A Mixed Hybrid Finite Volumes Solver for Robust Primal and Adjoint CFD

by

Mattia Oriani

Submitted in partial fullfilment of the requirements for the degree of
Doctor of Philosophy

School Of Engineering And Materials Science
Queen Mary, University of London
United Kingdom

November 2017

Statement of Originality

I, Mattia Oriani, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature: Mattia Oriani
Date: 29th November 2017

To Erin Frances Kelly.

Abstract

In the context of gradient-based numerical optimisation, the adjoint method is an efficient way of computing the gradient of the cost function at a computational cost independent of the number of design parameters, which makes it a captivating option for industrial CFD applications involving costly primal solves. The method is however affected by instabilities, some of which are inherited from the primal solver, notably if the latter does not fully converge. The present work is an attempt at curbing primal solver limitations with the goal of indirectly alleviating adjoint robustness issues.

To that end, a novel discretisation scheme for the steady-state incompressible Navier-Stokes problem is proposed: Mixed Hybrid Finite Volumes (MHFV). The scheme draws inspiration from the family of Mimetic Finite Differences and Mixed Virtual Elements strategies, rid of some limitations and numerical artefacts typical of classical Finite Volumes which may hinder convergence properties. Derivation of MHFV operators is illustrated and each scheme is validated via manufactured solutions: first for pure anisotropic diffusion problems, then convection-diffusion-reaction and finally Navier-Stokes. Traditional and novel Navier-Stokes solution algorithms are also investigated, adapted to MHFV and compared in terms of performance.

The attention is then turned to the discrete adjoint Navier-Stokes system, which is assembled in an automated way following the principles of Equational Differentiation, i.e. the differentiation of the primal discrete equations themselves rather than the algorithm used to solve them. Practical/computational aspects of the assembly are discussed, then the adjoint gradient is validated and a few solution algorithms for the MHFV adjoint Navier-Stokes are proposed and tested. Finally, two examples of full shape optimisation procedures on internal flow test cases (S-bend and U-bend) are reported.

Acknowledgments

This work has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration through the "AboutFlow" project, under grant agreement number 317006.

I owe my deepest gratitude to my industrial supervisor, Dr. Guillaume Pierrot, whose invaluable help went far beyond his job description: the present work would not have been possible without his continued commitment, guidance, patience, enthusiasm and outstanding expertise on a vast array of topics, both related to CFD (adjoint method, numerical schemes) and software development (data structures, abstraction, code optimisation).

I am thankful to my academic supervisor, Dr. Jens-Dominik Müller, for giving me the opportunity and encouraging me to join the AboutFlow Initial Training Network: this not only culminated in the present work, but also effectively launched my current career as a CFD software developer.

I was lucky to work in close collaboration with the other Early Stage Researchers in the AboutFlow project, which led to several inspiring discussions and ideas, some of which are reflected in this thesis. I also thank my colleagues at ESI Group and in particular those who shared with me the experience of being industry-based PhD candidates: Athanasios Liatsikouras for his help with meshing, setting up and running of test cases and his cooperation regarding mesh morphing tools; Gabriel Fougeron for sharing his impressive mathematical knowledge and understanding of numerical methods; George Eleftheriou for his support on all technical issues.

A special mention goes to my friends in Paris, London and Milan for their amazing ability to put up with me during the most difficult times. I would like to thank my family, as well: my beloved parents, Anna and Leonardo; my sister Irene and her partner Carlo; my sister Chiara, her husband Garry and their little Erin, who was born shortly after the start of this project and has made everything better since.

Table of Contents

List of Figures

List of Tables

List of Symbols

Geometric and inertial quantities

d	number of spatial dimensions
$\|C\|$	volume of cell C
$\|F\|$	area of face F
$\vec{x}_C$	centroid of C
$\vec{x}_F$	centroid of F
s_{FC}	conventional cell-face ordering sign between C and F
$\vec{F}$	unsigned area vector of F
$\vec{F}_C$	area vector of F outward w.r.t. C
F_C^i	area vector of F outward w.r.t. C (i-th component)
h	mesh coarseness indicator (typically: maximum cell-to-cell centroid distance)

MHFV nomenclature and operators

Q^h	space of cell-averaged scalars
X^h	space of face fluxes
$\widehat{X^h}$	broken space of (one-sided) face fluxes
$(\cdot)_C$	degree of freedom for cell C
$(\cdot)_F$	degree of freedom for face F
$(\cdot)_{\partial C}$	mapping to the boundary faces of C (oriented accordingly where applicable)
$(\cdot)^{X^h}$	de Rham map in X^h
$(\cdot,\cdot)$	canonical dot product
$\langle\cdot,\cdot\rangle_{Q^h}$	scalar product in Q^h
$\langle\cdot,\cdot\rangle_{X^h}$	scalar product in X^h
$\mathbb{M}_C$	local inner product matrix
$\mathbb{N}_C$	local inner product matrix with convective term
$\nabla_C^{\mathcal{G}}$	Gauss gradient approximation on C

$\nabla_C^{\mathcal{L},\lambda}$	least-squares, λ_{FC}-weighted gradient approximation on C
$\nabla_C^{\mathcal{L},\mu}$	least-squares, $\mu_{C'C}$-weighted gradient approximation on C
$\mathcal{H}$	global scalar product operator
$\mathcal{K}$	diffusive flux operator
$\mathcal{D}$	divergence operator
$\mathcal{F}_{\mathbb{K}}$	hybrid anisotropic diffusion operator
$\mathcal{F}_{\mathbb{K},\vec{U},\eta}$	hybrid anisotropic convection-diffusion-reaction operator
$\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$	hybrid isotropic convection-diffusion operator (for momentum equation)
$\boldsymbol{\mathcal{G}}$	gradient operator (acting on cell-averaged pressure)
$\boldsymbol{\mathcal{D}}$	divergence operator (acting on hybrid velocity components)

MHFV variables and quantities

$\boldsymbol{\phi}$	cell-averaged generic scalar in Q^h
ϕ_C	cell-averaged generic scalar at cell C
$\widetilde{\boldsymbol{\phi}}$	hybrid generic scalar
ϕ_F	hybrid generic scalar at face F
$\mathbf{u}$	cell-averaged velocity in Q^h
$\mathbf{u}^i$	cell-averaged velocity in Q^h (i-th component)
$\vec{u}_C$	cell-averaged velocity at cell C
u_C^i	cell-averaged velocity at cell C (i-th component)
$\widetilde{\mathbf{u}}$	hybrid velocity
$\widetilde{\mathbf{u}}^i$	hybrid velocity (i-th component)
$\vec{u}_F$	hybrid velocity at face F
u_F^i	hybrid velocity at face F (i-th component)
$\mathbf{p}$	cell-averaged pressure in Q^h
p_C	cell-averaged pressure at cell C
$\overline{p}_{FC}$	pressure reconstructed at face F from cell C
$\mathbf{V}$	generic flux in X^h (diffusive, convective-diffusive or momentum)
V_{FC}	generic flux through F as seen from C
$\mathbf{U}$	convecting flux in X^h
U_{FC}	convecting flux through F as seen from C
U_{FC}^{uw}	upwind convecting flux through F as seen from C
U_{FC}^{dw}	downwind convecting flux through F as seen from C
$\boldsymbol{\eta}$	cell-averaged reaction coefficient in Q^h
η_C	cell-averaged reaction coefficient at cell C
$\mathbf{f}$	cell-averaged generic source term in Q^h
f_C	cell-averaged generic source term at cell C

$\widetilde{\mathbf{f}}$	hybrid generic right-hand side
g_C^i	cell-averaged i-th momentum source term at cell C
$\widetilde{\mathbf{g}}$	hybrid momentum right-hand side
$\widetilde{\mathbf{g}}^i$	hybrid momentum right-hand side (i-th component)
λ_{FC}	stabilisation weight for F relative to C
$\mathbb{K}_C$	cell-averaged diffusivity tensor at cell C
ν_C	cell-averaged kinematic viscosity at cell C
Pe_{FC}	local downwind Peclet number at F as seen from C
Re_F	local Reynolds number

Adjoint MHFV nomenclature

J	cost/objective function
$\mathcal{A}$	Navier-Stokes tangent matrix (Jacobian)
$\widetilde{\mathcal{F}}$	Jacobian momentum block
$\mathcal{M}$	generic mesh morphing operator
$\mathbf{m}$	generic mesh morphing right-hand side
$\mathbf{r}_{NS}$	Navier-Stokes residual
$\widetilde{\mathbf{r}_u}$	momentum residual
$\widetilde{\mathbf{r}_u}^i$	momentum residual (i-th component)
$\mathbf{r}_p$	continuity residual
$\mathbf{g}^*$	adjoint Navier-Stokes right-hand side
$\widetilde{\mathbf{g}_u}^*$	adjoint hybrid momentum right-hand side
$\mathbf{g}_p^*$	adjoint continuity right-hand side
$\widetilde{\mathbf{u}}^*$	adjoint hybrid velocity
$\mathbf{p}^*$	adjoint cell-averaged pressure
$\boldsymbol{\alpha}$	shape parameters
$\boldsymbol{\delta\alpha}$	displacement applied to shape parameters
$\mathbf{x}$	nodal coordinates
$\boldsymbol{\delta}\mathbf{x}$	displacement of nodal coordinates
$\mathbf{s}_A$	gradient

