 7.10 Comparison for the case with 942K cells for Q_2Q_2 when run with different CFL numbers.

Table 7.2 Summary of the transient iterations (N_T), average Newton iterations ($\overline{N}_N$) per transient iteration, average linear iterations per Newton iteration ($\overline{N}_L$) and total time multiplied by the number of nodes (T), needed to solve the different cases of the Taylor–Couette flow at Re = 4000.

	Q_1Q_1				Q_2Q_2				Q_3Q_3			
No. Cells	N_T	$\overline{N}_N$	$\overline{N}_L$	T [h]	N_T	$\overline{N}_N$	$\overline{N}_L$	T [h]	N_T	$\overline{N}_N$	$\overline{N}_L$	T [h]
154K	704	1.5	2.6	0.2	1418	1.4	3.0	1.4	2126	1.0	3.7	9.7
942K	1412	1.0	2.2	1.4	2852	1.0	2.5	14.5	4295	1.0	2.6	95.6
6M	2854	1.0	1.9	15.6	5693	1.0	1.9	167.6	7686	1.0	1.8	1007.1

and ran it two additional times using a CFL of 2 and 4. The results can be seen in **Figure 7.10**, in terms of enstrophy and kinetic energy history, as well as energy balance. For the kinetic energy, all the simulations obtain the same result. In the case of the enstrophy, all of the simulations have a similar dissipation until the first peak; after that point, the simulation with the highest CFL differs from the reference solution. Although enstrophy does not show significant difference for the simulations with CFL equal to 1 and 2, in the energy balance plot, we can see a difference in artificial dissipation.

These results are even more interesting if analyzed conjunctly with the number of iterations and the time multiplied by the number of nodes needed for the simulations. This information is shown in **Table 7.3**. It is shown that if the CFL is doubled, the number of linear iterations is increased by 50%, and this leads to a reduction of computing time per node from 14.5 to 9.3 hours (the time needed to setup the preconditioner is reduced by 45%). This is further reduced when the CFL is quadrupled (the time needed to setup the preconditioner is reduced by 72%), but as we saw, the influence of this change already affects the enstrophy results, leading to considerable differences obtained with the other two fixed CFL numbers. To conclude this part, doubling the CFL number would lead to a reduction in the computational time needed by our solver for this benchmark without greatly influencing the physical results. This analysis raises the question of whether we should look at more efficient time-stepping schemes, like singly diagonally implicit Runge-Kutta methods (SDIRK) to further improve this aspect in our stabilized solver. This has been suggested before by John [30] for higher-order schemes and it is an active field of research, e.g., see [207].

Table 7.3 Summary of the transient iterations (N_T), average Newton iterations ($\overline{N}_N$) per transient iteration, average linear iterations per Newton iteration ($\overline{N}_L$) and total time multiplied by the number of nodes (T), needed to solve the 942K case with Q_2Q_2 case for the Taylor–Couette flow at Re = 4000.

	$\boldsymbol{Q_2Q_2}$			
CFL	N_T	$\overline{N}_N$	$\overline{N}_L$	T[h]
1	2852	1.0	2.5	14.5
2	1423	1.2	3.8	9.3
4	716	1.4	6.3	5.8

7.5 Conclusions and Further Research

In this work, we use the Taylor–Couette benchmark as recently proposed by Wang & Jourdan [201] to assess a stabilized continuous finite element solver. We consider nine experiments with three different meshes and finite element orders Q_pQ_p with $p = 1, 2, 3$.

These experiments allow to compare enstrophy and kinetic energy histories with reference results. The results show that the enstrophy history prediction is more accurate as the order of the element is increased and as the mesh is more refined. The kinetic energy is well predicted for all cases, with exception of the coarsest Q_1Q_1 mesh. While the vorticity and Q criterion profiles allow one to identify large and small structures in a two-dimensional slice of the geometry, they do not allow for a quantitative comparison; therefore, we extend the benchmark by analyzing the numerical dissipation included by the ILES approach.

The numerical dissipation is computed via the energy balance of the problem following the ideas first presented by Schranner et al. [203]. The results show that the dissipation decreases as the order of the element increases and the mesh is refined. This quantity also allows us to have a metric to compare the computational cost of the simulation. The results are highly dependent on the desired numerical dissipation; however, they show that the Q_3Q_3 case with the coarsest mesh is promising, as it achieves a similar numerical dissipation as the finest Q_1Q_1 case and the medium Q_2Q_2 case, with a smaller number of degrees of freedom and computational time per node.

The extension presented in this work not only computes the numerical dissipation for an implicit LES approach but provides more data for future studies that wish to assess other solvers and their numerical dissipation. Moreover, it proposes a more rigorous comparison in the case of higher order solvers that does not rely only on two-dimensional profiles of the vorticity but on a quantitative metric. Open questions regarding the improvement of the performance of the solver, suggest that future work on time-stepping schemes could be performed. Moreover, this work raises the question as to how much numerical dissipation is an acceptable dissipation. We think that further investigation could be done to find the link between energy dissipation and first- and second-order statistics to be able to draw conclusions for practical engineering applications.

7.6 Acknowledgements

The authors acknowledge collaboration with the `deal.II` community. The authors also acknowledge technical support and computing time provided by the Digital Research Alliance of Canada and Calcul Québec.

7.7 Appendix A : Test with Different Refinements next to the Wall

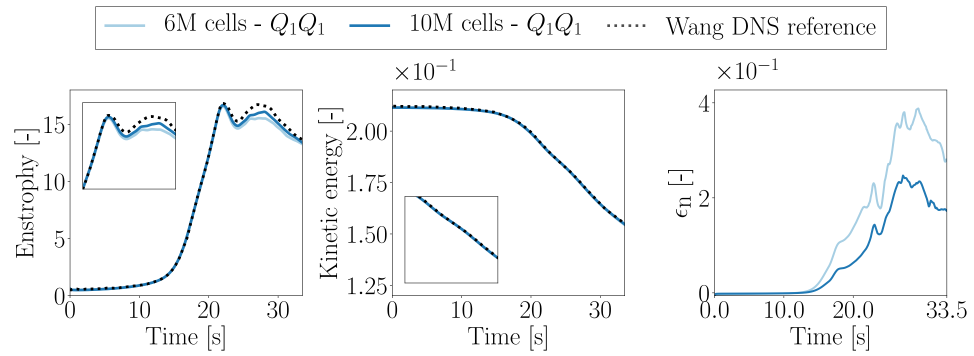

Figure 7.11 Comparison of the enstrophy, the kinetic energy and the balance of energy for the Q_1Q_1 simulation when one additional refinement next to the walls is performed.

The boundary layer plays an important role in this benchmark. The original benchmark as proposed by [201] had a mesh with a growth ratio in the radial direction. In contrast, we used a uniform mesh with one local refinement next to the walls of the inner and outer cylinders. This choice was motivated after running several experiments with different number of refinements next to the wall. In this appendix, we show one of these experiments that considers Q_1Q_1 elements with two different meshes: a mesh with one refinement next to the walls (6M cells) and a mesh with two refinements next to the walls (10M cells). The results for enstrophy, kinetic energy, and artificial dissipation are presented in **Figure 7.11**. As can be seen, the results do not vary considerably; however, they do affect the computational cost of the simulation due to the increase in number of cells. The simulation with 6M cells requires $\sim$15.6 hours to run, while the simulation with 10M cells requires $\sim$50.35 hours to run. Due to this, we decided to use only one refinement next to the walls in the experiments presented in Section 7.4.

CHAPTER 8 LOCAL MESH REFINEMENT AND GEOMETRIC MULTIGRID METHODS FOR FLOW PROBLEMS

In this chapter, we compare the two variants of matrix-free monolithic geometric multigrid preconditioners for locally-refined meshes introduced in Section 4.2.2: Local Smoothing (LS) and Global Coarsening (GC). Part of the material in this chapter was presented at the WCCM/PANACM Conference in Vancouver in July 2024 and the `deal.II` users and developers workshop in Fort Collins in August 2024. These two GMG variants were already extensively compared in a publication by Munch et al. [112] for Poisson problems and a Stokes flow problem with Taylor-Hood elements (using a block preconditioner). The performance of both multigrid approaches was evaluated using different metrics, and the main conclusions were as follows:

- For serial simulations, LS is faster than GC due to two reasons: the total number of cells to perform smoothing on is lower, and there is no cost for evaluating hanging node constraints.
- For parallel simulations, GC is faster than LS due to better load balance.
- For both serial and parallel simulations, GC needs the same amount or a lower number of iterations than LS.

In this chapter, the aim is to evaluate the behavior of these two multigrid approaches in the context of a monolithic stabilized Navier–Stokes solver and for two benchmarks: the transient turbulent Taylor-Couette flow with local static mesh refinement and the steady-state flow around a sphere with adaptive mesh refinement. The comparison is made in terms of the number of iterations, time to solution, time per iteration spent at each multigrid level and time for each component of the multigrid algorithm: pre-smoothing, residual computation, restriction, coarse grid solve, prolongation, edge correction (for LS only) and postsmoothing. A summary of the solver parameters used for each case is presented in **Table 8.1**.

8.1 Turbulent Taylor-Couette: Local Static Mesh Refinement

In this section, the Taylor–Couette flow benchmark, studied extensively in Chapter 7, is used to study the serial and large-scale parallel behavior of the two GMG preconditioners. The cases considered with an increasing number of refinements ℓ, the corresponding number of cells and DoFs are presented in **Table 8.2**.

Table 8.1 Summary of the solver parameters used for the local mesh refinement tests. It includes the parameters of the Newton non-linear solver and the GMRES linear solver preconditioned by geometric multigrid (GMG).

Component	**Parameter**	**Taylor–Couette**	**Sphere**
Newton solver	Absolute tolerance	10^{-5}	10^{-4}
GMRES linear solver	Relative residual	10^{-4}	10^{-4}
	Absolute residual	10^{-7}	10^{-5}
GMG preconditioner	**Smoother**		
	Inverse diagonal (ID)		
	Number of iterations	5	5
	Coarse-grid solver		
	Direct solver	✓	–
	GMRES preconditioned by ILU		
	Relative residual	–	10^{-2}

Table 8.2 Summary of the refinement level, number of cells and number of degrees of freedom (DoFs) for all the simulation cases considered for the Taylor–Couette problem with one local refinement next to the wall. The cases that are empty were not considered.

Refinement ℓ	**No. Cells**	**No. DoFs**		
		Q_1Q_1	Q_2Q_2	Q_3Q_3
2	5K	33K	231K	731K
3	28K		1M	3.4M
4	154K	765K	5M	18M
5	942K	4M	33M	
6	6M	27M		

8.1.1 Serial Run

Serial simulations are carried out only for cases with $\ell = 2$ and Q_1Q_1 and Q_2Q_2 elements. A summary of the iterations needed for the non-linear and linear solver is presented in **Table 8.3**. The results show that for Q_1Q_1 elements, the same amount of transient iterations and Newton iterations is performed for both multigrid variants. In terms of linear iterations, LS requires more linear iterations; however, each linear iteration is cheaper in terms of time than in the case of GC.

To investigate the difference in costs between both algorithms closely, **Figure 8.1** shows the

Table 8.3 Summary of transient iterations (N_T), average Newton iterations ($\overline{N}_N$) per transient iteration, average linear solver iterations per Newton iteration ($\overline{N}_L$), and total time needed to solve the Taylor–Couette flow at Re = 4000 cases with $\ell = 2$ with a single process using the global coarsening (GC) and local smoothing (LS) approaches.

	$\boldsymbol{Q_1Q_1}$								$\boldsymbol{Q_2Q_2}$							
	GC				**LS**				**GC**				**LS**			
$\boldsymbol{\ell}$	N_T	$\overline{N}_N$	$\overline{N}_L$	T [s]	N_T	$\overline{N}_N$	$\overline{N}_L$	T [s]	N_T	$\overline{N}_N$	$\overline{N}_L$	T [s]	N_T	$\overline{N}_N$	$\overline{N}_L$	T [s]
2	74	2.5	10.3	68.3	74	2.5	15.4	83.4	150	2.7	10.5	579	150	2.7	12.2	598

time per linear iteration for each of the multigrid levels for the Q_1Q_1 case. The results show that most of the time is spent at the finest level, while the coarse grid level (level 0) is cheap, since it only has 20 cells. A similar trend is observed for the Q_2Q_2 case (not shown here). For Q_1Q_1, a GC V-cycle is 22% times slower than a LS V-cycle. This makes sense since there are around 12% more cells in the case of global coarsening and the application hanging node constraints is more expensive. Similar conclusions were obtained in [112]. For Q_2Q_2 elements, the difference is larger with a GC V-cycle being 13% slower than a LS V-cycle (not shown).

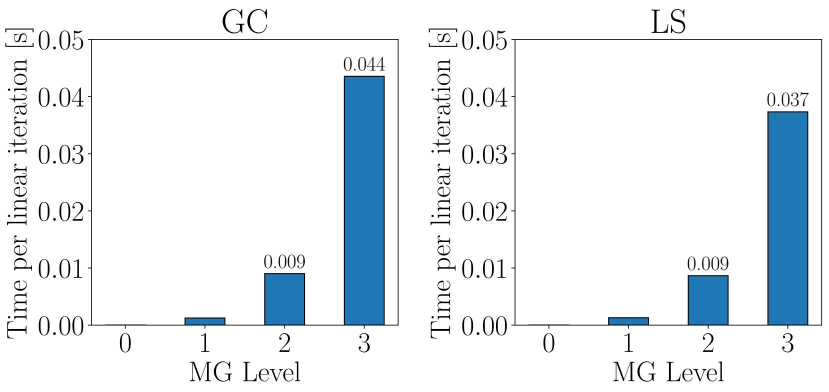

Figure 8.1 Cost of one V-cycle for the Taylor-Couette simulation with a single process for Q_1Q_1 elements using the global coarsening (GC) and the local smoothing (LS) approaches.

The times for each of the components of the multigrid algorithm for the Q_1Q_1 case are shown in **Figure 8.2**. As expected, the pre- and post-smoothing components are the most expensive steps of the algorithm. In addition to this, the results show that the cost of applying non-zero Dirichlet boundary conditions in the interface edges is significant for the LS approach; nevertheless, LS is cheaper than GC in terms of smoothing.

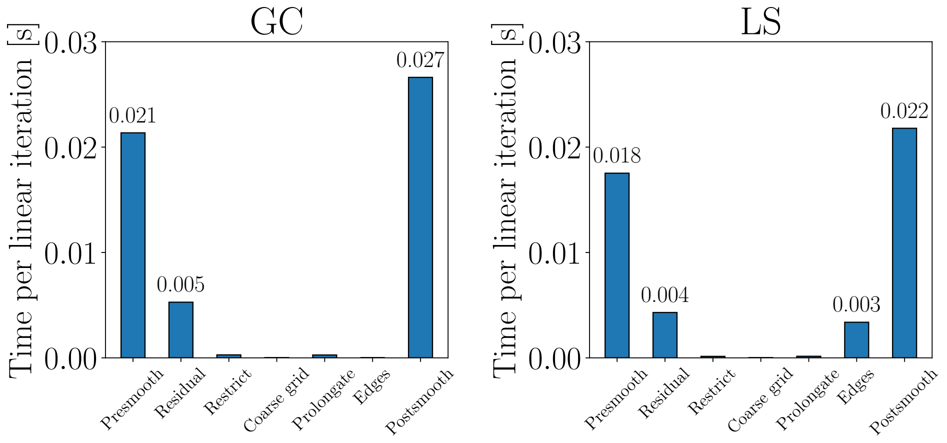

Figure 8.2 Cost of each of the components of the multigrid V-cycle for the Taylor-Couette simulation with a single process for Q_1Q_1 elements using the global coarsening (GC) and the local smoothing (LS) approaches.