List of Abbreviations

AC	Artificial Compressibility
AD	Algorithmic/Automatic Differentiation
AL	Augmented Lagrangian
ATC	Adjoint Transpose Convection
BCPL	Block-Coupled
BJC	Cell-Based Barth-Jespersen
BJF	Face-Based Barth-Jespersen
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CFD	Computational Fluid Dynamics
DDM	Domain Decomposition Method
ED	Equational Differentiation
ENO	Essentially Non-Oscillatory
FD	Finite Differences
FE	Finite Elements
FPI	Fixed-Point Iteration
FTL	FORTRAN Template Library
FV	Finite Volumes
FVSG	Finite Volumes Scharfetter-Gummel
GD	Grad-Div
HFE	Hybrid Finite Elements
HMM	Hybrid Mimetic Mixed
HTHE	Hybrid θ-Scheme
HUPW1	Hybrid First-Order Upwind
HUPW2	Hybrid Second-Order Upwind
HYBC	Hybrid Centred
ILU	Incomplete Lower-Upper
LBB	Ladyshenskaya-Brezzi-Babuška

LCO	Limit Cycle Oscillation
LSC	Least-Squares Commutator
LSQ	Least-Squares
LSR	Linear Solver Replacement
MFD	Mimetic Finite Differences
MFE	Mixed Finite Elements
MFV	Mixed Finite Volumes
MHFV	Mixed Hybrid Finite Volumes
MINS	Minimal Symmetric
MIXC	Mixed Centred
MMS	Method of Manufactured Solutions
MPFA	Multipoint Flux Approximation
MVE	Mixed Virtual Elements
NOC	Non-Orthogonal Corrector
ORTN	Orthogonal Non-Symmetric
ORTS	Orthogonal Symmetric
OVRN	Over-Relaxed Non-Symmetric
OVRNA	Over-Relaxed Non-Symmetric with Anisotropy
OVRS	Over-Relaxed Symmetric
PCD	Pressure Convection-Diffusion
PDE	Partial Differential Equation
PRS1	First-Order Pressure Scheme
PRS2	Second-Order Pressure Scheme
RANS	Reynolds-Averaged Navier-Stokes
RMMM	Rigid Motion Mesh Morphing
SA	Spalart-Allmaras
SHCFP	Soft Handle CAD-Free Parametrisation
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SIMPLEC	SIMPLE-Consistent
SPD	Symmetric Positive-Definite
SUPG	Streamline-Upwind Petrov-Galerkin
TVD	Total Variation Diminishing
ULSQR	Upwind Least-Squares
VCPL	Velocity-Coupled
VNKC	Cell-Based Venkatakrishnan
VNKF	Face-Based Venkatakrishnan
WENO	Weighted Essentially Non-Oscillatory
WLSQR	Weighted Least-Squares

Chapter 1

Introduction

1.1 Context and motivation

Flow control, intended as an attempt to control the mechanical and/or the thermodynamic state of a fluid in order to achieve a desired purpose [112], has been a practice of man since ancient times: dams, sluices, canals, valves, ducts etc. are all examples of flow control. Applied to the modern field of aerodynamics, the primary role of flow control often becomes that of reducing the adverse effects of some undesirable physical phenomena connected with the flow of a fluid.

To provide a concrete example, consider aeronautical flows: here, the target is usually the reduction of the resistance to motion of an object moving in a fluid: the drag force F_D or, in non-dimensional terms, the drag coefficient c_D of an aircraft. Drag reduction not only reduces fuel burn (thus reducing direct operating costs), but also and most importantly reduces pollution and CO_2 and NOx emissions, which has been the focus of numerous international agreements as well as independent initiatives in the past few decades due the public's increased awareness of matters related to sustainability and eco-efficiency [76]. Noise reduction is also often mentioned in this context. Although improvement in general has been saturating since the beginning of the 21st century, until a "game-changer" is found (i.e. a somewhat "revolutionary" aircraft design) flow control technologies remain one of the most relevant research areas for aerospace. Examples include *passive* techniques (e.g. *natural laminar flow*, which aims at delaying transition to turbulent flow in the boundary layer by controlling the pressure gradient via appropriate aerofoil shaping; *vortex generators*, protrusions or bumps on the wing surface which deliberately trigger turbulence in an attempt to delay flow separation, hence reducing pressure drag), and *active* techniques, requiring an input of energy (e.g. *blowing-suction*

holes either injecting or sinking momentum in the boundary layer of an aerofoil, depending on whether laminar or turbulent flow is desired; *heating-cooling* at the aerofoil's surface, in order to either trigger or delay turbulence). Similar techniques have also been introduced in trades other than aerospace, such as automotive [142] or wind power [197].

Computer Aided Engineering (CAE) is today one of the primary tools employed in industrial design processes; in particular the automotive, aviation, space and shipbuilding industries extensively rely on *Computational Fluid Dynamics* (CFD) - focus of this thesis - to simulate and predict product performance. The popularity of CAE software, combined with the need to supplement the design chain with efficient, robust and possibly automated procedures towards improvements in flow control techniques, has given rise to the field of *numerical optimisation*.

Numerical optimisation has been expanding at an astonishing rate during the last few decades, in particular for CFD - although there is a constantly increasing emphasis on the interdisciplinary nature of the field [12]. The idea is to make use of CFD analysis within numerical procedures in order to achieve a desired optimal flow behaviour in the simulations and thus, assuming that the computational model is accurate enough, in the real product. More specifically, a numerical optimisation problem aims at determining the values of a set of *design parameters* which will optimise the efficiency of the item being designed in terms of one or more *objective* or *cost functions*, while respecting given constraints. For instance, for a wind turbine blade a possible formulation of the problem could be "for a given chord length, find the aerofoil profile that will minimise the drag whilst maintaining a given lift value", or "for a given aerofoil profile, find the twist of the blade that will maximise the power production of the turbine".

The way a numerical optimisation process interacts with CFD depends on the specific approach. So-called zero-order optimisation methods (including iterative stochastic strategies such as *Evolutionary Algorithms* [210]) only require evaluation of cost function values; they are global techniques, ideal in the early stages of the optimisation when the goal is to explore a large design space - wide ranges of parameter values and combinations - and identify promising areas. They require a large number of CAE simulations per cycle, hence their feasibility in industrial settings heavily depends on the computational cost of each run. Substantial costs in CPU time are a notorious drawback of industrial CFD; therefore, in this context, deterministic procedures and in particular *gradient-based* methods [12] are typically a more viable option. These algorithms make use of the gradient of the cost function with respect to the design parameters (*sensitivity*) to perform an iterative local search around a certain design state (see Section 2.1); they are particularly fitted to the later stages of the optimisation process, or when the starting point is known to be near-optimal.

In gradient-based methods, sensitivity computation for the current state is required at each optimisation cycle. Efficient and accurate sensitivity analysis is a non-trivial task that has driven research to produce a variety of approaches in the past three decades [189]. Amongst them the *adjoint method* (Section 2.2), pioneered by Pironneau [196] and Jameson [131], is arguably the best candidate in terms of CPU-time efficiency: it allows sensitivity computation at a cost independent of the number of parameters and proportional to the number of cost functions being considered - typically only a few, if not just one. The potential benefits of adjoint methodologies towards industrial design processes, which operate under tight constraints in terms of time and resources, are evident. As a consequence, governmental bodies have recently been receptive to a number of proposals for research projects intended to advance investigation on the topic whilst facilitating cooperation between industry and academia.

The present thesis was funded by one of such projects: AboutFlow[1], an Initial Training Network (ITN) - funded by the European Commission - aimed at tackling stability and robustness issues of adjoint solvers for industrial CFD. The project ran from November 2012 to October 2016, it appointed a total of fourteen Early Stage Researchers (ESR) across the EU, and it enlisted five academic and four industrial partners including ESI Group[2] (France), host institution for the research presented here. AboutFlow was preceded by the FlowHead[3] project (mainly focused on adjoint for automotive applications) and was followed by the similar IODA[4] (concerned with integration of CAD into automated, adjoint-driven optimisation processes).

1.2 Starting point and objectives

The critical point of the adjoint method lies in the numerical solution of the adjoint (dual) equation, a linear PDE similar in complexity to the governing flow equations themselves (*primal* - typically the Navier-Stokes equations). The continuous adjoint approach (Section 2.2.2) derives the adjoint PDE at the continuous level and then discretises it, while the discrete approach (Section 2.2.3) derives the adjoint problem directly from the CFD-discretised primal. Regardless of the approach, solving the adjoint Navier-Stokes is not straightforward and often suffers from stability issues. In the continuous case, discretisation schemes for the adjoint require their own stability analysis which is yet to be fully investigated [122, 181]. In the discrete case, stability of the adjoint is understood

[1] http://aboutflow.sems.qmul.ac.uk/
[2] https://www.esi-group.com/
[3] http://flowhead.sems.qmul.ac.uk/
[4] http://ioda.sems.qmul.ac.uk/

to be intimately connected with that of the primal. More specifically, when the primal is solved via a fixed-point iterative algorithm (FPI) and the iterator is inherited by its adjoint counterpart, convergence of the latter depends on the *contractivity* of the original FPI iterator: it will be shown in Section 2.4 how a stalling/stagnating primal solution, despite being able to produce acceptable CFD results, may lead to a diverging discrete adjoint.

Considerable progress has been made towards devising robust solution strategies capable of tackling unstable discrete adjoint systems [47, 75, 243]. The present thesis shares the same motivation, but follows a different (and complementary) route: the focus is placed instead on the spatial discretisation scheme of the primal itself, and in particular on those features that may hamper convergence. The approach is arguably more radical and less explored by adjoint communities, who generally limit themselves to acknowledging that, for complex cases, the necessity of full primal convergence places an unrealistic robustness requirement on conventional flow solvers [141]. The discretisation schemes found in industrial CFD codes often feature a number of numerical artefacts - implemented within classical methods such as Finite Volumes (FV) - intended to produce satisfactory results on models that are challenging in terms of physics and/or geometry; the introduction of such artefacts comes is some cases at the expense of an only partially converged solution. In Section 2.4, flux limiters and Non-Orthogonal Correctors (NOC) are reported as fitting examples of such artefacts. Hence the primary objective is to investigate alternative CFD discretisation schemes that are, as much as possible, consistent and stable without the need for additional numerical fabrications, under the hypothesis that this may indirectly alleviate convergence problems affecting the corresponding discrete adjoint.

To that end, the *Mixed Virtual Element* (MVE) method [15, 39] is identified as a promising candidate for a starting point (Section 3.2). MVE is the most recent evolution of *Mimetic Finite Differences* (MFD), a family of methods originally proposed by Brezzi, Lipnikov and Shashkov [42] towards the numerical solution of anisotropic diffusion problems featuring complex geometries and/or discontinuous material properties. MFD/MVE schemes for pure diffusion problems have been extensively investigated over the last two decades, while the development of convection-diffusion-reaction operators (especially for convection-dominated regimes) and Navier-Stokes is ongoing. A reference book on the topic was recently published by Beirão da Veiga et al. [20]. In the context of this thesis, attractive features of MVE-like schemes include their inherent consistency and stability which is independent of mesh "quality" in the FV sense (the mesh can indeed be composed of highly general polyhedral cells, including strongly non-orthogonal faces and non-convex shapes), as well as their fully implicit nature [16] which removes the need

for additional features such as the above-mentioned NOCs, thus facilitating convergence to stricter tolerances.

Less stringent requirements on mesh quality constitute a further benefit of MVE for CFD-based shape optimisation. A gradient-based shape optimisation process needs to update the shape of the computational boundary at each cycle. Rather than re-meshing - which is not trivial to automate - it is desirable to adapt the existing mesh to the new shape whilst respecting an *iso-connectivity* constraint, i.e. without altering the number of nodes, faces and cells or the incidence graphs amongst these, which also preserves consistency of the discrete adjoint sensitivity. The task is usually carried out by *mesh morphers* (see Section 6.3). Standard morphers may struggle to maintain FV-adequate mesh quality in later stages of the deformation process; in such circumstances the mesh-independent nature of MVE is advantageous. Antonietti et al. [7, 8] demonstrated this on simple cases of control and shape optimisation problems.

The main axe of research for the present thesis is identified based on these considerations: the primary objective is to derive a spatial discretisation strategy capable of tackling convection-dominated CFD problems of industrial interest whilst exhibiting some of the desirable traits typical of MVE schemes - consistency, stability, mesh-independence and the ability to converge fully. The *Mixed Hybrid Finite Volumes* (MHFV) scheme presented in the main body of this thesis (Chapters 3 through 5) is developed to that end. The work is limited to the steady-state, incompressible Navier-Stokes problem, in order to be able to engage with existing MVE literature and build upon it.

A secondary topic of interest concerns solution algorithms for incompressible Navier-Stokes. Industrial codes often rely on the so-called SIMPLE-type segregated algorithms [86, 229], developed specifically for Oseen-type problems such as Navier-Stokes (see Section 5.2.1). Such schemes are known to suffer from stalling [136, 158]. Modern computing capabilities allow to consider alternative Oseen preconditioning techniques that were previously deemed unfeasible, often because of excessive memory requirements. If these novel schemes exhibit better convergence behaviour, then they may have a direct positive impact on the discrete adjoint robustness issues discussed above. To that end, the present work outlines the adaptation of some existing algorithms to MHFV.

It is understood that, while the discrete adjoint context provides a motivation and starting point, the subjects treated in the core of this work are intended to move towards benefits that extend beyond the subject of adjoint-based numerical optimisation, since improvements on the stability, accuracy and performance of CFD solvers are desirable regardless of adjoint computation.

1.3 Plan of the thesis and contribution

An introduction to the main concepts of the adjoint method (discrete in particular) and the motivation for the remainder of the thesis are provided in Chapter 2. The MHFV anisotropic diffusion operator is derived following the principles of MVE in Chapter 3. MHFV is then extended to convection-diffusion-reaction problems in Chapter 4. The MHFV steady-state incompressible Navier-Stokes operator is derived in Chapter 5. In Chapter 6, the attention turns to describing and validating the specific methodology deployed for assembling and solving the MHFV discrete adjoint Navier-Stokes. Chapter 7 illustrates practical examples of full optimisation processes using the MHFV primal and adjoint solvers. Chapter 8 draws conclusions and outlines areas with potential for future work.

As mentioned, the main contribution of this thesis consists in assessing how an improvement on the primal CFD scheme - namely from classical FV to a MVE-like strategy - impacts the behaviour of its corresponding discrete adjoint. The present work includes incremental research upon previous literature along several axes, summarised as follows:

- In Chapter 2 the concept of *Equational Differentiation* (ED) is introduced. Although not novel per se, ED formalises a specific way of deriving discrete adjoints which is functional in the investigation on solution algorithms carried out in Chapter 6.

- In Chapter 3, an original comparison is drawn between MVE and classical FV and exploited to suggest a number of novel choices for the weighting coefficients appearing in the MHFV flux stabilisation term (Section 3.3.4). The scheme is validated and an experimental comparison is drawn among the various weight expressions.

- In Chapter 4, the original scheme by Droniou [70] for convection-dominated regimes is extended to second-order accuracy. A number of FV and FE-inspired stabilisation strategies are then adapted to the MHFV framework - *flux limiters*, *Streamline-Upwind Petrov-Galerkin* (SUPG), *Weighted Least-Squares* (WLSQR) - and a novel one specific to MHFV is introduced: *Upwind Least Squares* (ULSQR, Section 4.2.5). Validation and comparisons are carried out on purposely designed test cases.

- In Chapter 5, the Navier-Stokes scheme by Droniou and Eymard [72] is extended to second-order accuracy for both velocity and pressure variables. Existing solution algorithms are then adapted to the scheme (Section 5.2), including a MHFV-specific version of the traditional SIMPLEC and the novel Augmented Lagrangian (AL) preconditioner. Validation and performance are assessed against benchmark cases.

- In Chapter 6, the solution algorithms devised for the MHFV primal are adapted to the adjoint system, and their performance is compared.

The MHFV solver developed for this thesis was coded in ESI Group's in-house *FORTRAN Template Library* (FTL) framework [193], a toolbox which supplements the FORTRAN language with object-oriented and templating capabilities. The project is currently known internally by the working title *MimFlow*.

Chapter 2

The Adjoint Method

2.1 Numerical optimisation

Optimisation processes - as described in the introduction - fall into the category of *minimisation problems*, where one seeks to find the minimum of a given function within a certain design space. An optimisation problem in the context of aerodynamics/fluid dynamics involves:

- an *objective* or *cost function* J (e.g. drag force), the function to be minimised;
- a set of *design* or *control variables* $\boldsymbol{\alpha}$ (which e.g. control size and shape of a turbine blade), representing those parameters that can be controlled directly in the design process;
- a set of *state variables* W (e.g. velocity, pressure, density);
- a set of *governing equations* (e.g. the Navier-Stokes equations), relating the value of state variables to that of control variables;
- a set of *constraints* (e.g. minimum thickness of the blade, fixed lift value, maximum chord length).

The present work will deal with *shape optimisation* problems, where the design variables define or control the shape of the object being designed; therefore, the terms "design/control variables" and "shape parameters" will occasionally be used interchangeably.

There is a wide range of numerical methods aimed at the solution of minimisation

problems. Amongst them are stochastic methods such as *Evolutionary* or *Genetic Algorithms* [210]: these, in analogy with the Darwinian evolutionary model, start by generating several random combinations of design variables, i.e. a generation of individuals, then select the "fittest" ones - those which, upon numerical solution of the governing equations, produce lower values for the cost function - and mutate and cross-breed them in an attempt to evolve, generation after generation, towards the optimal configuration. Such methods however, because of their nature, typically require a large number of evaluations per generation - easily in the order of thousands for a case with as few as 10-15 design variables; the number increases very rapidly with the size of the design space. This is ultimately unfeasible in most CFD applications, where each evaluation of the cost function requires a full flow solve and, for industrial cases, each of the flow solves typically entails a considerable computational expense [167].