8.1.2 Large-Scale Parallel Run

To evaluate the parallel behavior of both multigrid variants, several simulations of the Taylor–Couette flow were performed considering a refinement level ℓ between 2 and 6. The results are shown in **Figure 8.3**. In general, GC scales better and is faster than LS for all refinement levels ℓ. For example, using 8 nodes for Q_2Q_2 elements and a refinement level $\ell = 4$, the GC algorithm is 1.3 times faster than the LS algorithm; when using one node, the difference between both approaches decreases but GC is slightly faster. The reason for this is that even with one node, there is load imbalance for the LS approach.

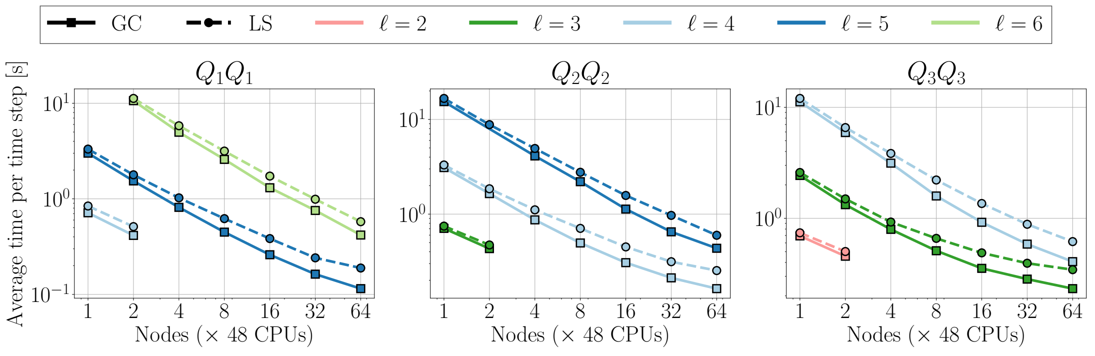

Figure 8.3 Average time per time step for several strong scaling simulations for the Taylor–Couette flow problem considering $4 \leq \ell \leq 6$ for Q_1Q_1, $3 \leq \ell \leq 5$ for Q_2Q_2 and $2 \leq \ell \leq 4$ for Q_3Q_3.

8.2 Flow Around a Sphere: Adaptive Mesh Refinement

This section considers the three-dimensional flow around a sphere with Re = 100. In order to reach the desired Reynolds number, a ramp up of the Re with an initial static mesh is done by performing four simulations with Re = $10, 27, 55$ and 80, before proceeding to solve the case using the latest solution as initial condition, performing the adaptive mesh refinement and solving the problem once again. It is very similar to the flow around a sphere case presented in Section 6.4), however, in this case we consider adaptive mesh adaption by means of a Kelly error estimator [208]. This error estimator approximates the error per cell by integration of the jump of the gradient of a specific variable along the faces of each cell. We use the pressure for error estimation, therefore, the cells around the sphere and downstream will be refined, see **Figure 8.4**. We consider only the inverse diagonal preconditioner with a GMRES method as coarse-grid solver with an ILU preconditioner. The experiments are run with one single process and 192 processes, the cases considered are summarized in **Table 8.4**.

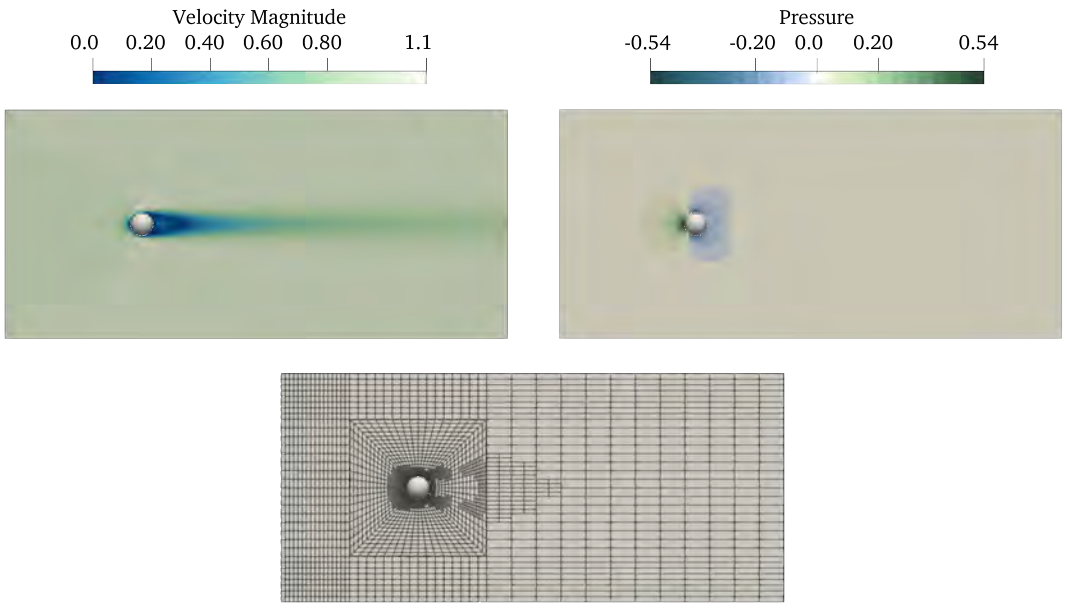

Figure 8.4 Velocity magnitude, pressure and mesh for the simulation of the flow around a sphere case with adaptive mesh refinement.

8.2.1 Serial Run

Serial simulations are considered for all four cases. A summary of the iterations needed for the non-linear and linear solver, and the total time to solution, is presented in **Table 8.5**.

Table 8.4 Summary of the initial number of cells and number of degrees of freedom (DoFs) for all the simulation cases considered for the flow around a sphere with Re = 100 and adaptive mesh refinement.

Initial refinement	**Initial**			**Final**		
	No. Cells	**No. DoFs**		**No. Cells**	**No. DoFs**	
$\boldsymbol{\ell}$		$\boldsymbol{Q_1Q_1}$	$\boldsymbol{Q_2Q_2}$		$\boldsymbol{Q_1Q_1}$	$\boldsymbol{Q_2Q_2}$
1	8K	37K	280K	13K	65K	497K
2	65K	280K	2.1M	109K	482K	3.7M

The results show approximately the same number of iterations and a similar time to solution for both multigrid variants. However, the iterations increase significantly with increasing order p of the elements Q_pQ_p.

The time per linear iteration for the Q_1Q_1 case and $\ell = 2$ is shown in **Figure 8.5**. Most of the time is spent on the finest levels of the multigrid hierarchy; the GC algorithm is more expensive than the LS algorithm since it considers the hanging nodes within the smoothing steps. Even though the coarse-grid has 1024 cells for this case, the cost of the coarse-grid solver is not significant in comparison to the other multigrid levels. The times in terms of the multigrid components are shown in **Figure 8.6**. Similar conclusions as in the transient case are observed; the pre- and post-smoothing steps are the most expensive ones, and the cost of accounting for interface edges is significant in the case of LS.

8.2.2 Moderately Parallel Run

The parallel performance of the solver is evaluated by running the simulations with 192 processes. The summary of the nonlinear and the linear iterations, as well as the time to solution is shown in **Table 8.6**. The GC approach is 1.2 times faster than the LS approach

Table 8.5 Summary of Newton iterations ($\overline{N}_N$), average linear solver iterations per Newton iteration ($\overline{N}_L$), and total time needed to solve the flow around a sphere problem at Re = 100 with a single process using the global coarsening (GC) and local smoothing (LS) approaches.

	$\boldsymbol{Q_1Q_1}$						$\boldsymbol{Q_2Q_2}$					
	GC			**LS**			**GC**			**LS**		
$\boldsymbol{\ell}$	$\overline{N}_N$	$\overline{N}_L$	T [s]	$\overline{N}_N$	$\overline{N}_L$	T [s]	$\overline{N}_N$	$\overline{N}_L$	T [s]	$\overline{N}_N$	$\overline{N}_L$	T [s]
1	6	20	37	6	20	37	3	24	207	3	24	207
2	3	27	173	3	27	182	1	42	363	1	38	346

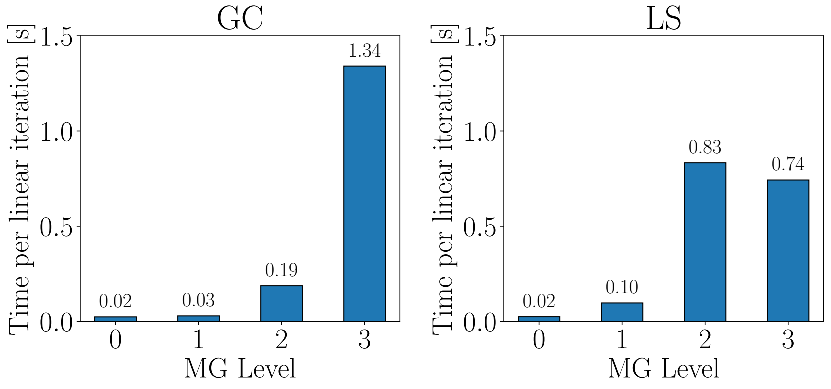

Figure 8.5 Cost of one V-cycle for the flow around a sphere simulation with a single process for $\ell = 2$ and Q_1Q_1 elements using the global coarsening (GC) and the local smoothing (LS) approaches.

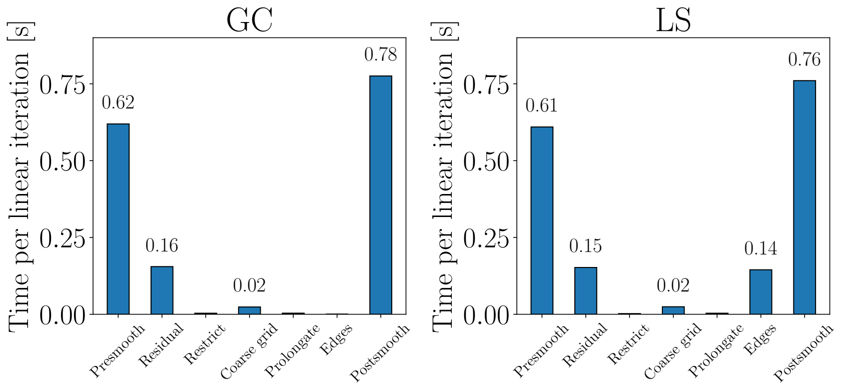

Figure 8.6 Cost of each of the components of the multigrid V-cycle for the flow around a sphere simulation with adaptive mesh refinement a single process for $\ell = 2$ and Q_1Q_1 elements using the global coarsening (GC) and the local smoothing (LS) approaches.

for Q_1Q_1 elements, even with the same number of iterations. A larger difference is observed for Q_2Q_2 elements, with a speedup of 1.6 of GC over LS, even with local smoothing needing slightly less iterations.

Figure 8.7 presents the average, minimum and maximum times spent for each of the multigrid components for the 192 processes. What particularly stands out is that the coarse-grid problem is the bottleneck of these simulations since it has 1024 cells. This is due to the choice of the preconditioner of the coarse-grid iterative solver. In addition to this, one can clearly observe the load imbalance present in the results of the LS algorithm (see the difference be-

Table 8.6 Summary of Newton iterations ($\overline{N}_N$), average linear solver iterations per Newton iteration ($\overline{N}_L$), and total time needed to solve the flow around a sphere problem at Re = 100 with 192 process using the global coarsening (GC) and local smoothing (LS) approaches.

	Q_1Q_1						Q_2Q_2					
	GC			**LS**			**GC**			**LS**		
ℓ	$\overline{N}_N$	$\overline{N}_L$	T [s]	$\overline{N}_N$	$\overline{N}_L$	T [s]	$\overline{N}_N$	$\overline{N}_L$	T [s]	$\overline{N}_N$	$\overline{N}_L$	T [h]
1	6	20	3.46	6	20	3.46	3	25	7.84	3	25	12.7
2	3	27	4.13	3	27	5.22	1	42	7.51	1	39	12.2

tween the bars and the average time across all processes illustrated with a black line). This is expected since the finest levels do not consider the whole computational domain, which makes the distribution of work among several processes difficult.

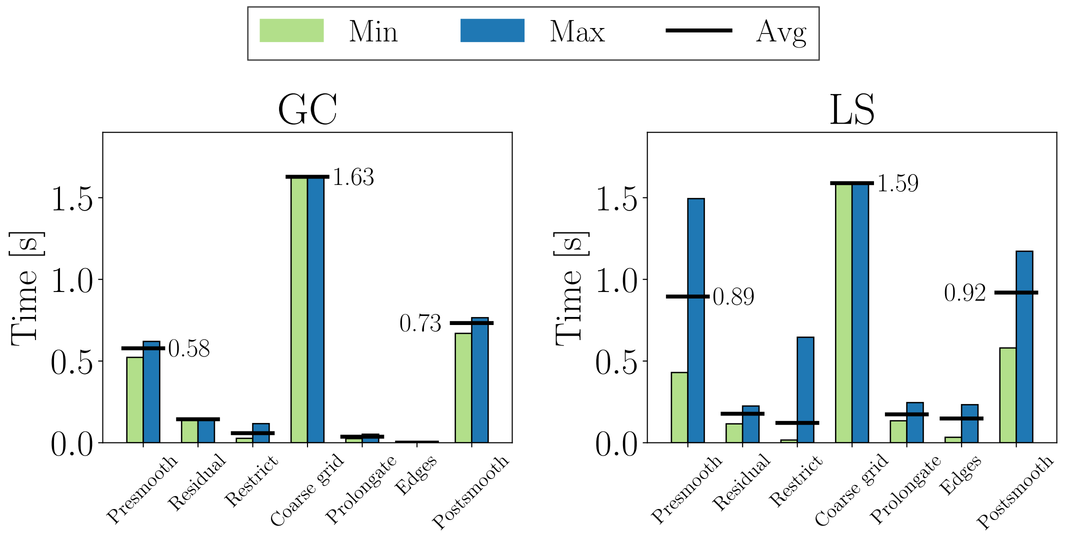

Figure 8.7 Cost of one V-cycle for the flow around a sphere simulation with 192 process for $\ell = 2$ and Q_1Q_1 elements using the global coarsening (GC) and the local smoothing (LS) approaches. It showcases the average time, as well as the minimum and maximum time across all processes.

Two additional metrics can be used to further study the difference between both algorithms: the vertical efficiency and the workload efficiency. The former is an indication of how much data needs to be exchanged during intergrid transfer, a low value, means a large volume of communication. The latter is a measure that considers the amount of cells processed by each processor, a low value translates to load imbalance during smoothing steps. The vertical efficiency for GC has a value of 7.48% while for LS it has a value of 99.83%; this means that there is a large volume of communication for the GC algorithm. In the case of the workload efficiency, LS has a value of 52% and the GC has a value of 99%, indicating a better load balance for the GC algorithm. In our experiments the smoothing steps dominate the overall costs of the V-cycle; as a consequence, workload efficiency is more relevant in our case and indicates that the GC algorithm is more suitable to run these kind of problems in parallel.

CHAPTER 9 GENERAL DISCUSSION

The main objective of this thesis was to develop and implement an efficient and accurate Navier–Stokes solver to model the turbulent flow encountered in process-intensified devices; the goal was fulfilled by adhering to four specific objectives. In what follows, a brief summary of the results and of the conclusions of each specific objective is presented:

Specific Objective 1

This objective focused on studying the turbulence modeling aspect of this project. The results were presented in the form of a publication in Chapter 5. The aim was to study the accuracy of the Implicit Large-Eddy Simulation (ILES) approach when simulating a turbulent flow over periodic hills. This benchmark was chosen because it shares several complex flow features that are also encountered in the flow within process intensified devices: boundary layers, flow separation, flow reattachment, recirculation, and strong pressure gradients. Several simulations with Reynolds numbers $\mathrm{Re} = 5600$ were performed, and the results were compared with experimental and computational data. The latter was obtained with a FVM code that considered a traditional LES approach for modeling turbulence.