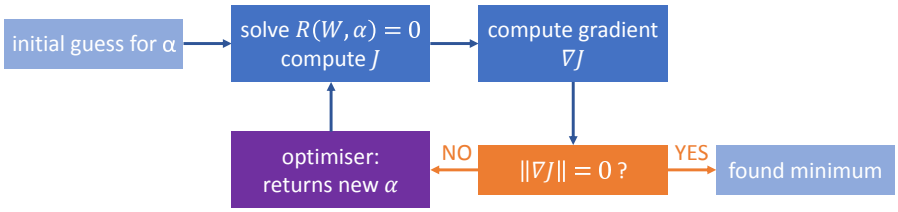


Figure 2.1: Basic flowchart of a generic gradient-based optimisation loop.

Therefore, in the context of industrial CFD optimisation, *gradient-based* algorithms are often preferred. The idea is to start with an initial configuration of design variables $\boldsymbol{\alpha}$ and, at each cycle, compute $\nabla J = \frac{dJ}{d\boldsymbol{\alpha}}$, i.e. the gradient or *sensitivity* of the cost function, in order to iteratively reach a local minimum of J where $\|\nabla J\| = 0$. The process is schematically represented in Figure 2.1. Unlike stochastic methods, gradient-based optimisation only traces a one-dimensional path along a descent direction. If the "landscape" (i.e. the gradient field) is smooth and isotropic, convergence will be independent of the number of design variables. In practice, many fine design spaces do lead to anisotropic gradient fields and non-smooth behaviour. An appropriate choice of parametrisation, i.e. a selection of control variables leading to a reasonably smooth gradient field, is then important (but this topic will not be discussed in this thesis).

Bartholomew-Biggs [12] reports the following examples of popular gradient-based methods:

- *Steepest descent method*: at each design iteration n the gradient ∇J_n is computed, and $p_n = -\nabla J_n$ is taken as the direction which causes J to decrease most rapidly; a *line search* is performed to find the minimum of J along p_n, then the process is repeated until $\|\nabla J\|$ is sufficiently small. The method is intuitive and easy to implement, but convergence is slow.

- *Newton method*: J is approximated locally at α_n as a quadratic function via a truncated Taylor series:

$$J(\alpha_n + s_n) \approx J(\alpha_n) + s_n^T g_n + \frac{1}{2}\left(s_n^T H_n s_n\right)$$

  where s_n is an arbitrary step in the design space, $g_n = \nabla J_n$ is the gradient vector and H_n the Hessian matrix; the direction minimising such a model is thus $p_n = -H_n^{-1} g_n$. A line search is performed along p_n (often inexact, i.e. not finding the actual minimum but ensuring sufficient reduction of J), then the process is repeated until $\|g_n\|$ falls below a set tolerance. The method exhibits quadratic convergence; however computation of the Hessian and computation of p_n are rather expensive. Besides, the method needs *safeguarding* where H_n is not positive-definite, or else might converge to a maximum or even diverge.

- *Quasi-Newton methods*: these are based on the same quadratic approximation as the Newton method; however, rather than requiring computation of the Hessian, they construct an approximation of it (or of its inverse directly) and additionally enforce its positive-definiteness. The advantage is the elimination of Hessian computation; convergence is found to be super-linear, but slower than that of the Newton method.

Clearly the blocking factor of any gradient-based algorithm is the gradient computation itself, which has to be performed at each iteration. Arguably the most straightforward method is via *Finite Differences* (FD), which allows to compute an approximate directional derivative of J by introducing a perturbation $\boldsymbol{\delta\alpha}_k$ in the design variables:

$$\frac{\partial J}{\partial \boldsymbol{\alpha}_k} \approx \frac{J(\boldsymbol{\alpha} + \boldsymbol{\delta\alpha}_k) - J(\boldsymbol{\alpha})}{\boldsymbol{\delta\alpha}_k} \ .$$

FD-based gradient computation is however unfeasible in an industrial CFD context for the following reason: in order to assemble the full gradient ∇J, one must compute as many directional derivatives as the number of design variables - each stemming from a perturbation in direction of one design variable only. Therefore, the assembly of the full gradient requires computing $J(\boldsymbol{\alpha} + \boldsymbol{\delta\alpha}_k)$, and thus solving the governing equations, for as many times as the number of design variables. In a situation involving a high number of parameters (in the order of thousands or greater) and generally very expensive CFD solves, the computational cost of the FD approach quickly becomes prohibitive.

The FD method also suffers from the well-known issue of having to choose an appropriate step-length $\boldsymbol{\delta\alpha}_k$: if chosen too large, truncation error will incur; if chosen too small, round-off error will incur. The *tangent linearisation* approach [167] provides a

solution to this problem - i.e. it allows to compute exact directional derivatives - but still scales with the number of design variables and requires as many extra solves.

Conversely the *adjoint method*, described in the following section, allows computing all components of the sensitivity vector ∇J at a cost that is essentially independent of the number of design variables, and roughly equivalent to that of one extra CFD solve.

2.2 Adjoint-based sensitivity analysis

Adjoint equations for fluid dynamics have been investigated since as early as the 1970's (see e.g. Pironneau [196]) and subsequently pioneered throughout the 1980's and 1990's, most prominently by Jameson [131, 132] who, in cooperation with others, applied the method to optimal control theory and shape optimisation combined with the Euler [201] and Navier-Stokes [134] equations, developing what is known today as the continuous adjoint approach (see Section 2.2.2). The adjoint method proved itself to be an incredibly powerful tool since its early stages, as testified by numerous examples of cost-effective gradient computation for shape optimisation of complex structures for aerodynamics, such as 2D aerofoils [201], 3D aircraft wings [134] or even complete 3D aircraft configurations [202]. More recently, interesting developments have also surfaced for automotive applications, which up until recent years were lagging behind; notable examples of industrial relevance include shape optimisation of individual car components such as exhaust systems [122], external car aerodynamics [181, 190] and noise reduction [185]. Adjoint-based shape optimisation for turbo-machinery applications [228, 242] is also becoming increasingly popular.

2.2.1 State-of-the-art industrial adjoint CFD

The adjoint approach is nowadays acknowledged as a powerful innovation in optimisation processes. Considerable efforts have been made in recent years to improve its feasibility when it comes to full-sized, complex industrial cases and improve the robustness of the method, to a point where its inclusion as a standard feature in commercial CFD solvers and its systematic usage for industrial optimisation cycles is now a reality.

While showing continued interest in academic research advances, the CFD industry has meanwhile taken steps to be able to provide a first generation of mainstreamed, user-friendly adjoint solvers. Two prominent examples are ANSYS, who currently include

adjoint capabilities in standard releases of their flagship CFD solver Fluent[1], and CD-adapco (recently acquired by Siemens[2]), who offer similar features in their multi-physics tool STAR-CCM+[3]; both of these are based on the discrete adjoint approach (see Section 2.2.3). On the other hand, a continuous adjoint solver is included in the standard release of the popular open-source toolbox OpenFOAM[4]. Several other OpenFOAM-based continuous adjoint implementations exist, e.g. the one currently maintained by ENGYS[5]. A high-performing discrete adjoint based on the Auotmatic Differentiation (AD, see Section 2.3.2) of a subset of OpenFOAM has also been developed at RWTH University, Aachen [223, 224].

There also exist several CAE software companies that, while not yet including adjoint capabilities in their customer releases, do possess in-house tools which allow them to perform adjoint computations as consultants. ESI Group - host institution for the research presented here - developed the i-Adjoint library [193, 220]: an independent tool capable of automatically generating a discrete adjoint of a CFD solver with minimal code intrusion (through a Finite Differencing-based reverse assembly, see Section 6.1.4), provided that certain user subroutines are available. i-Adjoint in particular is the tool used for the developments presented in this work, and its functionalities shall be described in more detail in Chapter 6. Finally, there are also instances of adjoint codes developed by research partners in conjunction with manufacturing companies, if not directly by their own R&D departments, and integrated with their in-house CFD solvers; an example is the HYDRA[6] code (discrete, AD-based), extensively employed by Rolls-Royce[7] in particular for turbomachinery applications.

2.2.2 Formulation of the continuous adjoint equations

The earliest derivation of the adjoint approach, outlined below, was formulated via a Lagrange multiplier argument [131, 196] operating directly on the original continuous problem before it is discretised. This is today referred to as *continuous adjoint*; Pironneau [196], Jameson [131, 134], Baysal and Eleshaky [14], Anderson and Venkatakrishnan [5], are early examples of the derivation of the continuous adjoint flow equations (Euler and Navier-Stokes in particular).