In general, the results showed that an ILES approach in the context of FEM with SUPG and PSPG stabilization could accurately predict average velocities, Reynolds stresses, and reattachment point for the flow over periodic hills. The influence of the mesh, the time averaging period used to obtain turbulent statistics, and the choice of time step size were evaluated in detail. If the time-step size is reduced for the simulations considering a coarse mesh (with around 250K cells), the prediction of first- and second-order statistics, as well as reattachment point, deteriorates. On the other hand, for finer meshes (with 1M cells or more), the time-step size has less influence in the predictions and the usage of an implicit time-stepping scheme shows that it is possible to use larger time steps than the ones typically used in the literature without affecting accuracy if an implicit time-stepping scheme is used.

The time averaging period was further studied since in the literature a wide range of values could be found. Simulations were carried out considering 44 to 88 flow-through times, and for all meshes the prediction of average velocities and Reynolds stresses was found to be independent of the time-averaging period. However, in the case of the reattachment point, the time-averaging period and the size of the mesh did have a large influence on the accuracy of the prediction. As the mesh is refined and the time-averaging period is increased, a more accurate prediction of the reattachment point is obtained.

Experiments for a higher Reynolds number Re = 37000 were also conducted and showed promising results for simulations using ILES if the effect of all numerical parameters is taken into account. The final conclusion of this objective is that the ILES approach was indeed promising for the prediction of the behavior of complex turbulent flows and that accurate results can be obtained using coarser meshes than the ones typically used for traditional LES simulations, leading to a reduction in computational cost.

Specific Objective 2

The second objective was concerned with the efficient numerical resolution of the solver. The results were presented in Chapter 6 in the form of a publication. The main goal was to develop an efficient and scalable implementation of the stabilized Navier-Stokes solver. For this, a solver was reimplemented using a matrix-free approach, which also required the implementation of a suitable geometric multigrid preconditioner. To verify and validate the solver the following numerical experiments were carried out: a manufactured solution in 2D, a steady-state flow around a sphere, and the Taylor Green vortex benchmark. All of them considered different boundary conditions and element Peclet numbers; moreover, these test cases allowed us to test the robustness and accuracy of the solver when using elements Q_pQ_p of order $p = 1, 2$ and 3.

The results demonstrated that the solver obtained the right order of convergence for all element orders when using equal-order velocity-pressure elements. A good scalability of the solver was demonstrated for up to 2560 cores and a billion DoFs for steady-state problems and a half billion DoFs for transient problems. The performance of the solver was compared with that of its matrix-based version showing lower solution times, lower memory requirements, and better scalability for all element orders. The difference between the matrix-free and matrix-based versions in terms of time-to-solution was larger as the element order was increased. The usefulness of high-order elements was demonstrated by quantifying the drag coefficient in the case of flow around a sphere. The results showed that a higher accuracy can be obtained with a lower number of degrees of freedom.

We showed that the geometric multigrid preconditioner is essential to achieve that performance. Two smoothers were considered: an inverse diagonal and an additive Schwarz method. The latter was more robust than the former for steady-state simulations, however, it has a larger memory requirement. In the case of transient simulations, the simpler smoother, the inverse diagonal one, was proved to be robust. The main conclusion of this specific objective was that the matrix-free stabilized solver with a monolithic geometric multigrid preconditioner can indeed simulate challenging steady-state and transient problems with a competitive time-to-solution and an efficient use of computational resources.

Specific Objective 3

The third objective aimed to assess the performance of both the ILES approach and the matrix-free solver when simulating rotating flow between two concentric cylinders, also known as the Taylor–Couette benchmark. This specific objective led to a third publication presented in Chapter 7. Nine simulations with different mesh refinements and finite element orders up to $p = 3$ were carried out. The results were validated by comparing them with computational DNS results available in literature. The results showed that the solver can accurately predict the histories of enstrophy and kinetic energy. For the former, the prediction improves as the order of the element is increased and as the mesh is more refined. The kinetic energy prediction is less challenging, and an accurate result is obtained with all meshes except the coarser one with first-order elements.

In addition to those quantities, vorticity and Q criterion distributions were also extracted, and allowed to observe the large and small structured that are captured by each of the simulations with different mesh resolutions. Challenges with high-order rendering, led to the development of a numerical artifical dissipation study via an energy balance in the system. This study allowed to obtain quantitative data to evaluate the effect of this artificial dissipation in the prediction of enstrophy. As expected, the artificial dissipation is decreased as the mesh is refined and the order of the element is increased. Moreover, it is a specific metric for this benchmark that allowed to observe whether it is worth it or not to use high-order elements and raised questions regarding what an appropriate numerical dissipation is deemed acceptable for practical industrial cases. This specific objective also led to the generation of numerical dissipation data that can be used in the future to further compare turbulence modeling approaches.

Specific Objective 4

The fourth objective focused on extending the matrix-free solver for turbulent-flow problems with local refinement. Two geometric multigrid variants were implemented and compared to two problems with locally refined meshes. The first one considered a transient turbulent Taylor–Couette flow with a local static refinement next to the wall, while the latter considered a flow around a sphere with adaptive mesh based on a Kelly error estimator. The efficiency of the multigrid methods was evaluated for serial and parallel computations. The results were obtained in terms of transient, non-linear, and linear iterations. Moreover, a detailed comparison of the time for a V-cycle and the time spent at each of the multigrid components was presented. The results showed that the global coarsening approach is faster than the local smoothing approach for parallel computations, whereas the local smoothing approach is cheaper for serial computations. This is consistent with results previously obtained for simple Poisson problems. Since the flows encountered in process intensification are mostly transient turbulent flows, global coarsening seems to be more suitable for the kind of applications tackled in this work.

CHAPTER 10 CONCLUSION

This chapter concludes this thesis with a summary of the limitations, possible future research directions and a final word.

10.1 Limitations

Even though the main objective was fulfilled through the four specific objectives, the solver developed here has some limitations at the moment:

- The solver has been extensively tested for fluid flow only. Although some experiments have been carried out that combine the solver with heat transfer and a passive tracer (advection-diffusion equation), a detailed analysis of its performance has yet to be conducted, specifically when it comes to the behavior and scalability of the geometric multigrid preconditioners, with a special emphasis on the effectiveness of the smoothers. For an idea of the status of the capabilities of the solver when coupled with a heat transfer solver, refer to Appendix F.

- The matrix-free implementation is not yet compatible for GPU computations. While one of the main advantages of the matrix-free approach is that it is particularly suitable for the usage of GPU, more effort is required on the implementation side to be able to use it. Current developments in `deal.II` indicate that this could be possible in the near future using `Kokkos`, a library that provides abstractions for CUDA programming. A prototype for a Poisson equation and a Stokes problem is already available in `Lethe`, but an extension to Navier-Stokes has not yet been done.

- The solver uses a monolithic approach which is not common for CFD software and a detailed comparison with a segregated approach is yet to be conducted; this was a comment received repeatedly during the peer review process of Article 2 presented in Chapter 6. This thesis has only demonstrated the capabilities and advantages of the developed matrix-free solver over a matrix-based one with the same resolution strategy.

- An implicit time-stepping scheme, BDF2, was used throughout this work. No detailed comparison of this time-stepping scheme with others used in the literature was made.

- The ILES approach was studied in detail when using a SUPG/PSPG stabilization; however, there are more advanced techniques that were not studied at all in the context of this thesis. For example, variational multiscale strategies or even the GLS stabilization technique, which was implemented but not validated.

- The geometric multigrid preconditioner only supports two preconditioners: the inverse diagonal and the additive Schwarz smoother. The latter considers a naive implementation; however, there are other techniques in the literature that could be considered and could further improve the results presented (see [101, 102, 174, 175]). The former works well for transient flows but has difficulties with steady-state problems, further investigation is required on this regard.

10.2 Future Research

This study raises questions that lead to numerous future research possibilities. Some are motivated by the listed limitations: GPU implementation, coupling with other physics, comparison to a matrix-free segregated approach for practical flows, study of time-stepping schemes, other advanced stabilization schemes and futher research in terms of smoothers. Other research directions that arise and their current status can be summarized as follows:

- The implementation of the operators of the other fluid solvers of `Lethe` using a matrix-free implementation. So far, the development of a Volume-Averaged Navier–Stokes solver has started within the research group; which could lead to the first CFD-DEM matrix-free solver. Initial results for this solver were presented at the 12th International Conference for Multiphase flows in 2025 [209].

- Extension of the solver to non-Newtonian flows where the viscosity depends on the shear rate. The matrix-based solver of `Lethe` supports simulations of this type of flow, see the publication by Daunais et al. [163]. The matrix-free solver has already been extended to support this type of flow by adding a new operator that considers additional viscosity models that depend on the shear rate (for details on how it changes the Jacobian and the residual of the operator, see Appendix E). This implementation was accomplished in approximately two weeks, indicating the robustness and flexibility of the architecture developed throughout this work. Further testing is needed with complex benchmarks, however, first results indicate that the multigrid solver is robust.

- Extension to *hp*-FEM since they yield better convergence properties than only *h*- and *p*- adaptive methods and can be interesting in the context of multiple physics [210]. In addition, `deal.II` supports all the underlying structures needed to implement a *hp*-matrix-free Navier–Stokes solver with a geometric multigrid preconditioner. Prototypes for a Poisson solver and an advection-diffusion equation were developed recently during a research group hackathon.

10.3 Final Word

To conclude, the matrix-free stabilized Navier–Stokes solver implemented throughout this thesis is promising for the simulation of turbulent flows encountered within process intensified devices. It requires further work to achieve multiphysics simulations, but it provides a robust and efficient foundation that will eventually allow to fully integrate CFD as a tool in process intensification. Although not shown in this thesis, this solver has already been used successfully to study the hydrodynamics of a Spinning Disc Reactor currently developed at Polytechnique Montreal. The outcome of that work is part of the thesis of another Ph.D. student in an experimental research group (see **Figure 10.1** for an example of the results of a simulation). The author of this thesis firmly believes that with more research and more interdisciplinary collaboration, the usage of CFD as a tool that allows the design of more environmentally friendly devices in a variety of chemical processes will become the rule rather than the exception.

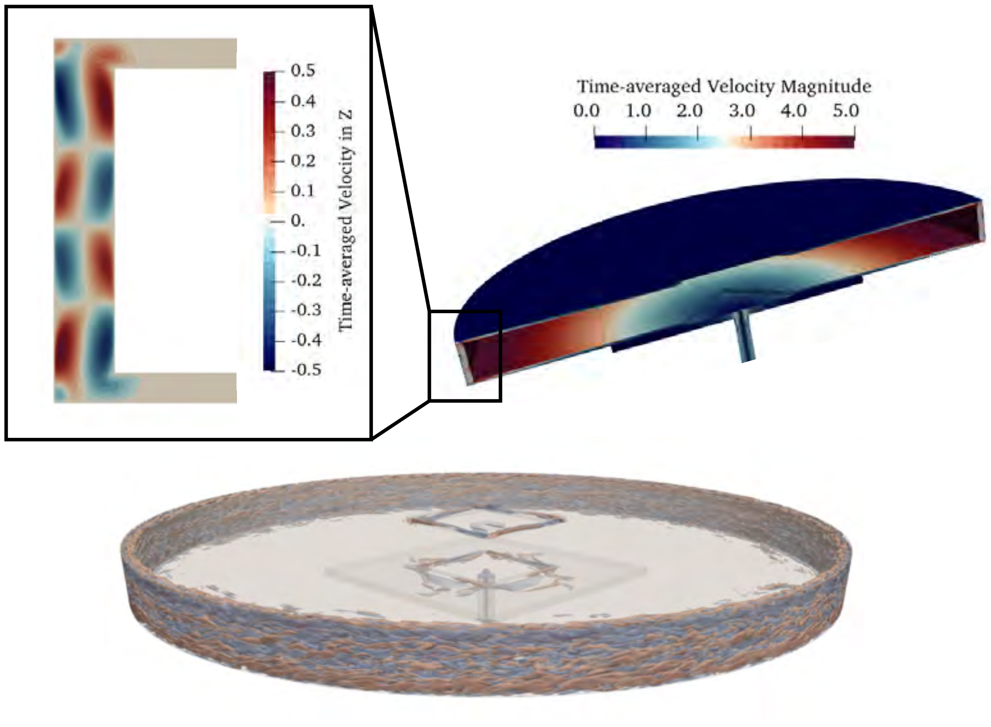

Figure 10.1 Results of a spinning disc reactor simulation performed using the stabilized Navier–Stokes solver developed in this work.

REFERENCES

[1] A. A. Kiss, M. Jobson, and X. Gao, “Reactive distillation: stepping up to the next level of process intensification,” *Ind. Eng. Chem. Res.*, vol. 58, pp. 5909–5918, 2019. [Online]. Available: https://doi.org/10.1021/acs.iecr.8b05450

[2] P. Xie *et al.*, “Characteristics of liquid flow in a rotating packed bed for CO2 capture: A CFD analysis,” *Chemical Engineering Science*, vol. 172, pp. 216–229, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.ces.2017.06.040

[3] D. C. Boffito and D. Fernandez Rivas, “Process intensification connects scales and disciplines towards sustainability,” *The Canadian Journal of Chemical Engineering*, vol. 98, pp. 2489–2506, 2020. [Online]. Available: https://doi.org/10.1002/cjce.23871

[4] W. L. Oberkampf and C. J. Roy, *Verification and validation in scientific computing.* Cambridge University Press, 2010.

[5] United Nations Development Programme, “The SDGS in action.” https://www.undp.org/sustainable-development-goals, 2015, accessed: 2025-05-22.

[6] A. K. Tula, M. R. Eden, and R. Gani, “Computer-aided process intensification: Challenges, trends and opportunities,” *AIChE Journal*, vol. 66, no. 1, pp. 1–12, 2020.

[7] E. A. López-Guajardo *et al.*, “Process intensification 4.0: A new approach for attaining new, sustainable and circular processes enabled by machine learning,” *Chemical Engineering and Processing - Process Intensification*, 2021.

[8] Q. Xiong *et al.*, “Major trends and roadblocks in CFD-aided process intensification of biomass pyrolysis,” *Chemical Engineering and Processing - Process Intensification*, vol. 127, no. April, pp. 206–212, 2018.

[9] E. Quiroz-Pérez, C. Gutiérrez-Antonio, and R. Vázquez-Román, “Modelling of production processes for liquid biofuels through CFD: A review of conventional and intensified technologies,” *Chemical Engineering and Processing - Process Intensification*, vol. 143, no. December 2018, p. 107629, 2019.

[10] F. Ghadyanlou, A. Azari, and A. Vatani, “A review of modeling rotating packed beds and improving their parameters: Gas—liquid contact,” *Sustainability (Switzerland)*, vol. 13, no. 14, pp. 1–42, 2021.

[11] J. Oliveira De Brito Lira *et al.*, “An overview of photoreactors and computational modeling for the intensification of photocatalytic processes in the gas-phase: state-of-art,” *Journal of Environmental Chemical Engineering*, vol. 9, no. 2, 2021.

[12] P. Ranganathan *et al.*, “Recent advances in computational fluid dynamics (CFD) modelling of photobioreactors: design and applications,” *Bioresource Technology*, vol. 350, no. January, p. 126920, 2022.

[13] K. Boodhoo and F. R. Abegão, “HiGee process intensification in biorefineries : innovations , challenges , and outlook,” *Current Opinion in Chemical Engineering*, vol. 48, p. 101119, 2025. [Online]. Available: https://doi.org/10.1016/j.coche.2025.101119

[14] J. M. Lu *et al.*, “New advance in application research of high-gravity process intensification technology,” *Current Opinion in Chemical Engineering*, vol. 47, p. 101057, 2025. [Online]. Available: https://doi.org/10.1016/j.coche.2024.101057

[15] K. Boodhoo, “Spinning disc technology for intensified continuous flow processing: an overview of recent progress and future prospects,” *Chemical Engineering and Processing - Process Intensification*, vol. 212, no. March, 2025.

[16] J. Kleiner *et al.*, “CFD simulation of single-phase heat transfer in a rotor-stator spinning disc reactor,” *Chemical Engineering and Processing - Process Intensification*, vol. 131, no. July, pp. 150–160, 2018.

[17] C. J. Hop *et al.*, “Hydrodynamics of a rotor-stator spinning disk reactor: investigations by large-eddy simulation,” *Physics of Fluids*, vol. 35, no. 3, 2023.

[18] Y. Wang *et al.*, “CFD simulation of liquid flow characteristics in a rotor-stator spinning disc reactor,” *Journal of the Taiwan Institute of Chemical Engineers*, vol. 115, pp. 20–27, 2020.

[19] L. Mazzei *et al.*, “Analysis of a stator-rotor-stator spinning disk reactor in single-phase and two-phase boiling conditions using a thermo-fluid flow network and CFD,” *Fluids*, vol. 7, no. 2, 2022.

[20] V. Gravemeier, “The variational multiscale method for laminar and turbulent incompressible flow,” Ph.D. dissertation, University of Stuttgart, 2003.

[21] P. Davidson, *Turbulence: an introduction for scientists and engineers.* Oxford University Press, 06 2015.

[22] A. Smits, "Lectures in fluid mechanics: viscous flows and turbulence."

[23] *26th ONR Symposium on Naval Hydrodynamics*, ser. A comparative study of RANS, DES and LES, 2006.

[24] Y. Zhiyin, "Large-eddy simulation: past, present and the future," *Chinese Journal of Aeronautics*, vol. 28, no. 1, pp. 11–24, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.cja.2014.12.007

[25] F. F. Grinstein, L. G. Margolin, and W. J. Rider, Eds., *Implicit large eddy simulation: computing turbulent fluid dynamics.* Cambridge University Press, 2007.

[26] S. Pope, *Turbulent flows.* Cambridge University Press, 2000. [Online]. Available: https://books.google.ca/books?id=HZsTw9SMx-0C

[27] N. Fehn, M. Kronbichler, and G. Lube, "From anomalous dissipation through Euler singularities to stabilized finite element methods for turbulent flows," *Flow, Turbulence and Combustion*, vol. 115, no. 1, pp. 347–388, 2025.

[28] J. Donea and A. Huerta, *Finite Element Methods for Flow Problems.* John Wiley & Sons, Ltd, 2003. [Online]. Available: 10.1002/0470013826

[29] H. Elman, D. Silvester, and A. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Oxford University Press, 06 2014.

[30] V. John, *Finite Element Methods for Incompressible Flow Problems*, ser. Springer Series in Computational Mathematics. Springer International Publishing, 2016. [Online]. Available: https://doi.org/10.1007/978-3-319-45750-5

[31] J. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, ser. Texts in Applied Mathematics. Springer New York, 2007. [Online]. Available: https://books.google.ca/books?id=APQkDOmwyksC

[32] D. Gottlieb and S. A. Orszag, *Numerical analysis of spectral methods.* Society for Industrial and Applied Mathematics, 1977. [Online]. Available: 10.1137/1.9781611970425

[33] M. O. Deville, P. F. Fischer, and E. H. Mund, *High-order methods for incompressible fluid flow.* Cambridge University Press, 8 2002, no. 1. [Online]. Available: 10.1017/CBO9780511546792

[34] Z. Wang *et al.*, “High-order CFD methods: current status and perspective,” *Int. J. Numer. Methods Fluids*, vol. 72, no. 8, pp. 811–845, jul 2013.

[35] P. Fischer *et al.*, “Scalability of high-performance PDE solvers,” *International Journal of High Performance Computing Applications*, vol. 34, no. 5, pp. 562–586, 2020.

[36] D. Moxey, R. Amici, and M. Kirby, “Efficient matrix-free high-order finite element evaluation for simplicial elements,” *SIAM J. Sci. Comput.*, vol. 42, no. 3, pp. C97–C123, jan 2020. [Online]. Available: https://epubs.siam.org/doi/10.1137/19M1246523

[37] N. Jansson *et al.*, “Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics,” *Comput. Fluids*, vol. 275, no. March, p. 106243, 2024.

[38] P. Fischer *et al.*, “NekRS, a GPU-accelerated spectral element Navier–Stokes solver,” *Parallel Comput.*, vol. 114, no. August, p. 102982, 2022.

[39] C. D. Cantwell *et al.*, “Nektar++: An open-source spectral/hp element framework,” *Comput. Phys. Commun.*, vol. 192, pp. 205–219, 2015.

[40] D. Arndt *et al.*, “ExaDG: High-order discontinuous galerkin for the exa-scale,” in *Software for exascale computing - SPPEXA 2016-2019*, H.-J. Bungartz *et al.*, Eds. Cham: Springer International Publishing, 2020, pp. 189–224.

[41] A. N. Brooks and T. J. Hughes, “Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 32, no. 1-3, pp. 199–259, sep 1982. [Online]. Available: 10.1016/0045-7825(82)90071-8

[42] T. Tezduyar, “Stabilized finite element formulations for incompressible flow computations,” *Adv. Appl. Math.*, vol. 28, pp. 1–44, 1991.

[43] T. J. Hughes, L. P. Franca, and M. Balestra, “A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzi condition: a stable Petrov-Galerkin formulation of the stokes problem accommodating equal-order interpolations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 59, no. 1, pp. 85–99, 1986. [Online]. Available: 10.1016/0045-7825(86)90025-3

[44] L. P. Franca and T. J. Hughes, “Two classes of mixed finite element methods,” *Computer Methods in Applied Mechanics and Engineering*, vol. 69, no. 1, pp. 89–129, 1988. [Online]. Available: 10.1016/0045-7825(88)90168-5

[45] N. Ahmed *et al.*, "A review of variational multiscale methods for the simulation of turbulent incompressible flows," *Archives of Computational Methods in Engineering*, vol. 24, no. 1, pp. 115–164, 2017.

[46] T. E. Tezduyar, "Stabilized finite element formulations for incompressible flow computations," in *13th Computational Fluid Dynamics Conference*, ser. Advanced Applied Mathematics, J. W. Hutchinson and T. Y. Wu, Eds. Elsevier, 1991, vol. 28, pp. 1–44. [Online]. Available: https://doi.org/10.1016/S0065-2156(08)70153-4

[47] T. J. Hughes, "Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 127, no. 1, pp. 387–401, 1995. [Online]. Available: 10.1016/0045-7825(95)00844-9

[48] F. Brezzi *et al.*, "A priori error analysis of residual-free bubbles for advection-diffusion problems," *SIAM Journal on Numerical Analysis*, vol. 36, no. 6, pp. 1933–1948, jan 1999.

[49] F. Brezzi, D. Marini, and E. Süli, "Residual-free bubbles for advection-diffusion problems: the general error analysis," *Numerische Mathematik*, vol. 85, no. 1, pp. 31–47, mar 2000.

[50] T. J. Hughes *et al.*, "The variational multiscale method - A paradigm for computational mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 166, no. 1-2, pp. 3–24, 1998.

[51] T. J. Hughes, L. Mazzei, and K. E. Jansen, "Large Eddy Simulation and the variational multiscale method," *Computing and Visualization in Science*, vol. 3, no. 1-2, pp. 47–59, 2000.

[52] U. Rasthofer and V. Gravemeier, "Recent developments in variational multiscale methods for large-eddy simulation of turbulent flow," *Archives of Computational Methods in Engineering*, vol. 25, pp. 647–690, 2018.

[53] R. Codina *et al.*, "Variational multiscale methods in computational fluid dynamics," *Encyclopedia of Computational Mechanics Second Edition*, pp. 1–28, 2017.

[54] R. Codina, "Stabilized finite element approximation of transient incompressible flows using orthogonal subscales," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 39-40, pp. 4295–4321, 2002.

[55] Y. Bazilevs *et al.*, “Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows,” *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 1-4, pp. 173–201, 2007.

[56] M. Braack and G. Lube, “Finite elements with local projection stabilization for incompressible flow problems,” *Journal of Computational Mathematics*, vol. 27, no. 2, pp. 116–147, 2009.

[57] V. Gravemeier and W. A. Wall, “Variational multiscale methods for premixed combustion based on a progress-variable approach,” *Combustion and Flame*, vol. 158, no. 6, pp. 1160–1170, 2011.

[58] M. Larson and F. Bengzon, *The finite element method: theory, implementation, and applications*, ser. Texts in Computational Science and Engineering. Springer Berlin Heidelberg, 2013.

[59] J. N. Shahid, R. S. Tuminaro, and H. F. Walker, “An inexact Newton method for fully coupled solution of the Navier–Stokes equations with heat and mass transport,” *Journal of Computational Physics*, vol. 137, no. 1, pp. 155–185, 1997. [Online]. Available: 10.1006/jcph.1997.5798

[60] M. Benzi, G. H. Golub, and J. Liesen, “Numerical solution of saddle point problems,” *Acta Numer.*, vol. 14, pp. 1–137, 2005.

[61] G. Karniadakis and S. Sherwin, *Spectral/hp element methods for computational fluid dynamics, second edition*, ser. Numerical Mathematics and Scientific Computation. OUP Oxford, 2005.

[62] A. Chorin and J. Marsden, *A mathematical introduction to fluid mechanics*, ser. Texts in Applied Mathematics. Springer New York, 2013.

[63] J. L. Guermond, P. Minev, and J. Shen, “An overview of projection methods for incompressible flows,” *Comput. Methods Appl. Mech. Eng.*, vol. 195, no. 44, pp. 6011–6045, 2006.

[64] N. Fehn, “Robust and Efficient Discontinuous Galerkin Methods for Incompressible Flows,” Ph.D. dissertation, Technical University of Munich, 2021.

[65] N. Offermans *et al.*, “On the strong scaling of the spectral element solver Nek5000 on petascale systems,” *ACM International Conference Proceeding Series*, 2016.

[66] J. Liu *et al.*, “The nested block preconditioning technique for the incompressible Navier–Stokes equations with emphasis on hemodynamic simulations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 367, p. 113122, 2020. [Online]. Available: https://doi.org/10.1016/j.cma.2020.113122

[67] H. Elman *et al.*, “A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations,” *Journal of Computational Physics*, vol. 227, no. 3, pp. 1790–1808, 2008.

[68] E. C. Cyr, J. N. Shahid, and R. S. Tuminaro, “Stabilization and scalable block preconditioning for the Navier-Stokes equations,” *J. Comput. Phys.*, vol. 231, no. 2, pp. 345–363, 2012.

[69] Y. Saad, *Iterative methods for sparse linear systems: second edition*, ser. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2003.

[70] W. Briggs, V. Henson, and S. McCormick, *A Multigrid tutorial: second edition*, ser. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2000.

[71] A. Voronin *et al.*, “Monolithic algebraic multigrid preconditioners for the Stokes equations,” 2023.

[72] P. L. Bacq *et al.*, “An all-at-once algebraic multigrid method for finite element discretizations of Stokes problem,” *International Journal for Numerical Methods in Fluids*, vol. 95, no. 2, pp. 193–214, 2023.

[73] S. Saberi, G. Meschke, and A. Vogel, “Adaptive geometric multigrid for the mixed finite cell formulation of Stokes and Navier–Stokes equations,” *International Journal for Numerical Methods in Fluids*, vol. 95, no. 7, pp. 1035–1053, 2023.

[74] A. Heinlein, C. Hochmuth, and A. Klawonn, “Reduced dimension GDSW coarse spaces for monolithic Schwarz domain decomposition methods for incompressible fluid flow problems,” *International Journal for Numerical Methods in Engineering*, vol. 121, no. 6, pp. 1101–1119, 2020.

[75] K. E. Jansen, C. H. Whiting, and G. M. Hulbert, “A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method,” *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 3-4, pp. 305–319, 10 2000. [Online]. Available: 10.1016/S0045-7825(00)00203-6

[76] A. Hay *et al.*, “hp-Adaptive time integration based on the BDF for viscous flows,” *J. Comput. Phys.*, vol. 291, pp. 151–176, 2015.

[77] A. Montlaur, S. Fernandez-Mendez, and A. Huerta, “High-order implicit time integration for unsteady incompressible flows,” *International Journal for Numerical Methods in Fluids*, vol. 70, no. 5, pp. 603–626, oct 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/fld.2703

[78] M. Anselmann and M. Bause, “A geometric multigrid method for space-time finite element discretizations of the Navier-Stokes equations and its application to 3D flow simulation,” *ACM Transactions on Mathematical Software*, vol. 49, no. 1, pp. 1–29, 2023.

[79] W. Bangerth *et al.*, “Algorithms and data structures for massively parallel generic adaptive finite element codes,” *ACM Trans. Math. Softw.*, vol. 38, no. 2, Jan. 2012. [Online]. Available: https://doi.org/10.1145/2049673.2049678

[80] J. Brown *et al.*, “libCEED: Fast algebra for high-order element-based discretizations,” *J. Open Source Softw.*, vol. 6, no. 63, p. 2945, 2021.

[81] M. Kronbichler and K. Kormann, “A generic interface for parallel cell-based finite element operator application,” *Comput. Fluids*, vol. 63, pp. 135–147, 2012.

[82] P. C. Africa *et al.*, “The deal.ii library, version 9.6,” *Journal of Numerical Mathematics*, vol. 32, no. 4, pp. 369–380, 2024. [Online]. Available: https://doi.org/10.1515/jnma-2024-0137

[83] R. Anderson *et al.*, “MFEM: A modular finite element methods library,” *Comput. Math. Appl.*, vol. 81, pp. 42–74, 2021, development and Application of Open-source Software for Problems with Numerical PDEs.

[84] G. F. Castelli, “Numerical investigation of Cahn-Hilliard-type phase-field models for battery active particles,” Ph.D. dissertation, Karlsruher Instituts fuer Technologie, 2021.

[85] S. D. Proell *et al.*, “A highly efficient computational framework for fast scan-resolved simulations of metal additive manufacturing processes on the scale of real parts,” *Additive Manufacturing*, 2023. [Online]. Available: https://doi.org/10.1016/j.addma.2023.103921

[86] P. Munch *et al.*, "On the construction of an efficient finite-element solver for phase-field simulations of many-particle solid-state-sintering processes," *Computational Materials Science*, vol. 231, no. June 2023, p. 112589, 2024. [Online]. Available: https://doi.org/10.1016/j.commatsci.2023.112589

[87] R. Schussnig *et al.*, "Matrix-free higher-order finite element methods for hyperelasticity," *Computer Methods in Applied Mechanics and Engineering*, pp. 1–22, 2024. [Online]. Available: https://doi.org/10.1016/j.cma.2024.117600

[88] S. A. Orszag, "Spectral methods for problems in complex geometries," *J. Comput. Phys.*, vol. 37, no. 1, pp. 70–92, 1980.

[89] P. F. Fischer *et al.*, "Recent advances in parallel spectral element simulation of unsteady incompressible flow," *Comput. Struct.*, vol. 30, pp. 217–231, 1988.

[90] P. F. Fischer, "Analysis and application of a parallel spectral element method for the solution of the Navier-Stokes equations," *Comput. Methods Appl. Mech. Eng.*, vol. 80, no. 1-3, pp. 483–491, 1990.

[91] P. F. Fischer and A. T. Patera, "Parallel spectral element solution of the Stokes problem," *J. Comput. Phys.*, vol. 92, no. 2, pp. 380–421, 1991.

[92] H. M. Tufo and P. F. Fischer, "Terascale spectral element algorithms and implementations," *ACM/IEEE SC 1999 Conference, SC 1999*, pp. 67–68, 1999.