[1] http://www.ansys.com/products/fluids/ansys-fluent
[2] https://www.siemens.com/
[3] https://mdx.plm.automation.siemens.com/star-ccm-plus
[4] https://www.openfoam.com/
[5] https://engys.com/
[6] https://www.mpls.ox.ac.uk/research-section/the-hydra-code-rolls-royces-standard-aerodynamic-design-tool
[7] https://www.rolls-royce.com/

The cost function $J = J(W(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ is typically a scalar quantity that depends directly on both the design variables $\boldsymbol{\alpha}$ and the state variables W (also referred to as flow variables in a CFD context), the latter being themselves dependent on $\boldsymbol{\alpha}$ through the governing equations of the problem. A shape perturbation $\boldsymbol{\delta\alpha}$ results in a change in J as:

$$\delta J = \frac{\partial J}{\partial W}\delta W + \frac{\partial J}{\partial \boldsymbol{\alpha}}\boldsymbol{\delta\alpha} \tag{2.1}$$

where the first term represents the contribution caused by changes in the flow field, and the second is the change associated directly with the shape modification $\boldsymbol{\delta\alpha}$. According to the principles of control theory, the governing equations of the flow are considered as a constraint. The equations are symbolically written as

$$R(W(\boldsymbol{\alpha}), \boldsymbol{\alpha}) = 0 \tag{2.2}$$

to highlight the dependence of W and $\boldsymbol{\alpha}$ within the flow domain. Expression (2.2) is referred to as the *primal* problem, and it represents the set of partial differential equations (PDEs) modelling the flow behaviour (e.g. the Euler or Navier-Stokes equations). Since (2.2) is to be satisfied for any admissible configuration of $\boldsymbol{\alpha}$ it follows that, linearising the constraint about a certain state, it always holds that:

$$\delta R = \frac{\partial R}{\partial W}\delta W + \frac{\partial R}{\partial \boldsymbol{\alpha}}\boldsymbol{\delta\alpha} = 0 \ . \tag{2.3}$$

Adding this constraint to (2.1) via a Lagrange multiplier W^*, the change in cost function becomes

$$\begin{aligned} \delta J &= \frac{\partial J}{\partial W}\delta W + \frac{\partial J}{\partial \boldsymbol{\alpha}}\boldsymbol{\delta\alpha} - W^{*T}\left(\frac{\partial R}{\partial W}\delta W + \frac{\partial R}{\partial \boldsymbol{\alpha}}\boldsymbol{\delta\alpha}\right) \\ &= \left(\frac{\partial J}{\partial W} - W^{*T}\frac{\partial R}{\partial W}\right)\delta W + \left(\frac{\partial J}{\partial \boldsymbol{\alpha}} - W^{*T}\frac{\partial R}{\partial \boldsymbol{\alpha}}\right)\boldsymbol{\delta\alpha} \ . \end{aligned} \tag{2.4}$$

It is now easily verified that, choosing W^* such that it satisfies the *dual* or *adjoint* problem

$$\left(\frac{\partial R}{\partial W}\right)^T W^* = \left(\frac{\partial J}{\partial W}\right)^T \ , \tag{2.5}$$

the first term in (2.4) vanishes. Hence the change in objective can be expressed as

$$\delta J = \left(\frac{\partial J}{\partial \alpha} - W^{*T}\frac{\partial R}{\partial \boldsymbol{\alpha}}\right)\boldsymbol{\delta\alpha} \tag{2.6}$$

and, taking the limit for $\boldsymbol{\delta\alpha} \to 0$, one obtains the sensitivity of J, i.e. its gradient with respect to design variables $\boldsymbol{\alpha}$, required by each optimisation cycle of most gradient-based algorithms. The approach allows gradient computation at the extra cost of discretising and solving the adjoint problem (2.5), which is a set of PDEs similar in form and

complexity to the primal problem. The advantage is that (2.6) is independent of the bothersome term δW, the computation of which would require as many extra primal solves as the number of design variables, if one were to opt for e.g. a FD approach. This is where the power of adjoint methods becomes evident: sensitivity computation comes at a computational cost comparable to that of a flow field evaluation and, as anticipated, it is made essentially independent of the number of design parameters - barring the computation of $\frac{\partial J}{\partial \boldsymbol{\alpha}}$ and $\frac{\partial R}{\partial \boldsymbol{\alpha}}$, the cost if which is negligible compared to that of a primal solve. This is paramount in aerodynamic shape optimisation problems of industrial interest, where a primal run can incur a significant computational cost and the number of design parameters can be in the order of millions (potentially, each surface node can be a design parameter); without the adjoint approach, gradient-based optimisation in such cases would be practically unfeasible.

2.2.3 Discrete adjoint via linear algebra

In contrast to the continuous method, a second approach known as *discrete adjoint* starts by considering the primal as the set of discrete governing equations, rather than the original PDE. The Lagrange multipliers approach from the previous section is still applicable in this case. Alternatively, a discrete adjoint may be derived via a linear algebraic procedure, as presented by e.g. Giles and Pierce [101] and Müller [167], which is arguably more compact/straightforward to develop; the procedure is outlined below. In order not to confuse this specific context with the continuous formulation of Section 2.2.2, it is worth introducing here a different, more specific notation for the discrete primal problem:

$$\mathbf{r}\left(\mathbf{w}\left(\boldsymbol{\alpha}\right), \boldsymbol{\alpha}\right) = \mathbf{0} \ . \tag{2.7}$$

Here, $\boldsymbol{\alpha}$ represents a finite set of n_α design parameters and $\mathbf{w}$ the vector of degrees of freedom of the discrete primal, i.e. the discrete flow field. For shape optimisation, the choice of $\boldsymbol{\alpha}$ could fall in principle on the coordinates of all mesh nodes lying on the surface to be optimised, which is sometimes referred to as the *infinite-dimensional problem* [167]. Such a choice allows each surface node to move independently thus maintaining the richest possible design space, but it is likely to violate practical feasibility of the final shape and/or aesthetic constraints as the optimisation progresses. It is therefore common practice to select shape parameters differently, e.g. as a set of "handle" nodes that control surface deformation whilst enforcing a certain degree of smoothness [148], or as CAD-based parameters [115, 172, 204, 241, 245].

Assuming that the problem at hand counts a total of n_w degrees of freedom for the discrete state variables, then $\mathbf{w}$ is a vector of size n_w belonging to the finite space in

which the discrete state variables are defined, and $\mathbf{r}$ stands for the n_w-sized residual vector resulting from the operator - typically non-linear for CFD - arising from the discretisation of the flow equations. In other words, (2.7) represents the residual vector that a CFD solver drives to zero when solving for a steady-state $\mathbf{w}$. As expressed in (2.7), $\mathbf{r}$ depends on the discrete flow field $\mathbf{w}$ as well as on the set of n_α shape parameters $\boldsymbol{\alpha}$. Once again, according to the principles of control theory, the discrete primal (2.7) acts as a constraint to be satisfied regardless of the value of $\boldsymbol{\alpha}$, a constraint that can be written in linearised form as

$$\frac{\partial \mathbf{r}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \boldsymbol{\alpha}} = -\frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}} \tag{2.8}$$

or, in compact notation:

$$\mathbb{A}\mathbb{V} = \mathbb{F} \ . \tag{2.9}$$

On the left-hand side of (2.9) one can identify the $n_w \times n_w$ matrix $\mathbb{A} = \frac{\partial \mathbf{r}}{\partial \mathbf{w}}$ as the Jacobian of the primal system about a converged state $\mathbf{w}$ satisfying (2.7), and the $n_w \times n_\alpha$ matrix $\mathbb{V} = \frac{\partial \mathbf{w}}{\partial \boldsymbol{\alpha}}$ representing the change in each degree of freedom of $\mathbf{w}$ caused by a change in each shape parameter in $\boldsymbol{\alpha}$. On the right-hand side, the $n_w \times n_\alpha$ matrix $\mathbb{F} = -\frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}}$ holds the (negative) partial derivatives of the residual vector with respect to a shape change, i.e. the direct dependency of $\mathbf{r}$ on $\boldsymbol{\alpha}$. Introducing J as the discrete cost function

$$J = J\left(\mathbf{w}\left(\boldsymbol{\alpha}\right), \boldsymbol{\alpha}\right) \ , \tag{2.10}$$

its sensitivity with respect to $\boldsymbol{\alpha}$:

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \frac{\partial J}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \boldsymbol{\alpha}} \tag{2.11}$$

can be rewritten as

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \mathbf{g}^T \mathbb{V} \tag{2.12}$$

where $\mathbf{g} = \left(\frac{\partial J}{\partial \mathbf{w}}\right)^T$ is the (transposed) vector of partial derivatives of J with respect to the discrete flow variables $\mathbf{w}$. The dual (adjoint) problem is then introduced; in a discrete framework, it takes directly the form of a linear system:

$$\begin{array}{ccccc} \left(\dfrac{\partial \mathbf{r}}{\partial \mathbf{w}}\right)^T & \mathbf{w}^* & = & \left(\dfrac{\partial J}{\partial \mathbf{w}}\right)^T & \text{i.e.} \\ \mathbb{A}^T & \mathbf{w}^* & = & \mathbf{g} & \end{array} \tag{2.13}$$

with $\mathbf{w}^*$ being the discrete adjoint field. If a $\mathbf{w}^*$ is found that satisfies (2.13), then the following equivalence:

$$\mathbf{g}^T \mathbb{V} = \left(\mathbb{A}^T \mathbf{w}^*\right)^T \mathbb{V} = \left(\mathbf{w}^*\right)^T \mathbb{A}\mathbb{V} = \left(\mathbf{w}^*\right)^T \mathbb{F} \tag{2.14}$$

implies that the sensitivity vector (2.12) can alternatively be computed as

$$\frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + (\mathbf{w}^*)^T \mathbb{F} \ . \tag{2.15}$$