[93] P. F. Fischer, G. W. Kruse, and F. Loth, "Spectral element methods for transitional flows in complex geometries," *J. Sci. Comput.*, vol. 17, no. 1-4, pp. 81–98, 2002.

[94] M. Franco *et al.*, "High-order matrix-free incompressible flow solvers with GPU acceleration and low-order refined preconditioners," *Comput. Fluids*, vol. 203, 2020.

[95] H. S. Tang, R. D. Haynes, and G. Houzeaux, "A review of domain decomposition methods for simulation of fluid flows: Concepts, algorithms, and applications," *Archives of Computational Methods in Engineering*, vol. 28, no. 3, 2021. [Online]. Available: https://doi.org/10.1007/s11831-019-09394-0

[96] V. John and L. Tobiska, "Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 33, no. 4, pp. 453–473, 2000.

[97] N. Kohl and U. Rüde, "Textbook efficiency: massively parallel matrix-free multigrid for the Stokes system," *SIAM J. Sci. Comput.*, vol. 44, no. 2, pp. C124–C155, 2022.

[98] A. Rafiei and S. MacLachlan, "Improved monolithic multigrid methods for high-order Taylor-Hood discretizations," 2025. [Online]. Available: http://arxiv.org/abs/2502.01130

[99] S. P. Vanka, "Block-implicit multigrid solution of Navier-Stokes equations in primitive variables," *Journal of Computational Physics*, vol. 65, no. 1, pp. 138–158, 1986. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999186900082

[100] R. Abu-Labdeh, S. MacLachlan, and P. E. Farrell, "Monolithic multigrid for implicit Runge–Kutta discretizations of incompressible fluid flow," *J. Comput. Phys.*, vol. 478, pp. 1–44, 2023.

[101] A. Voronin *et al.*, "Low-order preconditioning of the Stokes equations," *Numer. Linear Algebra Appl.*, vol. 29, no. 3, 2022.

[102] ——, "Monolithic multigrid preconditioners for high-order discretizations of Stokes equations," 2024.

[103] D. Jodlbauer *et al.*, "Matrix-Free monolithic multigrid methods for Stokes and generalized Stokes problems," *SIAM J. Sci. Comput.*, vol. 46, no. 3, pp. A1599–A1627, 2024.

[104] J. H. Adler *et al.*, "Monolithic multigrid methods for magnetohydrodynamics," *SIAM J. Sci. Comput.*, vol. 43, no. 5, pp. S70–S91, jan 2021.

[105] B. Blais *et al.*, "Lethe: An open-source parallel high-order adaptative CFD solver for incompressible flows," *Softw. X*, vol. 12, p. 100579, 2020.

[106] A. Alphonius *et al.*, "Lethe 1.0: an open-source high-performance and high-order computational fluid dynamics software for single and multiphase flows," 2025. [Online]. Available: 10.2139/ssrn.5090483

[107] C. Burstedde, L. C. Wilcox, and O. Ghattas, "`p4est`: scalable algorithms for parallel adaptive mesh refinement on forests of octrees," *SIAM J. Sci. Comput.*, vol. 33, no. 3, pp. 1103–1133, 2011.

[108] T. Trilinos project team, *The Trilinos project website*, 2024 (acccessed Sep 9, 2024). [Online]. Available: https://trilinos.github.io

[109] D. Arndt *et al.*, "The `deal.II` library, version 9.5," *J. Numer. Math.*, vol. 31, no. 3, pp. 231–246, 2023.

[110] B. Krank *et al.*, "A high-order semi-explicit discontinuous Galerkin solver for 3D incompressible flow with application to DNS and LES of turbulent channel flow," *Journal of Computational Physics*, vol. 348, pp. 634–659, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2017.07.039

[111] N. Fehn, W. A. Wall, and M. Kronbichler, "Robust and efficient discontinuous Galerkin methods for under-resolved turbulent incompressible flows," *J. Comput. Phys.*, vol. 372, pp. 667–693, 2018.

[112] P. Munch *et al.*, "Efficient distributed matrix-free multigrid methods on locally refined meshes for FEM computations," *ACM Trans. Parallel Comput.*, pp. 1–34, 2023.

[113] T. C. Clevenger *et al.*, "A flexible, parallel, adaptive geometric multigrid method for FEM," *ACM Trans. Math. Softw.*, vol. 47, no. 1, 2020.

[114] V. Ryaben'kii and S. Tsynkov, *A Theoretical Introduction to Numerical Analysis.* CRC Press, 2006.

[115] P. Munch and M. Kronbichler, "Cache-optimized and low-overhead implementations of additive Schwarz methods for high-order FEM multigrid computations," *Int. J. High Perform. Comput. Appl.*, vol. 38, no. 3, pp. 192–209, 2024.

[116] R. Varga, *Matrix iterative analysis*, ser. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 1999.

[117] A. Voronin, "Monolithic multigrid for saddle point systems," Ph.D. dissertation, University of Illinois Urbana-Champaign, 2024.

[118] The Amesos2 project team, *The Amesos2 project website*, 2024 (acccessed Sep 9, 2024). [Online]. Available: https://trilinos.github.io/amesos2.html

[119] The ML project team, *The ML project website*, 2024 (acccessed Sep 9, 2024). [Online]. Available: https://trilinos.github.io/ml.html

[120] The Ifpack project team, *The Ifpack project website*, 2024 (acccessed Sep 9, 2024). [Online]. Available: https://trilinos.github.io/ifpack.html

[121] ERCOFTAC, "2d periodic hill flow," 2017. [Online]. Available: https://kbwiki.ercoftac.org/w/index.php?title=Abstr:2D_Periodic_Hill_Flow

[122] X. Gloerfelt and P. Cinnella, "Large eddy simulation requirements for the flow over periodic hills," *Flow Turbul. Combust.*, vol. 103, p. 55–91, 06 2019. [Online]. Available: https://doi.org/10.1007/s10494-018-0005-5

[123] A. D. Beck *et al.*, “High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations,” *Int. J. Numer. Methods Fluids*, vol. 76, no. 8, pp. 522–548, 2014. [Online]. Available: https://doi.org/10.1002/fld.3943

[124] R. Mokhtarpoor and S. Heinz, “Dynamic large eddy simulation: Stability via realizability,” *Phys. Fluids*, vol. 29, no. 10, p. 105104, 2017. [Online]. Available: https://doi.org/10.1063/1.4986890

[125] R. Wang *et al.*, “Implicit large-eddy simulations of turbulent flow in a channel via spectral/hp element methods,” *Phys. Fluids*, vol. 33, no. 3, p. 035130, 2021. [Online]. Available: https://doi.org/10.1063/5.0040845

[126] S. Hickel, T. Kempe, and N. A. Adams, “Implicit large-eddy simulation applied to turbulent channel flow with periodic constrictions,” *Theor. Comput. Fluid Dyn.*, vol. 22, p. 227–242, 2008. [Online]. Available: https://doi.org/10.1007/s00162-007-0069-7

[127] Z. L. Chen *et al.*, “Wall modeling for implicit large-eddy simulation and immersed-interface methods,” *Theor. Comput. Fluid Dyn.*, vol. 28, pp. 1–21, 2014. [Online]. Available: https://doi.org/10.1007/s00162-012-0286-6

[128] Z. Li, Y. Zhang, and H. Chen, “A low dissipation numerical scheme for implicit large eddy simulation,” *Comput. Fluids*, vol. 117, pp. 233–246, 2015. [Online]. Available: https://doi.org/10.1016/j.compfluid.2015.05.016

[129] P. Balakumar and G. I. Park, “DNS/LES simulations of separated flows at high Reynolds numbers,” in *49th AIAA Fluid Dynamics Conference*, 2015. [Online]. Available: https://doi.org/10.2514/6.2015-2783

[130] B. Krank, M. Kronbichler, and W. A. Wall, “Direct numerical simulation of flow over periodic hills up to re=10,595,” *Flow Turbul. Combust.*, vol. 101, pp. 521–551, 2018. [Online]. Available: https://doi.org/10.1007/s10494-018-9941-3

[131] C. Rapp, “Experimentelle studie der turbulenten strömung über periodische hügel,” Ph.D. dissertation, Technische Universität München, 01 2009. [Online]. Available: https://mediatum.ub.tum.de/node?id=677970

[132] M. Breuer *et al.*, “Flow over periodic hills – numerical and experimental study in a wide range of Reynolds numbers,” *Comput. Fluids*, vol. 38, no. 2, pp. 433–457, 2009. [Online]. Available: https://doi.org/10.1016/j.compfluid.2008.05.002

[133] D. Arndt *et al.*, “The `deal.II` library, version 9.3,” *J. Numer. Math.*, vol. 29, no. 3, pp. 171–186, 2021. [Online]. Available: https://doi.org/10.1515/jnma-2021-0081

[134] ——, “The `deal.II` library, version 9.4,” *J. Numer. Math.*, vol. 30, no. 3, pp. 231–246, 2022. [Online]. Available: https://doi.org/10.1515/jnma-2022-0054

[135] C. P. Mellen, J. Fröhlich, and W. Rodi, “Large-eddy simulation of the flow over periodic hills,” in *16th IMACS world congress*, ser. 16th IMACS world congress, M. Deville and R. Owens, Eds., Lausanne, Switzerland, August 2000.

[136] C. Rapp and M. Manhart, “Flow over periodic hills: An experimental study,” *Exp. Fluids*, vol. 51, pp. 247–269, 07 2011. [Online]. Available: https://doi.org/10.1007/s00348-011-1045-y

[137] J. Fröhlich *et al.*, “Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constrictions,” *J. Fluid Mech.*, vol. 526, p. 19–66, 2005. [Online]. Available: https://doi.org/10.1017/S0022112004002812

[138] X. Gloerfelt and P. Cinnella, “Investigation of the flow dynamics in a channel constricted by periodic hills,” in *45th AIAA Fluid Dynamics Conference.* American Institute of Aeronautics and Astronautics (AIAA), 2015. [Online]. Available: https://doi.org/10.2514/6.2015-2480

[139] C. Benocci and A. Pinelli, “The role of the forcing term in the large eddy simulation of equilibrium channel flow,” in *Engineering Turbulence Modeling and Experiments.* Elsevier, 1990, pp. 287–296, international Symposium On Engineering Turbulence Modelling And Measurements , Dubrovnik, Yugoslavia, Sep 24-28, 1990. [Online]. Available: https://eprints.ucm.es/id/eprint/21905/

[140] W. Wang, “A non-body conformal grid method for simulations of laminar and turbulent flows with a compressible large eddy simulation solver,” Ph.D. dissertation, Iowa State University, 2009. [Online]. Available: https://www.proquest.com/dissertations-theses/non-body-conformal-grid-method-simulations/docview/304905306/se-2

[141] B. Chaouat and R. Schiestel, “Hybrid RANS/LES simulations of the turbulent flow over periodic hills at high Reynolds number using the PITM method,” *Comput. Fluids*, vol. 84, pp. 279–300, 2013. [Online]. Available: https://doi.org/10.1016/j.compfluid.2013.06.012

[142] R. Mokhtarpoor, S. Heinz, and M. Stoellinger, “Dynamic unified RANS-LES simulations of high Reynolds number separated flows,” *Phys. Fluids*, vol. 28, no. 9, 2016. [Online]. Available: http://dx.doi.org/10.1063/1.4961254

[143] M. De la Llave Plata, V. Couaillier, and M. C. le Pape, “DNS and LES of the flow over periodic hills based on a discontinuous Galerkin approach,” *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, vol. 135, pp. 27–40, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-60387-2_3

[144] G. Lodato and J. B. Chapelier, “Evaluation of the spectral element dynamic model for LES on unstructured, deformed meshes,” *ERCOFTAC Series*, vol. 25, pp. 39–45, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-04915-7_6

[145] L. T. Diosady and S. M. Murman, “Dns of flows over periodic hills using a discontinuous-Galerkin spectral-element method,” in *44th AIAA Fluid Dynamics Conference, 16-20 June 2014, Atlanta, GA*, A. I. of Aeronautics and Astronautics, Eds. American Institute of Aeronautics and Astronautics (AIAA), 2014. [Online]. Available: https://doi.org/10.2514/6.2014-2784

[146] H. Xiao, S. Laizet, and L. Duan, “Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations,” *Comput. Fluids*, p. 104431, 01 2020. [Online]. Available: https://doi.org/10.1016/j.compfluid.2020.104431

[147] Z. Chen, “Wall Modeling for Implicit Large-Eddy Simulation,” Ph.D. dissertation, Technical University of Munich, 2010. [Online]. Available: https://mediatum.ub.tum.de/node?id=1007276

[148] M. Kornhaas, D. C. Sternel, and M. Schäfer, “Influence of Time Step Size and Convergence Criteria on Large Eddy Simulations with Implicit Time Discretization,” in *Quality and Reliability of Large-Eddy Simulations*, J. Meyers, B. J. Geurts, and P. Sagaut, Eds. Springer-Verlag Berlin Heidelberg, 2008, vol. 12, p. 119. [Online]. Available: https://doi.org/10.1007/978-1-4020-8578-9_10

[149] T. E. Tezduyar *et al.*, “Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements,” *Comput. Methods Appl. Mech. Eng.*, vol. 95, no. 2, pp. 221–242, 1992.

[150] F. Ilinca, K. R. Yu, and B. Blais, “The effect of viscosity on free surface flow inside an angularly oscillating rectangular tank,” *Comput. Fluids*, 2019. [Online]. Available: https://doi.org/10.1016/j.compfluid.2019.02.021

[151] B. Blais and F. Ilinca, "Development and validation of a stabilized immersed boundary CFD model for freezing and melting with natural convection," *Comput. Fluids*, 2018. [Online]. Available: https://doi.org/10.1016/j.compfluid.2018.03.037

[152] M. C. Hsu *et al.*, "Improving stability of stabilized and multiscale formulations in flow simulations at small time steps," *Comput. Methods Appl. Mech. Eng.*, vol. 199, no. 13-16, pp. 828–840, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cma.2009.06.019

[153] R. Calderer and A. Masud, "Residual-based variational multiscale turbulence models for unstructured tetrahedral meshes," *Comput. Methods Appl. Mech. Eng.*, vol. 254, pp. 238–253, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.cma.2012.09.015

[154] P. Gamnitzer, V. Gravemeier, and W. A. Wall, "Time-dependent subgrid scales in residual-based large eddy simulation of turbulent channel flow," *Comput. Methods Appl. Mech. Eng.*, vol. 199, no. 13-16, pp. 819–827, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cma.2009.07.009

[155] J. P. Slotnick *et al.*, "CFD vision 2030 study: a path to revolutionary computational aerosciences," Langley Research Center, Tech. Rep. 20140003093, March 2014.

[156] J. Brown, "Efficient nonlinear solvers for nodal high-order finite elements in 3D," *J. Sci. Comput.*, vol. 45, no. 1-3, pp. 48–63, 2010.

[157] C. D. Cantwell *et al.*, "From h to p efficiently: Strategy selection for operator evaluation on hexahedral and tetrahedral elements," *Comput. Fluids*, vol. 43, no. 1, pp. 23–28, 2011.

[158] A. J. Chorin, "A numerical method for solving incompressible viscous flow problems," *J. Comput. Phys.*, vol. 2, no. 1, pp. 12–26, 1967.

[159] G. E. Karniadakis, M. Israeli, and S. A. Orszag, "High-order splitting methods for the incompressible Navier-Stokes equations," *J. Comput. Phys.*, vol. 97, no. 2, pp. 414–443, 1991.

[160] P. Ohm *et al.*, "A monolithic algebraic multigrid framework for multiphysics applications with examples from resistive MHD," vol. 94661, pp. 1–29, 2021. [Online]. Available: 10.48550/arXiv.2103.07537

[161] F. Brezzi and M. Fortin, *Mixed and hybrid finite element methods*, ser. Springer Series in Computational Mathematics. Springer New York, 2012.

[162] L. Prieto Saavedra *et al.*, “An implicit large-eddy simulation perspective on the flow over periodic hills,” *Comput. Fluids*, vol. 283, p. 106390, 2024.

[163] C.-A. Daunais, L. Barbeau, and B. Blais, “An extensive study of shear thinning flow around a spherical particle for power-law and Carreau fluids,” *J. Non-Newtonian Fluid Mech.*, vol. 311, p. 104951, 2023.

[164] T. El Geitani, S. Golshan, and B. Blais, “Toward high-order CFD-DEM: development and validation,” *Ind. Eng. Chem. Res.*, vol. 62, no. 2, pp. 1141–1159, 2023.

[165] V. Bibeau *et al.*, “Artificial neural network to predict the power number of agitated tanks fed by CFD simulations,” *Can. J. Chem. Eng.*, vol. 101, no. 10, pp. 5992–6002, 2023.

[166] T. E. Geitani, S. Golshan, and B. Blais, “A high-order stabilized solver for the volume averaged Navier-Stokes equations,” *Int. J. Numer. Methods Fluids*, vol. 95, no. 6, pp. 1011–1033, 2023.

[167] L. Barbeau *et al.*, “High-order moving immersed boundary and its application to a resolved CFD-DEM model,” *Comput. Fluids*, vol. 268, p. 106094, 2024.

[168] P. Wesseling and C. W. Oosterlee, “Geometric multigrid with applications to computational fluid dynamics,” *J. Comput. Appl. Math.*, vol. 128, no. 1-2, pp. 311–334, 2001.

[169] D. Knoll and D. Keyes, “Jacobian-free Newton–Krylov methods: a survey of approaches and applications,” *J. Comput. Phys.*, vol. 193, no. 2, pp. 357–397, Jan. 2004.

[170] P. Munch *et al.*, “On the construction of an efficient finite-element solver for phase-field simulations of many-particle solid-state-sintering processes,” *Comput. Mater. Sci*, vol. 231, p. 112589, 2024.

[171] N. Fehn *et al.*, “Hybrid multigrid methods for high-order discontinuous Galerkin discretizations,” *J. Comput. Phys.*, vol. 415, p. 109538, 2020.

[172] P. Bastian *et al.*, “Matrix-free multigrid block-preconditioners for higher order discontinuous Galerkin discretisations,” *J. Comput. Phys.*, vol. 394, pp. 417–439, 2019.

[173] W. Pazner and P.-O. Persson, “Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods,” *J. Comput. Phys.*, vol. 354, pp. 344–369, 2018.

[174] P. D. Brubeck and P. E. Farrell, “A scalable and robust vertex-star relaxation for high-order FEM,” *SIAM J. Sci. Comput.*, vol. 44, no. 5, pp. A2991–A3017, 2022.

[175] R. Abu-Labdeh, S. MacLachlan, and P. E. Farrell, “Monolithic multigrid for implicit Runge–Kutta discretizations of incompressible fluid flow,” *J. Comput. Phys.*, vol. 478, p. 111961, 2023.

[176] B. Blais and F. Bertrand, “On the use of the method of manufactured solutions for the verification of CFD codes for the volume-averaged Navier–Stokes equations,” *Comput. Fluids*, vol. 114, pp. 121–129, 2015.

[177] F. Ham and G. Iaccarino, “Energy conservation in collocated discretization schemes on unstructured meshes,” *CTR Annu. Res. Briefs*, vol. 2004, no. 3-14, p. 118, 2004.

[178] H. Sundar *et al.*, “Parallel geometric-algebraic multigrid on unstructured forests of octrees,” in *SC ’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012, pp. 1–11.

[179] C. Siefert *et al.*, “Algebraic multigrid techniques for discontinuous Galerkin methods with varying polynomial order,” *Comput Geosci*, vol. 18, pp. 597–612, 2014.

[180] F. W. Roos and W. W. Willmarth, “Some experimental results on sphere and disk drag,” *AIAA Journal*, vol. 9, no. 2, pp. 285–291, 1971.

[181] R. Knaus, “A fast matrix-free approach to the high-order control volume finite element method with application to low-mach flow,” *Comput. Fluids*, vol. 239, p. 105408, 2022.

[182] Y. Li *et al.*, “Assessment of wind hazard at wind turbine sites based on CFD simulation under tropical cyclone conditions,” *Sustainable Energy Technol. Assess.*, vol. 73, p. 104109, 2025.

[183] S. Kasmaiee, S. Kasmaiee, and A. Farshad, “Unsteady CFD simulation of a rotor blade under various wind conditions,” *Sci Rep*, vol. 14, p. 19176, 2024.

[184] S. Hawner *et al.*, “Development of carbon composite blades within the context of the experimental validation of a CFD-based design tool for contra-rotating, electric fan engines,” *Aerosp.*, vol. 11, no. 7, 2024.

[185] M. Bargal *et al.*, “CFD modeling of two-phase flow (oil/air) with and without rotary mixer inside a vertical pipe for upstream of multiphase pump,” *Arab J Sci Eng*, 2024.

[186] P. Meeuwse and M. van Lieshout, “General applicable residence time distribution model to estimate reaction rates in a rotor–stator spinning disc reactor,” *ChemEngineering*, vol. 9, no. 1, 2025.

[187] G. Bhat, A. Qayoum, and S. Saleem, “Multi-parameter assessment of fluid flow and heat transfer in a spinning disk: COMSOL Multiphysics® simulation and experimental insight,” *Multiscale and Multidiscip. Model. Exp.*, 2024.

[188] S. Grossmann, D. Lohse, and C. Sun, “High – Reynolds number Taylor-Couette turbulence,” *Annual Review of Fluid Mechanics*, 2016.

[189] F. Wendt, “Turbulente Strömungen zwischen zwei rotierenden konaxialen Zylindern,” *Ing. Arch.*, vol. 4, no. 6, pp. 577–591, 1933.

[190] C. D. Andereck, S. S. Liu, and H. L. Swinney, “Flow regimes in a circular Couette system with independently rotating cylinders,” *J. Fluid. Mech.*, vol. 164, p. 155–183, 1986.

[191] D. P. Lathrop, J. Fineberg, and H. L. Swinney, “Transition to shear-driven turbulence in Couette-Taylor flow,” *Phys. Rev. A*, vol. 46, pp. 6390–6405, Nov 1992.

[192] ——, “Turbulent flow between concentric rotating cylinders at large Reynolds number,” *Phys. Rev. Lett.*, vol. 68, pp. 1515–1518, Mar 1992.

[193] G. S. Lewis and H. L. Swinney, “Velocity structure functions, scaling, and transitions in high-Reynolds-number Couette-Taylor flow.” *Phys. Rev. E*, vol. 59 5 Pt B, pp. 5457–67, 1999.

[194] S. Dong, “Direct numerical simulation of turbulent Taylor - Couette flow,” *J. Fluid. Mech.*, vol. 587, pp. 373–393, 2007.

[195] D. Pirrò and M. Quadrio, “Direct numerical simulation of turbulent Taylor-Couette flow,” *Eur. J. Mech. B. Fluids*, vol. 27, no. 5, pp. 552–566, 2008.

[196] R. Ostilla-Mónico, R. Verzicco, and D. Lohse, “Effects of the computational domain size on direct numerical simulations of Taylor-Couette turbulence with stationary outer cylinder,” *Phys. Fluids*, vol. 27, no. 2, 2015. [Online]. Available: http://dx.doi.org/10.1063/1.4913231

[197] S. Poncet, S. Viazzo, and R. Oguic, “Large eddy simulations of Taylor-Couette-Poiseuille flows in a narrow-gap system,” *Phys. Fluids*, vol. 105108, no. 26, 2014.

[198] W. Cheng, D. Pullin, and R. Samtaney, “Large-eddy simulation and modelling of Taylor-Couette flow,” *J. Fluid. Mech.*, vol. 890, 2020.

[199] Y. Bazilevs and I. Akkerman, “Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and the residual-based variational multiscale method,” *J. Fluid. Mech.*, vol. 229, no. 9, pp. 3402–3414, 2010.

[200] L. Aydinbakar *et al.*, “Space–time VMS isogeometric analysis of the Taylor–Couette flow,” *Comput. Mech.*, vol. 67, no. 5, pp. 1515–1541, 2021.

[201] Z. J. Wang and E. Jourdan, “Benchmark for scale-resolving simulation with curved walls: the Taylor Couette flow,” *Adv. Aerodyn.*, vol. 3, no. 1, 2021.

[202] R. Bird, W. Stewart, and E. Lightfoot, *Transport Phenomena*, ser. Transport Phenomena. Wiley, 2006, no. v. 1. [Online]. Available: https://books.google.ca/books?id=L5FnNlIaGfcC

[203] F. S. Schranner *et al.*, “Assessing the numerical dissipation rate and viscosity in numerical simulations of fluid flows,” *Comput. Fluids*, vol. 114, pp. 84–97, Jul. 2015.

[204] F. Cadieux, G. Sun, and J. A. Domaradzki, “Effects of numerical dissipation on the interpretation of simulation results in computational fluid dynamics,” *Comput. Fluids*, vol. 154, pp. 256–272, Sep. 2017.

[205] L. Prieto Saavedra, P. Munch, and B. Blais, “A matrix-free stabilized solver for the incompressible Navier-Stokes equations,” *J. Compt. Phys.*, 2024.

[206] W. Bangerth and O. Kayser-Herold, “Data structures and requirements for hp finite element software,” *ACM Trans. Math. Softw.*, vol. 36, no. 1, Mar. 2009.

[207] K. R. Holst, R. S. Glasby, and R. B. Bond, “On the effect of temporal error in high-order simulations of unsteady flows,” *J. Comput. Phys.*, vol. 402, p. 108989, 2020.

[208] D. W. Kelly *et al.*, “A posteriori error analysis and adaptive processes in the finite element method: Part i—error analysis,” *International Journal for Numerical Methods in Engineering*, vol. 19, no. 11, pp. 1593–1619, 1983. [Online]. Available: https://doi.org/10.1002/nme.1620191103

[209] ICMF, “International conference on multiphase flows,” https://www.icmf2025.com/, 2025, accessed: 2025-06-16.

[210] M. Fehling and W. Bangerth, “Algorithms for Parallel Generic hp-Adaptive Finite Element Software,” *ACM Transactions on Mathematical Software*, vol. 49, no. 3, 2023.

APPENDIX A ALTERNATIVES TO THE NAVIER–STOKES EQUATIONS

Two alternative turbulence modeling approaches to DNS and ILES were introduced: the Reynolds-averaged Navier-Stokes (RANS) and Large-Eddy Simulation (LES) approaches. The equations solved in RANS are obtained by applying Reynolds decomposition and time-averaging to the Navier–Stokes equations. The equations solved in LES are the filtered Navier–Stokes equations, which are derived using spatial filtering instead of time averaging. In the following, the derivation of both alternatives is presented.

Reynolds-Averaged Navier–Stokes equations

The basis of RANS is the Reynolds decomposition of velocity and pressure into mean and fluctuating components:

$$\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}', \quad p = \overline{p} + p', \tag{A.1}$$

where $\overline{\boldsymbol{u}}$ and $\overline{p}$ are mean quantities averaged in time and $\boldsymbol{u}'$ and p' are the fluctuating components. The decomposition is inserted into the continuity equation and the momentum equation:

$$\nabla \cdot (\overline{\boldsymbol{u}} + \boldsymbol{u}') = \nabla \cdot \overline{\boldsymbol{u}} + \nabla \cdot \boldsymbol{u}' = 0, \tag{A.2}$$

$$\partial_t (\overline{\boldsymbol{u}} + \boldsymbol{u}') + ((\overline{\boldsymbol{u}} + \boldsymbol{u}') \cdot \nabla) (\overline{\boldsymbol{u}} + \boldsymbol{u}') + \nabla (\overline{p} + p') - \nu \Delta (\overline{\boldsymbol{u}} + \boldsymbol{u}') - \boldsymbol{f} = 0. \tag{A.3}$$

Then, the time average can be applied, taking into account the following properties: $\overline{\boldsymbol{u}'} = 0$, $\overline{\boldsymbol{u} + \boldsymbol{v}} = \overline{\boldsymbol{u}} + \overline{\boldsymbol{v}}$, $\overline{\overline{\boldsymbol{u}}} = \overline{\boldsymbol{u}}$, $\overline{\nabla \cdot \boldsymbol{u}} = \nabla \overline{\boldsymbol{u}}$, and $\overline{\boldsymbol{u} \cdot \boldsymbol{v}} \neq \overline{\boldsymbol{u}} \cdot \overline{\boldsymbol{v}}$. This leads to the following for the continuity equation:

$$\overline{\nabla \cdot \overline{\boldsymbol{u}}} + \overline{\nabla \cdot \boldsymbol{u}'} = \nabla \cdot \overline{\boldsymbol{u}} + \nabla \cdot \overline{\boldsymbol{u}'} = \nabla \cdot \overline{\boldsymbol{u}} = 0. \tag{A.4}$$

For the momentum equation, the following is obtained:

$$\partial_t \overline{\boldsymbol{u}} + (\overline{\boldsymbol{u}} \cdot \nabla) \overline{\boldsymbol{u}} + \overline{(\boldsymbol{u}' \cdot \nabla) \boldsymbol{u}'} + \nabla \overline{p} - \nu \Delta \overline{\boldsymbol{u}} - \boldsymbol{f} = 0. \tag{A.5}$$

For the second term arising from the convective term, one can use the product rule to rewrite it as:

$$\overline{(\boldsymbol{u}' \cdot \nabla)\boldsymbol{u}'} = \nabla \left(\overline{\boldsymbol{u}' \otimes \boldsymbol{u}'}\right), \tag{A.6}$$

which yields the Reynolds stress term that is model through a turbulence model, such as κ-ϵ and κ-ω.