The advantage of using the dual formulation becomes clear when comparing the two sensitivity expressions (2.12) and (2.15). The former requires knowledge of matrix $\mathbb{V} = \frac{\partial \mathbf{w}}{\partial \boldsymbol{\alpha}}$, which must be determined such that constraint (2.9) is satisfied; in practice, this means solving n_α linear systems, with each solution providing a column of $\mathbb{V}$. The adjoint sensitivity, on the other hand, only requires knowledge of the adjoint flow field $\mathbf{w}^*$ (which is obtained by solving the adjoint system (2.13)) and the assembly of terms $\frac{\partial J}{\partial \boldsymbol{\alpha}}$ and $(\mathbf{w}^*)^T \mathbb{F} = -(\mathbf{w}^*)^T \frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}}$ which, as mentioned in the continuous scenario, is inexpensive compared to a primal solve. Therefore, the gradient of J is obtained at the cost of solving one linear system of the same size and similar complexity as the primal, independently of the number n_α of shape parameters.

The discrete adjoint as outlined above assumes the presence of only one cost function J, and therefore only one gradient to be computed. This assumption simplifies the argument and reflects most real-life optimisation problems, where typically the objective is to either minimise one quantity (such as drag, pressure drop, etc.) or, in the case of multi-objective optimisation, a combination of two or more which, in practice, reverts to minimising a single cost function. This is however, from a mathematical viewpoint, a specific case of a more general formulation involving n_J cost functions. If the goal was to compute gradients of each J separately, then the sensitivity $\frac{dJ}{d\boldsymbol{\alpha}}$ and the term $\mathbf{g}$ in (2.12) would algebraically be represented by $n_J \times n_\alpha$ matrices. This would give rise in (2.13) to n_J adjoint systems, to be solved for n_J adjoint fields $\mathbf{w}^*$ (i.e. one adjoint problem per cost function). In the extreme case of a problem featuring several cost functions and only one shape parameter, expression (2.9) would reduce to a single linear system to be solved for $\mathbb{V}$ (now a vector), and subsequently all n_J sensitivities could be computed at once via (2.12) at a cost essentially independent of n_J itself. Such an approach (*tangent linearisation*) would therefore be more advantageous when dealing with a case with fewer design parameters than cost functions ($n_\alpha \ll n_J$). However, as stated above, in a typical CFD shape optimisation application it is found that $n_\alpha \gg n_J$: typically one cost function only, against a large number of shape parameters; in the infinite-dimensional case, where $\boldsymbol{\alpha}$ represents the coordinates of all surface nodes, n_α could be in the order of millions. This highlights once again the superiority of the dual/adjoint approach and its potential in flow optimisation problems of industrial interest.

2.2.4 Continuous vs. discrete adjoint

As mentioned, the continuous adjoint approach operates at the level of the original PDE and yields an adjoint equation (2.5) also in the form of a set of PDEs which requires a suitable discretisation in order to be solved numerically. Conversely, a discrete adjoint starts with the already discretised primal and leads directly to the formulation of the discrete adjoint problem in the form of a linear system (2.13). Hence the continuous adjoint follows a *differentiate-then-discretise* path, while the discrete adjoint follows *discretise-then-differentiate*, as shown in Figure 2.2.

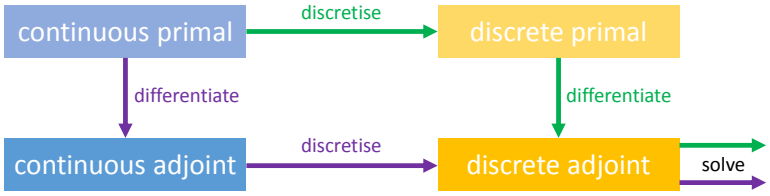


Figure 2.2: Different paths to adjoint derivation: continuous (purple) and discrete (green).

Both paths ultimately produce a linear discrete adjoint operator, but they do not commute: the produced adjoint system will be different, and so will be the sensitivity vector. In terms of gradients, the difference between the two can be expressed as stated by Nadarajah [174]: a continuous adjoint will compute the inexact gradient of the exact cost function, while a discrete adjoint will produce the exact gradient of the inexact cost function. To clarify: the continuous approach relies on a formulation of the dual problem (2.5) in the form of a set of PDEs, which in turn leads to the exact continuous expression of the adjoint sensitivity (2.6) (under the assumption that the cost function J is indeed continuously differentiable with respect to $\boldsymbol{\alpha}$). The adjoint problem itself takes a form similar to that of the primal, meaning that if the primal is a set of PDEs for which the analytical solution cannot be found, the same will be true for its adjoint counterpart. The adjoint problem will thus have to be discretised and solved numerically, typically via discrete operators and solution algorithms similar to those used for the primal; this introduces a discretisation (truncation) error in the computed adjoint field W^*, and this error is ultimately reflected in the gradient computed via (2.6). Conversely, the discrete approach leads to an adjoint formulation which is already in a discrete form - the linear system (2.13). Assuming that this system can be solved down to an arbitrarily small tolerance, then the sensitivity computed via (2.15) will be the exact gradient of the inexact (discrete) cost function J (2.10).

Either approach has its relative merits and drawbacks, extensively discussed in the literature [100, 167, 173, 174]. Despite the dichotomy being strongly marked in the adjoint

community, attempts have been made to bridge the gap between continuous and discrete adjoint by showing theoretical asymptotic equivalence between the two on infinitely refined meshes, at least for certain CFD discretisation schemes [133, 194]; continuous-discrete hybrid approaches have also been developed [217]. In terms of implementation a continuous adjoint seems to be a favourable choice since, once the dual problem is formulated at a PDE level, a well modularised CFD library will allow a developer to write an adjoint solver with relative ease by recycling data structures, discrete operators and solution algorithms used in the primal solver. There is of course the caveat that, beforehand, one must analytically derive not only the dual PDE but also the dual boundary conditions, as shown by e.g. Giles et al. [100]. While the duals of the most common continuous operators and boundary condition types do not pose an issue and their derivation can be found in the literature, this is not the case when e.g. a non-standard operator or new boundary condition type is added to the primal solver and thus its continuous adjoint counterpart must also be derived, which requires some lengthy and error-prone preliminary work. A typical example in CFD is the implementation of turbulence models for Reynolds-Averaged Navier-Stokes (RANS), which usually entails the addition of extra PDEs to the problem and potentially new types of boundary conditions as well, such as when a wall function approach is employed. Considerable work in this sense was carried out recently notably by Giannakoglou's research team at the National Technical University of Athens (NTUA) [184, 186].

When dealing with discrete adjoints, on the other hand, deriving the Jacobian of a discrete operator is arguably more straightforward than a continuous one and is attainable even by the less analytically skilled developers, since it often just requires familiarity with derivative expressions of basic mathematical operations; it is however easily as error-prone (if not more) than the continuous case. Hand-derivation does require a full thorough knowledge of the discrete operators featured in the primal solver; there are however alternative ways of assembling the adjoint system (2.13), such as Finite Differences or Algorithmic/Automatic Differentiation (see Section 2.3), which do not require any hand-derivation at all and can be automated.

Some other practical aspects to be taken into account when comparing continuous and discrete adjoint are:

- Code maintenance: the continuous approach presents considerable difficulties in this sense. Each time a new feature - a new boundary condition, a new model, a new equation, etc. - is added to the primal, the adjoint solver has to be manually updated after deriving and discretising the continuous adjoint of such feature. Similar issues arise with a hand-derived discrete adjoint with the addition that, in this case, modifications of the discretisation scheme itself must be taken into account as well,

lest the adjoint code no longer produce derivative values consistent with the primal. An automatically generated discrete adjoint, however, is by definition not affected by this. It should be stressed from now that "automated" does not necessarily imply the use of Automatic Differentiation tools (described in Section 2.3.2), but rather any process that aims at assembling the discrete adjoint system without resorting to hand-derivation - hence without explicit knowledge of primal operators.

- Boundary conditions: in a continuous context, the adjoint of each type of boundary condition must be derived by hand. In a discrete context, treating boundary conditions is arguably more straightforward in the sense that, once applied in the primal system, their consistent adjoint counterparts will naturally appear in the Jacobian - and therefore in the adjoint system - and, in principle, no specific treatment is required. Some literature [98, 167] however does discuss the treatment of hard Dirichlet boundary conditions - the case of boundary values being enforced directly on a subset of the degrees of freedom of the solution field, as in CFD node-based or face-based solvers. In this scenario, the degrees of freedom corresponding to Dirichlet boundaries are effectively eliminated from the problem and the right-hand side augmented accordingly. If the primal solver takes care of the extraction, namely by zeroing all matrix entries linking any other row of the residual expression to a Dirichlet boundary, then it will be reflected in the adjoint system. If not, then special care must be taken: at the boundary, the adjoint field takes value $\frac{\partial J}{\partial \mathbf{w}_D}$ ($\mathbf{w}_D$ being the imposed value), which may scale very differently with respect to the internal adjoint field and hence produce strong oscillations; in turn, convergence may be impaired or even inhibited. Giles et al. [98] provide guidelines for treating such cases.