Filtered Navier–Stokes equations

In the context of Large-Eddy simulation $\overline{\boldsymbol{u}}$ and $\overline{p}$ are the filtered (or resolved) components, and $\boldsymbol{u}'$ and p' are the residual or subgrid-scale (SGS) components. A filter is usually defined as:

$$\overline{\boldsymbol{u}} = \int_\Omega G(\boldsymbol{r}, \boldsymbol{x})\boldsymbol{u}(\boldsymbol{x} - \boldsymbol{r}, t)\mathrm{d}\boldsymbol{r}, \tag{A.7}$$

and the special filter function G satisfied the normalization condition $\int_\Omega G(\boldsymbol{r}, \boldsymbol{x})\mathrm{d}\boldsymbol{r} = 1$. There are several filter functions; see the turbulence books by Davidson [21] and Pope [26] for an overview. The subgrid-scale components are defined as:

$$\boldsymbol{u}' = \boldsymbol{u} - \overline{\boldsymbol{u}},\ p' = p - \overline{p}. \tag{A.8}$$

The main difference between this decomposition and the one in the case of RANS is that $\overline{\boldsymbol{u}}$ is a random field and, in general, the filtered fluctuation is not equal to zero, that is, $\overline{\boldsymbol{u}'} \neq 0$. The spatial filtering can then be applied to the continuity equation and the momentum equation in a similar way as in RANS but obtaining a different convective term:

$$\nabla \cdot \overline{\boldsymbol{u}} = 0, \tag{A.9}$$

$$\partial_t \overline{\boldsymbol{u}} + \nabla \overline{\boldsymbol{u} \otimes \boldsymbol{u}} + \nabla \overline{p} - \nu \Delta \overline{\boldsymbol{u}} - \boldsymbol{f} = 0. \tag{A.10}$$

Then the subgrid-scale stress tensor is defined as:

$$\boldsymbol{\tau}_{\mathrm{SGS}} = \overline{\boldsymbol{u} \otimes \boldsymbol{u}} - \overline{\boldsymbol{u}} \otimes \overline{\boldsymbol{u}}, \tag{A.11}$$

to rewrite the momentum equation as follows:

$$\partial_t \overline{\boldsymbol{u}} + \nabla \overline{\boldsymbol{u}} \otimes \overline{\boldsymbol{u}} + \nabla \cdot \boldsymbol{\tau}_{\mathrm{SGS}} + \nabla \overline{p} - \nu \Delta \overline{\boldsymbol{u}} - \boldsymbol{f} = 0. \tag{A.12}$$

One can further develop the subgrid-scale stress tensor. For this, we use $\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}'$ as follows:

$$\boldsymbol{u} \otimes \boldsymbol{u} = \overline{\boldsymbol{u}} \otimes \overline{\boldsymbol{u}} + \overline{\boldsymbol{u}} \otimes \boldsymbol{u}' + \boldsymbol{u}' \otimes \overline{\boldsymbol{u}} + \boldsymbol{u}' \otimes \boldsymbol{u}', \tag{A.13}$$

and apply the filtering operator to obtain:

$$\overline{\boldsymbol{u} \otimes \boldsymbol{u}} = \overline{\overline{\boldsymbol{u}} \otimes \overline{\boldsymbol{u}}} + \overline{\overline{\boldsymbol{u}} \otimes \boldsymbol{u}'} + \overline{\boldsymbol{u}' \otimes \overline{\boldsymbol{u}}} + \overline{\boldsymbol{u}' \otimes \boldsymbol{u}'}, \tag{A.14}$$

which finally leads to:

$$\boldsymbol{\tau}_{\text{SGS}} = \underbrace{\overline{\overline{\boldsymbol{u}} \otimes \overline{\boldsymbol{u}}} - \overline{\boldsymbol{u}} \otimes \overline{\boldsymbol{u}}}_{\text{Leonard stress tensor}} + \underbrace{\overline{\overline{\boldsymbol{u}} \otimes \boldsymbol{u}'} + \overline{\boldsymbol{u}' \otimes \overline{\boldsymbol{u}}}}_{\text{Cross stress tensor}} + \underbrace{\overline{\boldsymbol{u}' \otimes \boldsymbol{u}'}}_{\text{SGS Reynolds stress}} . \tag{A.15}$$

The Leonard-stress tensor that contains all terms that depend only on the resolved filter solution $\overline{\boldsymbol{u}}$. The cross stress tensor has both the resolved and subgrid scale quantities, while the SGS tensor is defined only in terms of the unresolved velocity $\boldsymbol{u}'$. As in RANS this leads to a closure problem, and this tensor must be modeled as a function of resolved quantities. The models used for this closure are known as subgrid-scale models. There are several models; one of them is the well-known Smagorinsky model. This derivation is based on [26] and [52].

APPENDIX B NEWTON'S METHOD FOR THE RESOLUTION OF NONLINEAR SYSTEMS

Algorithm 3 presents a pseudocode of a Newton method for the Navier–Stokes equations with a simple line-search strategy.

Algorithm 3 Pseudocode for Newton's algorithm with line search mechanism to find the solution $\boldsymbol{x}$ of a system of nonlinear equations by solving a linear system $\boldsymbol{F'}\delta\boldsymbol{x} = \boldsymbol{R}$ for each Newton iteration using a suitable preconditioner $\boldsymbol{M}$. An initial guess $\boldsymbol{x}^{(0)}$, a tolerance ϵ and a maximum number of iterations n_{max} is needed to start the iteration.

```
function NEWTON($\boldsymbol{x}^{(0)}$, $n_{max}$, $\epsilon$)
    $n \leftarrow 0$
    while ($r < \epsilon$ and $n < n_{max}$) do
        $\boldsymbol{F'}^{(n)} \leftarrow$ assemble_matrix($\boldsymbol{x}^{(n)}$)
        $\boldsymbol{M}^{(n)} \leftarrow$ setup_preconditioner($\boldsymbol{F'}^{(n)}$)        // e.g., AMG, ILU
        $\boldsymbol{R}^{(n)} \leftarrow$ assemble_rhs($\boldsymbol{x}^{(n)}$)
        $r^{(n)} \leftarrow$ l2_norm($\boldsymbol{R}^{(n)}$)
        $\delta\boldsymbol{x}^{(n)} \leftarrow$ solve_linear_system($\boldsymbol{F'}^{(n)}, \boldsymbol{R}^{(n)}, \boldsymbol{M}^{(n)}$)        // e.g., GMRES solver
        $r_\alpha \leftarrow r^{(n)}$
        for ($\alpha = 1,\ \alpha > 10^{-1}; \alpha = 0.5\alpha$) do        // Line search algorithm
            $\boldsymbol{x}^{n+1} \leftarrow \boldsymbol{x}^{n} + \alpha\delta\boldsymbol{x}^{n}$
            $\boldsymbol{R}^{(n+1)} \leftarrow$ assemble_rhs($\boldsymbol{x}^{(n+1)}$)
            $r^{(n+1)} \leftarrow$ l2_norm($\boldsymbol{R}^{(n+1)}$)
            if ($r^{(n+1)} > r_\alpha$) then        // Keep same alpha if residual is increased
                $\alpha = 2\alpha$
                $\boldsymbol{x}^{n+1} \leftarrow \boldsymbol{x}^{n} + \alpha\delta\boldsymbol{x}^{n}$
                break;
            $r_\alpha \leftarrow r^{(n+1)}$
        $n \leftarrow n + 1$
        $r^{(n)} \leftarrow$ l2_norm($\boldsymbol{R}^{(n+1)}$)
        return $\boldsymbol{x}$
```

APPENDIX C LINEARIZED SYSTEM OF THE NAVIER–STOKES EQUATIONS

The variational formulation of the steady-state incompressible Navier–Stokes equations and a zero-traction boundary condition is given as:

$$\begin{aligned} F(\boldsymbol{u},p) := {} & (q, \nabla\cdot\boldsymbol{u})_\Omega + (\boldsymbol{v}, \partial_t\boldsymbol{u}) + (\boldsymbol{v}, (\boldsymbol{u}\cdot\nabla)\boldsymbol{u})_\Omega \\ & - (p, \nabla\cdot\boldsymbol{v})_\Omega + \nu(\nabla\boldsymbol{u}, \nabla\boldsymbol{v})_\Omega - (\boldsymbol{v}, \boldsymbol{f})_\Omega = 0, \end{aligned} \tag{C.1}$$

where $(\cdot,\cdot)_\Omega$ denotes the L_2-inner product in a domain Ω. We seek to find a system of the form $\boldsymbol{F'}(\boldsymbol{u},p)[\delta\boldsymbol{u},\delta p]^\top = -[R_u, R_p]$. The right-hand side corresponds to the terms that involve the velocity or the pressure in the strong residual $R(\boldsymbol{u},p) = F(\boldsymbol{u},p)$. The matrix $\boldsymbol{F'}$ corresponds to the Jacobian and it is found by means of a Gateaux derivative (or directional derivative):

$$\boldsymbol{F'}(\boldsymbol{u},p)[\delta\boldsymbol{u},\delta p]^\top = \lim_{\epsilon\to 0}\frac{R(\boldsymbol{u}+\epsilon\delta\boldsymbol{u}, p+\epsilon\delta p) - R(\boldsymbol{u},p)}{\epsilon} \tag{C.2}$$

This yields:

$$\begin{aligned} R(\boldsymbol{u}+\epsilon\delta\boldsymbol{u}, p+\epsilon\delta p) = {} & (q, \nabla\cdot(\boldsymbol{u}+\epsilon\delta\boldsymbol{u}))_\Omega + (\boldsymbol{v}, \partial_t(\boldsymbol{u}+\epsilon\delta\boldsymbol{u})) \\ & + (\boldsymbol{v}, ((\boldsymbol{u}+\epsilon\delta\boldsymbol{u})\cdot\nabla)(\boldsymbol{u}+\epsilon\delta\boldsymbol{u}))_\Omega \\ & - (p+\epsilon\delta p, \nabla\cdot\boldsymbol{v})_\Omega + \nu(\nabla(\boldsymbol{u}+\epsilon\delta\boldsymbol{u}), \nabla\boldsymbol{v})_\Omega - (\boldsymbol{v}, f) \\ = {} & (q, \nabla\cdot\boldsymbol{u})_\Omega + (\boldsymbol{v}, \partial_t\boldsymbol{u}) + (\boldsymbol{v}, (\boldsymbol{u}\cdot\nabla)\boldsymbol{u})_\Omega \\ & - (p, \nabla\cdot\boldsymbol{v})_\Omega + \nu(\nabla\boldsymbol{u}, \nabla\boldsymbol{v})_\Omega - (\boldsymbol{v}, \boldsymbol{f})_\Omega \\ & + (q, \nabla\cdot\epsilon\delta\boldsymbol{u})_\Omega + (\boldsymbol{v}, \partial_t(\epsilon\delta\boldsymbol{u})) \\ & + (\boldsymbol{v}, (\epsilon\delta\boldsymbol{u}\cdot\nabla)\boldsymbol{u})_\Omega + (\boldsymbol{v}, (\boldsymbol{u}\cdot\nabla)\epsilon\delta\boldsymbol{u})_\Omega + (\boldsymbol{v}, (\epsilon\delta\boldsymbol{u}\cdot\nabla)\epsilon\delta\boldsymbol{u})_\Omega, \\ & - (\epsilon\delta p, \nabla\cdot\boldsymbol{v})_\Omega + \nu(\nabla(\epsilon\delta\boldsymbol{u}), \nabla\boldsymbol{v})_\Omega \end{aligned} \tag{C.3}$$

which leads to:

$$\begin{aligned} \boldsymbol{F'}(\boldsymbol{u},p)[\delta\boldsymbol{u},\delta p]^\top = {} & (q, \nabla\cdot\delta\boldsymbol{u})_\Omega + (\boldsymbol{v}, \partial_t\delta\boldsymbol{u}) + (\boldsymbol{v}, (\delta\boldsymbol{u}\cdot\nabla)\boldsymbol{u})_\Omega + (\boldsymbol{v}, (\boldsymbol{u}\cdot\nabla)\delta\boldsymbol{u})_\Omega \\ & -(\delta p, \nabla\cdot\boldsymbol{v})_\Omega + \nu(\nabla(\delta\boldsymbol{u}), \nabla\boldsymbol{v})_\Omega. \end{aligned} \tag{C.4}$$

Introducing a set of vector-valued basis functions $\boldsymbol{\phi}_j$ for the velocity and a scalar basis functions ψ_j for the pressure, and using a CG formulation, i.e., $\boldsymbol{\phi}_i$ for $\boldsymbol{v}$ and ψ_i for q, the Jacobian is given as:

$$\begin{aligned}\boldsymbol{F'}(\boldsymbol{u},p)[\delta\boldsymbol{u},\delta p]^\top =&\sum_j \delta u_j(\psi_i,\nabla\cdot\boldsymbol{\phi}_j)_\Omega+(\boldsymbol{\phi}_i,\partial_t\boldsymbol{\phi}_j)+\nu(\nabla\boldsymbol{\phi}_j,\nabla\boldsymbol{\phi}_i)_\Omega\\&+\sum_j\delta u_j(\boldsymbol{\phi}_i,(\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{u})_\Omega+(\boldsymbol{\phi}_i,(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_j)_\Omega\\&-\sum_j\delta p_j(\psi_j,\nabla\cdot\boldsymbol{\phi}_i)_\Omega,\end{aligned}\tag{C.5}$$

and the right hand side as:

$$\begin{aligned}R(\boldsymbol{u},p)=&(\psi_i,\nabla\cdot\boldsymbol{u})_\Omega+(\boldsymbol{\phi}_i,\partial_t\boldsymbol{u})+(\boldsymbol{\phi}_i,(\boldsymbol{u}\cdot\nabla)\boldsymbol{u})_\Omega\\&-(p,\nabla\cdot\boldsymbol{\phi}_i)_\Omega+\nu(\nabla\boldsymbol{u},\nabla\boldsymbol{\phi}_i)_\Omega-(\boldsymbol{\phi}_i,\boldsymbol{f})_\Omega,\end{aligned}\tag{C.6}$$

Writing Eq. C.5 and Eq. C.6 in matrix-vector form:

$$\begin{bmatrix}\boldsymbol{A}&\boldsymbol{B}\\\boldsymbol{B}^\top&\boldsymbol{0}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{U}\\\delta\boldsymbol{P}\end{bmatrix}=-\begin{bmatrix}\boldsymbol{R}_u\\\boldsymbol{R}_p\end{bmatrix},\tag{C.7}$$

where the matrix $\boldsymbol{A}$ comprises the convective and diffusive terms, while the matrix $\boldsymbol{B}$ includes the gradient term. For an entry (i,j) and an element k:

$$\boldsymbol{A}^k_{i,j}=(\boldsymbol{\phi}_i,\partial_t\boldsymbol{\phi}_j+(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_j+(\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{u})_{\Omega_k}+\nu(\nabla\boldsymbol{\phi}_j,\boldsymbol{\phi}_i)_{\Omega_k}\tag{C.8}$$

$$\boldsymbol{B}^k_{i,j}=-(\nabla\cdot\boldsymbol{\phi}_i,\psi_j)_{\Omega_k}.\tag{C.9}$$

The residual terms are similarly defined as:

$$\boldsymbol{R}^k_{u(i,j)}=-(\boldsymbol{\phi}_i,\partial_t\boldsymbol{u})-(\boldsymbol{\phi}_i,(\boldsymbol{u}\cdot\nabla)\boldsymbol{u})_{\Omega_k}+(p,\nabla\cdot\boldsymbol{\phi}_i)_{\Omega_k}-\nu(\nabla\boldsymbol{u},\nabla\boldsymbol{\phi}_i)_{\Omega_k}+(\boldsymbol{\phi}_i,\boldsymbol{f})_{\Omega_k}\tag{C.10}$$

$$\boldsymbol{R}^k_{p(i,j)}=(\psi_i,\nabla\cdot\boldsymbol{u})_{\Omega_k}\tag{C.11}$$

APPENDIX D LINEARIZED SYSTEM OF THE NAVIER–STOKES EQUATIONS WITH SUPG/PSPG STABILIZATION

When using a CG formulation along with stabilization techniques, terms are added to the Jacobian and the residual of the system of equations. In this section, the additional terms for the steady-state Navier–Stokes equations are presented. They are found using the same procedure as the traditional formulation shown in Appendix C. For the PSPG term:

$$\boldsymbol{F}_{PSPG} = \sum_k (\tau_{\text{PSPG}} \nabla q, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} - \boldsymbol{f})_{\Omega_k}, \tag{D.1}$$

and the Jacobian contribution is given as:

$$\boldsymbol{F}'_{PSPG}(\boldsymbol{u}, p)[\delta\boldsymbol{u}, \delta p]^\top = \sum_k (\tau_{\text{PSPG}} \nabla q, \partial_t \delta\boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\delta\boldsymbol{u} + (\delta\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla \delta p - \nu \Delta \delta\boldsymbol{u})_{\Omega_k}. \tag{D.2}$$

By introducing a set of vector-valued basis functions $\boldsymbol{\phi_j}$ for the velocity and a scalar basis function ψ_j for the pressure, and using a CG formulation, i.e., $\boldsymbol{\phi}_i$ for $\boldsymbol{v}$ and ψ_i for q, the Jacobian contribution is given as:

$$\begin{aligned}\boldsymbol{F}'_{PSPG}(\boldsymbol{u}, p)[\delta\boldsymbol{u}, \delta p]^\top =& \sum_k \sum_j u_j (\tau_{\text{PSPG}} \nabla \psi_i, \partial_t \boldsymbol{\phi}_j + (\boldsymbol{u} \cdot \nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{\phi}_j)_{\Omega_k} \\ &+ \sum_k \sum_j p_j (\tau_{\text{PSPG}} \nabla \psi_i, \nabla \psi_j),\end{aligned} \tag{D.3}$$

and the right-hand side takes the following form:

$$\boldsymbol{R}_{PSPG}(\boldsymbol{u}, p)[\delta\boldsymbol{u}, \delta p]^\top = \sum_k (\tau_{\text{PSPG}} \nabla \psi_i, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u})_{\Omega_k}. \tag{D.4}$$

The SUPG terms contribute more terms to the final formulation due to the non-linearity. Let us recall its definition:

$$\boldsymbol{F}_{SUPG} = \sum_k (\tau_{\text{SUPG}} (\boldsymbol{u} \cdot \nabla)\boldsymbol{v}, \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} - \boldsymbol{f})_{\Omega_k}. \tag{D.5}$$

The Jacobian is given as:

$$\begin{aligned}\boldsymbol{F}'_{SUPG}(\boldsymbol{u},p)[\delta\boldsymbol{u},\delta p]^\top =&\sum_k(\tau_{\mathrm{SUPG}}(\boldsymbol{u}\cdot\nabla)\boldsymbol{v},\partial_t\delta\boldsymbol{u}+(\boldsymbol{u}\cdot\nabla)\delta\boldsymbol{u}+(\delta\boldsymbol{u}\cdot\nabla)\boldsymbol{u}+\nabla\delta p-\nu\Delta\delta\boldsymbol{u})_{\Omega_k}\\ &+\sum_k(\tau_{\mathrm{SUPG}}(\delta\boldsymbol{u}\cdot\nabla)\boldsymbol{v},\partial_t\boldsymbol{u}+(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}+\nabla p-\nu\Delta\boldsymbol{u}-\boldsymbol{f})_{\Omega_k}.\end{aligned} \tag{D.6}$$

Introducing the set of basis functions, which were also used before for the PSPG term, leads to two terms since the test function in the SUPG term includes the velocity $\boldsymbol{u}$:

$$\begin{aligned}\boldsymbol{F}'_{SUPG,1}(\boldsymbol{u},p)[\delta\boldsymbol{u},\delta p]^\top =&\sum_k\sum_j\delta u_j(\tau_{\mathrm{SUPG}}(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_i,\partial_t\boldsymbol{\phi}_j+(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_j+(\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{u}-\nu\Delta\boldsymbol{\phi}_j)_{\Omega_k}\\ &+\sum_k\sum_j\delta p_j(\tau_{\mathrm{SUPG}}(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_i,\nabla\psi_j)_{\Omega_k},\end{aligned} \tag{D.7}$$

and:

$$\begin{aligned}\boldsymbol{F}'_{SUPG,2}(\boldsymbol{u},p)[\delta\boldsymbol{u},\delta p]^\top =&\sum_k\sum_j u_j(\tau_{\mathrm{SUPG}}(\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{\phi}_i,\partial_t\boldsymbol{u}+(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}-\nu\Delta\boldsymbol{u}-\boldsymbol{f})_{\Omega_k}\\ &+\sum_k\sum_j p_j(\tau_{\mathrm{SUPG}}(\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{\phi}_i,\nabla p)_{\Omega_k}.\end{aligned} \tag{D.8}$$

and a right-hand side:

$$\boldsymbol{R}_{SUPG}=\sum_k(\tau_{\mathrm{SUPG}}(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_i,\partial_t\boldsymbol{u}+(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}+\nabla p^*-\nu\Delta\boldsymbol{u}-\boldsymbol{f})_{\Omega_k}, \tag{D.9}$$

In matrix-vector form, the SUPG and PSPG terms modify the system of the traditional CG formulation as follows:

$$\begin{bmatrix}\widehat{\boldsymbol{A}} & \widehat{\boldsymbol{B}}\\ \boldsymbol{C} & \boldsymbol{D}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{U}\\ \delta\boldsymbol{P}\end{bmatrix}=-\begin{bmatrix}\widehat{\boldsymbol{R}}_u\\ \widehat{\boldsymbol{R}}_p\end{bmatrix}, \tag{D.10}$$

where an entry (i, j) for an element k on each of the blocks is defined as:

$$\begin{aligned}
\widehat{\boldsymbol{A}}^k_{i,j} &= \boldsymbol{A}^k_{i,j} + (\tau_{\mathrm{SUPG}}(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_i, \partial_t\boldsymbol{\phi}_j + (\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{\phi}_j)_{\Omega_k} \\
&\quad + (\tau_{\mathrm{SUPG}}(\boldsymbol{\phi_j}\cdot\nabla)\boldsymbol{\phi}_i, \partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{u} - \boldsymbol{f})_{\Omega_k} && \text{(D.11)} \\
&\quad + (\tau_{\mathrm{SUPG}}(\boldsymbol{\phi_j}\cdot\nabla)\boldsymbol{\phi}_i, \nabla p)_{\Omega_k}, \\
\widehat{\boldsymbol{B}}^k_{i,j} &= \boldsymbol{B}^k_{i,j} + (\tau_{\mathrm{SUPG}}(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_i, \nabla\psi_j)_{\Omega_k}, && \text{(D.12)} \\
\boldsymbol{C}^k_{i,j} &= \boldsymbol{B}^k_{j,i} + (\tau_{\mathrm{PSPG}}\nabla\psi_i, \partial_t\boldsymbol{\phi}_j + (\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_j + (\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{\phi}_j)_{\Omega_k}, && \text{(D.13)} \\
\boldsymbol{D}^k_{i,j} &= (\tau_{\mathrm{PSPG}}\nabla\psi_i, \nabla\psi_j). && \text{(D.14)}
\end{aligned}$$

The residual vectors also have additional terms which are expressed as:

$$\begin{aligned}
\widehat{\boldsymbol{R}}^k_{u(i,j)} =&\boldsymbol{R}^k_{u(i,j)} - \sum_k(\tau_{\mathrm{SUPG}}(\boldsymbol{u}\cdot\nabla)\boldsymbol{\phi}_i, \partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p^* - \nu\Delta\boldsymbol{u} - \boldsymbol{f})_{\Omega_k}, && \text{(D.15)} \\
\widehat{\boldsymbol{R}}^k_{p(i,j)} =&\boldsymbol{R}^k_{p(i,j)} - \sum_k(\tau_{\mathrm{PSPG}}\nabla\psi_i, \partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p^* - \nu\Delta\boldsymbol{u})_{\Omega_k}. && \text{(D.16)}
\end{aligned}$$

APPENDIX E SYSTEM OF EQUATIONS FOR NON-NEWTONIAN FLOW

The matrix-free solver was extended by adding an operator for non-Newtonian flow (this was joint work with Victor Oliveira Ferreira and Bruna Campos); preliminary results for a flow around a sphere in 2D show a good behavior of the GMG preconditioner. More tests need to be conducted in the future. In the following, the equations required for the operator are presented. In non-Newtonian flow, the viscosity depends on the shear rate; therefore, the diffusive term of the Navier–Stokes equations is modified as follows:

$$\partial_i u_i = 0 \tag{E.1}$$

$$\partial_t u_i + u_k \partial_k u_i + \partial_i p + \partial_k \tau_{ik} - f_i = 0, \tag{E.2}$$

where $\tau_{ik} = -\eta(\partial_i u_k + \partial_k u_i)$ is the deviatoric stress tensor and η is the dynamic viscosity. For non-Newtonian fluids, η is a function of the shear rate magnitude ($\dot{\gamma}$) and is the main source of changes in the formulation. For shear-thinning flows, two models are common (and currently supported by `Lethe`): the Power-law model and the Carreau model. For detailed information refer to the publication by Daunais et al. [163]. In what follows, we look at the residual (weak derivative), the strong residual for stabilization terms and the Jacobian.

Residual

To obtain the residual one can simply compute the weak form of the Navier–Stokes equations with the deviatoric stress tensor and integrate by parts the pressure terms and the diffusive term as follows:

$$(v_i, \partial_i p)_\Omega = (v_i n_i, p)_\Gamma - (\partial_i v_i, p)_\Omega, \tag{E.3}$$

$$(v_i, \partial_k \tau_{ik})_\Omega = (v_i, \tau_{ik} n_k)_\Gamma - (\partial_k v_i, \tau_{ik})_\Omega. \tag{E.4}$$

Setting the correct boundary conditions, the weak formulation is given as:

$$(q, \partial_i u_i)_\Omega + (v_i, \partial_t u_i + u_k \partial_k u_i - f_i)_\Omega - (\partial_k v_i, \tau_{ik})_\Omega - (\partial_i v_i, p)_\Omega = 0. \tag{E.5}$$

The same expression can be written in terms of τ_{ik}:

$$(q, \partial_i u_i)_\Omega + (v_i, \partial_t u_i + u_k \partial_k u_i - f_i)_\Omega + (\partial_k v_i, \eta(\partial_i u_k + \partial_k u_i))_\Omega - (\partial_i v_i, p)_\Omega = 0, \tag{E.6}$$

yielding the residual for the right-hand side of the matrix system.

Strong residual for stabilization terms

For non-Newtonian flows, the continuity equation does not change and the momentum equation (Eq. E.2) is as follows:

$$\partial_t u_i + u_k \partial_k u_i + \partial_i p - \partial_k \eta (\partial_i u_k + \partial_k u_i) - f_i = 0. \tag{E.7}$$

The viscous term can be further developed by using the product rule:

$$\begin{aligned} -\partial_k \eta (\partial_i u_k + \partial_k u_i) &= -(\partial_k \eta)(\partial_i u_k + \partial_k u_i) - \eta \partial_k (\partial_i u_k + \partial_k u_i) && \text{(E.8)} \\ &= -(\partial_k \eta)(\partial_i u_k + \partial_k u_i) - \eta \cancelto{\partial_k u_k = 0}{\partial_k \partial_i u_k} - \eta \partial_k \partial_k u_i && \text{(E.9)} \\ &= -(\partial_k \eta)(\partial_i u_k + \partial_k u_i) - \eta \partial_k \partial_k u_i; && \text{(E.10)} \end{aligned}$$

replacing this in Eq. E.7, yields:

$$\partial_t u_i + u_k \partial_k u_i + \partial_i p - (\partial_k \eta)(\partial_i u_k + \partial_k u_i) - \eta \partial_k \partial_k u_i - f_i = 0, \tag{E.11}$$

where $\partial_k \eta$ is computed using the linearization point, i.e., the solution at the previous Newton step.

Jacobian

The Jacobian is found using the Gateau derivative for the weak from presented in Eq. E.6. The term coming from the continuity equation, the time derivative, the convective term, the pressure term and the force term remain the same but considering the test and solution functions for $\boldsymbol{u}$, $\boldsymbol{v}$, p and q. The only term that changes is the diffusive term:

$$(\partial_k v_i, \eta(\partial_i u_k + \partial_k u_i))_\Omega. \tag{E.12}$$

The computation of the directional derivative in this case is more difficult, since the viscosity depends on the shear-rate magnitude, which in turn depends on the shear-rate. We start by writing the derivative in another notation ($\frac{\partial}{\partial u_k}$) and using the chain rule as follows:

$$\begin{aligned}\frac{\partial}{\partial u_k}(\partial_k v_i, \eta(\partial_i u_k + \partial_k u_i))_\Omega = (\partial_k v_i, \eta \frac{\partial}{\partial u_k}\,(\partial_i u_k + \partial_k u_i))_\Omega \\ + (\partial_k v_i, \left(\frac{\partial \eta}{\partial u_k}\right)(\partial_i u_k + \partial_k u_i))_\Omega\end{aligned} \tag{E.13}$$

Since the viscosity depends on the shear rate magnitude and the shear rate magnitude depends on the shear rate, the derivative of the viscosity can be written as:

$$\frac{\partial \eta}{\partial u_k} = \frac{\partial \eta}{\partial \dot{\gamma}} \frac{\partial \dot{\gamma}}{\partial u_k}. \tag{E.14}$$

The first term depends on the specific model used for the viscosity (e.g., the power law model or the Carreau model). However, for the second term, one can obtain a concrete expression by exploiting the definition of the shear rate magnitude:

$$\dot{\gamma} = \left(\frac{\gamma_{ij}\gamma_{ij}}{2}\right)^{1/2}, \tag{E.15}$$

where $\gamma_{ij} = (\partial_i u_k + \partial_k u_i)$. The derivative can be calculated as follows:

$$\frac{\partial \dot{\gamma}}{\partial u_k} = \frac{1}{2\dot{\gamma}} \underbrace{(\partial_i u_k + \partial_k u_i)(\partial_i u_k + \partial_k u_i)}_{\Pi(u)} \tag{E.16}$$

Defining $\Pi(u + \epsilon \delta u)$ as:

$$\Pi(u + \epsilon\delta u) = (\partial_i(u_k + \epsilon\delta u_k) + \partial_k(u_i + \epsilon\delta u_i))(\partial_i(u_k + \epsilon\delta u_k) + \partial_k(u_i + \epsilon\delta u_i)), \tag{E.17}$$

and computing the directional derivative:

$$\lim_{\epsilon \to 0} \frac{\Pi(u + \epsilon\delta u) - \Pi(u)}{\epsilon} = 2(\partial_i u_k + \partial_k u_i)(\partial_i \delta u_k + \partial_k \delta u_i) \tag{E.18}$$

This leads to the following final additional term for the Jacobian:

$$\begin{aligned}\frac{\partial}{\partial u_k}(\partial_k v_i, \eta(\partial_i u_k + \partial_k u_i))_\Omega = (\partial_k v_i, \eta\,(\partial_i \delta u_k + \partial_k \delta u_i))_\Omega + \\ \left(\partial_k v_i, \frac{\partial \eta}{\partial \dot{\gamma}} \frac{1}{2\dot{\gamma}}(2(\partial_i u_k + \partial_k u_i)(\partial_i \delta u_k + \partial_k \delta u_i)(\partial_i u_k + \partial_k u_i)\right)_\Omega\end{aligned} \tag{E.19}$$

APPENDIX F NON-ISOTHERMAL FLOW SIMULATION

The main equation to solve for heat transfer can be derived from the energy equation in incompressible flows. For a constant heat capacity C_p and dynamic viscosity μ, the equation takes the following form:

$$\rho C_p \frac{\partial T}{\partial t} + \rho C_p (\boldsymbol{u} \cdot \nabla) T - \nabla \cdot (k \nabla T) = -\phi + Q, \tag{F.1}$$

where T is the temperature, $\boldsymbol{u}$ is the velocity of the fluid, ρ is the density, C_p is the isobaric heat capacity, k is the thermal conductivity, Q is the energy source term, and ϕ is the viscous dissipation term defined as $\phi = \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top) : \nabla \boldsymbol{u}$. The solver is coupled with the fluid solver through the velocity field $\boldsymbol{u}$. In order to couple the solver in the other direction, one possibility is to add a buoyancy term in the Navier–Stokes equations based on the Boussinesq approximation as follows:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{F.2}$$

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla) \boldsymbol{u} + \nabla p^* - \nu \Delta \boldsymbol{u} = \underbrace{(1 - \beta(T - T_{\text{ref}}))\boldsymbol{g}}_{\text{buoyancy term}}, \tag{F.3}$$

where $\boldsymbol{g}$ is the gravitational force, β is the thermal expansion coefficient and T_{ref} is a reference temperature. This modified source term has an influence in some terms of the Jacobian and the residual of the Navier–Stokes equations. The implementation of the matrix-free solver already included a source term $\boldsymbol{f}$, e.g., used for the method of manufactured solutions. Therefore, the coupling only required the implementation of the source term. This coupling has been tested for a heated lid-driven cavity and for a two-dimensional Rayleigh–Bénard convection. While accurate results are obtained, further investigation of the efficiency of geometric multigrid preconditioner is required.