2.2.5 Physical interpretation of adjoint fields

It has been pointed out [101, 167] how adjoint fields lend themselves to a physical interpretation. This is hereby illustrated in the discrete adjoint case, but the same reasoning can be extended to the continuous adjoint. Let a source term $\mathbf{t}$ be added to the discrete primal (2.7):

$$\mathbf{r}\left(\mathbf{w}\left(\boldsymbol{\alpha}\right), \boldsymbol{\alpha}\right) = \mathbf{t} \ . \tag{2.16}$$

Differentiating the cost function J with respect to $\mathbf{t}$ gives

$$\frac{dJ}{d\mathbf{t}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} \underbrace{\frac{\partial \boldsymbol{\alpha}}{\partial \mathbf{t}}}_{=0} + \frac{\partial J}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \mathbf{t}} \tag{2.17}$$

where the first term is null since the parameters $\boldsymbol{\alpha}$ do not depend on $\mathbf{t}$. Similarly, differentiating the primal residual expression (2.16) leads to

$$\frac{\partial \mathbf{r}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \mathbf{t}} + \frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}} \underbrace{\frac{\partial \boldsymbol{\alpha}}{\partial \mathbf{t}}}_{=0} = \mathbb{I} \,, \tag{2.18}$$

(where $\mathbb{I}$ is the $n_w \times n_w$ identity matrix), i.e.

$$\frac{\partial \mathbf{w}}{\partial \mathbf{t}} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{w}} \right)^{-1} \,. \tag{2.19}$$

Replacing now (2.19) in (2.17) and transposing yields

$$\left(\frac{dJ}{d\mathbf{t}} \right)^T = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{w}} \right)^{-T} \left(\frac{\partial J}{\partial \mathbf{w}} \right)^T \tag{2.20}$$

and a comparison with the adjoint system (2.13) leads to the conclusion:

$$\mathbf{w}^* = \left(\frac{dJ}{d\mathbf{t}} \right)^T \,. \tag{2.21}$$

The adjoint field $\mathbf{w}^*$ thus quantifies the (first order) response of J to the introduction of $\mathbf{t}$ in the primal. It reveals how an infinitesimally small source term in the governing equations is transferred to an infinitesimal change in the cost function J, indicating the direction of the locally optimal source term that should be added to the primal in order to reduce the value of J. The physical interpretation of the perturbation is dictated by the nature of the primal equation being considered: for instance, for the momentum equation it will be dimensionally consistent with a force, and the resulting adjoint velocity field will provide the (negative) direction of the optimal external force that should be exerted on the flow to reduce J.

Figure 2.3 is an example of a drag-adjoint velocity field $\vec{U}^*$ (magnitude contour) around a cylinder immersed in a low-Re flow field, constrained via the incompressible Navier-Stokes equations: locations where the magnitude of $\vec{U}^*$ is larger correspond to locations where the introduction of an infinitesimal perturbation (i.e. an impulse introduced as source term) will result in greater changes in terms of drag; the direction of such perturbation will be determined by the direction of $\vec{U}^*$ itself, not shown in figure.

Therefore, by observing an adjoint field one can gain some physical insight into the dynamics of a flow that might be difficult to learn from observing the flow itself. However, from an engineering viewpoint the adjoint state on its own is only of limited relevance; a designer will typically be interested in computing the sensitivity of the cost function

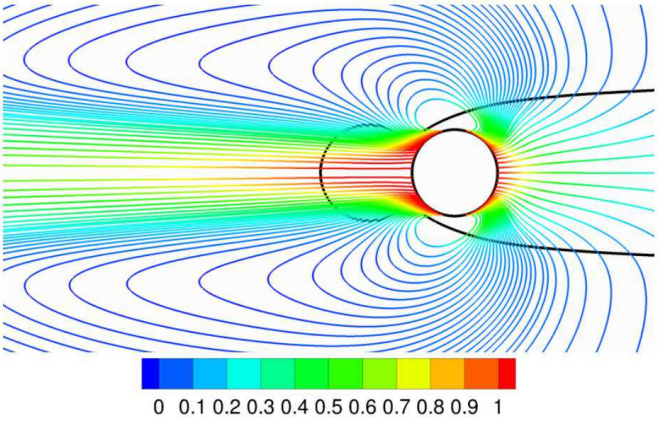


Figure 2.3: Magnitude of the drag-adjoint velocity field around a cylinder, $Re = 20$ [235].

with respect to a set of variables that can be directly controlled, i.e. the design parameters $\boldsymbol{\alpha}$. This sensitivity is given by the product of the adjoint solution with the source term $\frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}}$ induced by a design variable as it arises in (2.15) [167]; this further clarifies the role of adjoint variables as transfer weights, relating a change in cost function to a field perturbation and down to the change in design variables that would cause such a perturbation.

2.3 Practical aspects of discrete adjoints

As mentioned in Section 2.2.4, one of the most attractive features of discrete adjoints is the fact that their assembly can be automated, meaning that modifying the primal does not require any extra work on the adjoint side. The present work shall focus extensively on modifications of the primal intended to improve certain properties of the corresponding adjoint; in fact, an entire primal CFD solver will be developed de novo for this purpose. For this reason, the discrete adjoint is hereby chosen as the preferred approach in the context of this thesis. The following is a preliminary review of some practical aspects of discrete adjoint implementation.

2.3.1 Algorithmic Differentiation

The concept of *Algorithmic Differentiation* (AD) is closely associated with discrete adjoint code development. Griewank and Walther were at the forefront of the theoretical developments of AD, and authored the most comprehensive reference book on the topic [108]. The key idea of AD is to view an algorithm, or a computer program, no matter how com-

plex and lengthy, as a sequence of simple operations involving elementary mathematical functions, each having a well known analytical derivative expression. By computing the derivative value of each operation and applying the chain rule of differentiation to the whole sequence, one can obtain the derivatives of the final result with respect to a set of input variables. This is applicable to the task of computing cost function sensitivities in gradient-based CFD optimisation: the computer program is the CFD solver itself, the final evaluated quantity is the cost function J, and the set of input variables are the design parameters $\boldsymbol{\alpha}$.

The concept is clarified here following guidelines by Hascoët [117]. Considering a program that takes as input a vector-valued quantity $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0$ and performs a series of n_p operations p_k to ultimately compute a (vector) quantity $\mathbf{j} = p(\boldsymbol{\alpha}_0)$, such a sequence of operations can be written as

$$\mathbf{j} = p_{n_p} \circ p_{n_p-1} \circ \ldots \circ p_1 \left(\boldsymbol{\alpha}_0\right) \tag{2.22}$$

where "$\circ$" denotes function composition. Notice that expression (2.22) takes into account the generic case involving multiple cost functions, i.e. $\mathbf{j}$ is an array of size n_J. As a consequence, the required sensitivity of $\mathbf{j}$ is in fact a $n_\alpha \times n_J$ Jacobian matrix $\nabla\mathbf{j}$, its i-th row holding all gradient components of the i-th cost function with respect to $\boldsymbol{\alpha}$. Denoting by $\mathbf{y}_k$ a vector representing the intermediate state of all program variables resulting from the k-th operation, i.e.

$$\mathbf{y}_k = p_k \circ p_{k-1} \circ \ldots \circ p_1 \left(\mathbf{y}_0\right) \tag{2.23}$$

with $\mathbf{y}_0 = \boldsymbol{\alpha}_0$, and knowing the derivative expression p'_k of each statement p_k, a *directional derivative* $\dot{\mathbf{j}}$ in direction $\dot{\mathbf{d}}$ can be evaluated at $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0$ by applying the chain rule:

$$\dot{\mathbf{j}} = p'_{n_p}\left(\mathbf{y}_{n_p-1}\right) \cdot p'_{n_p-1}\left(\mathbf{y}_{n_p-2}\right) \cdot \ldots \cdot p'_1\left(\mathbf{y}_0\right) \cdot \dot{\mathbf{d}} \ . \tag{2.24}$$

The direction $\dot{\mathbf{d}}$ is evidently a vector belonging to the design space of $\boldsymbol{\alpha}$ and is referred to as the *seed* of the differentiation chain, i.e. the vector of partial derivatives of each element of $\boldsymbol{\alpha}$ with respect to a single scalar variable. Hence, choosing a $\dot{\mathbf{d}}$ corresponding to the direction of one element α_i of $\boldsymbol{\alpha}$, i.e. a null vector except for $\dot{d}_i = 1$, the quantity $\dot{\mathbf{j}}$ evaluated via the derivative chain will be an array holding the partial derivatives of each element of $\mathbf{j}$ with respect to parameter α_i, evaluated at $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0$. In other words, when the chain is seeded as described, a single evaluation of (2.24) yields the partial derivatives of all cost functions with respect to one single design variable, i.e. a column of the Jacobian $\nabla\mathbf{j}$. By seeding (2.24) with the direction of each α_i, the full sensitivity Jacobian can thus be assembled. The computational cost of this operation is easily estimated: assuming that evaluating the derivative value p'_k of each expression p_k is roughly as costly as

evaluating the expression itself (which is true for most common operators and functions), and noticing that each original evaluation p_k still needs to be performed in order to obtain intermediate values $\mathbf{y}_k$, then a full evaluation of (2.24) comes at approximately twice the cost of a run of the program itself, and therefore in the same order of magnitude. However, since a computer program evaluates (2.24) from right to left, i.e. starting from the seed, a full sensitivity computation would require evaluating (2.24) for as many times as the number of design variables; this process, known as *forward-mode* AD, is essentially the AD equivalent to the direct sensitivity computation defined by (2.12) via (2.9), and suffers from the same obvious practicality issues in engineering applications of industrial interest, where costly CFD solves and large numbers of shape parameters make the process prohibitively expensive in terms of runtime.

The concept of forward-mode AD can be further clarified via a simple example. The following pseudo-code for the subroutine `COST` contains the sequence of instructions evaluating function $J(x_1, x_2) = \sin^2(x_1) + \cos^2(3x_2)$:

Example 2.1.

```
SUBROUTINE COST(→x1,→x2,←J)
y1  = sin(x1)
y2  = y1*y1
y3  = 3*x2
y4  = cos(y3)
y5  = y4*y4
J   = y2 + y5
END
```

where "$\rightarrow$" and "$\leftarrow$" denote input and output arguments, respectively. Applying the concepts of forward-mode AD to differentiate the output `J` with respect to inputs `x1` and `x2` yields:

Example 2.2.

```
SUBROUTINE DCOST(→x1,→dx1,→x2,→dx2,←J,←dJ)
y1  = sin(x1)  ; dy1  = cos(x1)*dx1
y2  = y1*y1    ; dy2  = 2*y1*dy1
y3  = 3*x2     ; dy3  = 3*dx2
y4  = cos(y3)  ; dy4  = -sin(y3)*dy3
y5  = y4*y4    ; dy5  = 2*y4*dy4
J   = y2 + y5  ; dJ   = dy2 + dy5
```

```
END
```

which is essentially a repetition of each instruction followed by its differential expression. The reader can easily verify how, for instance, seeding (initialising) `dx1 = 1` and `dx2 = 0` returns in `dJ` the value of $\frac{\partial J}{\partial x_1}$.

An alternative is *reverse-mode* AD. Let the scalar-valued function

$$\bar{\mathbf{j}}^T \cdot \mathbf{j} = \bar{\mathbf{j}}^T \cdot p\left(\boldsymbol{\alpha}_0\right) \ , \tag{2.25}$$

i.e. a linear combination of components of $\mathbf{j}$ with weighting vector $\bar{\mathbf{j}}$, be considered. After transposition, its gradient $\bar{\mathbf{d}}$ results in

$$\bar{\mathbf{d}} = \bar{\mathbf{d}} \cdot p'^T\left(\boldsymbol{\alpha}_0\right) = p_1'^T\left(\mathbf{y}_0\right) \cdot \ldots \cdot p_{n_p-1}'^T\left(\mathbf{y}_{n_p-2}\right) \cdot p_{n_p}'^T\left(\mathbf{y}_{n_p-1}\right) \cdot \bar{\mathbf{j}} \ . \tag{2.26}$$

In this instance, the rightmost term $\bar{\mathbf{j}}$ constitutes a seed related to cost functions, rather than design variables. Hence, in analogy with what observed for (2.24), the derivative "direction" is now fixed in the space of $\mathbf{j}$ and the resulting vector $\bar{\mathbf{d}}$ may be interpreted as the sensitivity with respect to $\boldsymbol{\alpha}$, evaluated at $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0$, of some combination of cost functions weighted via $\bar{\mathbf{j}}$. Therefore, by seeding (2.26) with a null vector except for $\bar{j}_i = 1$, the computed $\bar{\mathbf{d}}$ will hold the full gradient of the i-th cost function with respect to all design variables, i.e. the i-th row of the sensitivity Jacobian $\nabla\mathbf{j}$. Since, again, a computer program evaluates (2.26) from right to left, the first AD computation refers to the transpose of p'_{n_p}, i.e. the derivative expression of the last operation performed by the original program. The transposed differentiation chain then accumulates derivative values "in reverse" (hence the name *reverse-mode*), with $p_1'^T\left(\mathbf{y}_0\right)$ being evaluated last. Reverse-mode AD thus requires n_J evaluations of (2.26) in order to compute the full Jacobian $\nabla\mathbf{j}$, at a CPU cost that is roughly equivalent to that of n_J runs of the original program, independently of n_α. It is thus a viable option for problems with few objectives but many shape parameters. There is an evident parallelism between reverse-mode AD and discrete adjoint sensitivity analysis as defined by (2.15), which is why a code produced by reverse-mode AD is usually referred to as adjoint code.

As an example, reverse-mode AD is applied to the pseudo-code from Example 2.1. In this case the differentiated subroutine is:

Example 2.3.

```
SUBROUTINE COSTB(→Jb,↔x1b,↔x2b)
y5b  = 1*Jb
y2b  = 1*Jb
```

```
y4b  = 2*y4*y5b
y3b  = -sin(y3)*y4b
x2b  = x2b + 3*y3b
y1b  = 2*y1*y2b
x1b  = x1b + cos(x1)*y1b
END
```

Assuming that a *forward sweep* of the original subroutine was performed and intermediate `y` values stored, the reverse-differentiated code produces derivative values in `x1b` and `x2b`. Seeding `Jb = 1` (and initialising to zero `x1b` and `x2b`) leads, with a single evaluation, to the computation of the complete gradient of J.

When multiple routines are called by the main program, the computational graph for reverse-AD routines is a backwards reflection of the primal graph: it starts with the differentiated routine of the last primal instruction and goes through the call tree in reverse, up to the input variables. Like for forward-mode, the computational cost associated with a reverse-differentiated procedure is in the same order of magnitude as the original routine. In fact Griewank shows that, under reasonable assumptions, "the evaluation of a gradient requires never more than five times the effort of evaluating the underlying function by itself" [108]; this bound includes the cost of "storing" and "fetching" the required intermediate variables.

Besides being a powerful tool for implementing an adjoint code, reverse-mode AD also sheds an interesting light on the matter as it allows to determine which output depends on which input, and to what value. It has been observed [167] how the backwards structure typical of an adjoint code leads the following interpretation: while a direct sensitivity analysis (or forward-mode AD) answers to the question: "If a change is applied to a design parameter, how does it affect all cost functions?", the dual/adjoint approach (or reverse-mode AD) views the problem in reverse, i.e. "If a perturbation is present in a cost function, how does it arise from perturbations in the design parameters?".

2.3.2 Automatic Differentiation

In principle, application of Algorithmic Differentiation to a piece of code - in both forward and reverse-mode - requires no knowledge of what the code is computing as long as the input and output variables are clearly defined. Provided that the primal source code is accessible, one may proceed to differentiate each instruction as shown in the examples in Section 2.3.1. The process is mechanical, tedious and time-consuming. However, and more importantly: the process is rule-based and thus can be automated.

The mechanical nature of AD has driven research efforts towards the development of so-called *Automatic Differentiation* tools, also known by the initialism AD, often standing for either Algorithmic and Automatic Differentiation interchangeably. As the name suggests, AD tools are software facilities that automatically produce the differentiated version of a piece of code they are fed with; the developer only needs to provide the code, define the relevant input and output variables (design parameters and cost functions, respectively), and specify whether forward or reverse-mode is sought. There are two categories of AD tools: *source transformation* and *operator overloading*. Source transformation modifies the primal source code by adding variables holding derivative values and lines of code to evaluate them (the code snippets shown in Examples 2.2 and 2.3 are examples of forward and reverse-mode source transformation); operator overloading in forward mode replaces floating point variables with an augmented type which additionally stores a differentiated value such that, for each operand acting on the original variable, its corresponding derivative operand will act on the additional one, while in reverse mode it keeps track of primal operands and results on a "tape" which is subsequently interpreted in reverse to produce the adjoint. The online community portal Autodiff.org[8] provides an exhaustive list of AD tools currently available and offers an overview of recent trends in the field.

Reverse-mode AD is indeed a powerful tool for considerably reducing the workload of coding the discrete adjoint of a legacy CFD solver. Some of its drawbacks - and state-of-the-art solutions - should also be mentioned:

- Memory consumption: as highlighted by the example in Section 2.3.1, values of primal variables may be needed in the adjoint run; therefore, reverse-mode AD requires in principle the full trajectory of the original code to be either stored or recomputed, which in naïve implementation may lead to prohibitive memory consumption or increased CPU time, respectively. *Checkpointing* is often put forward as a suitable trade-off: *snapshots* of the memory state are stored at given intervals (checkpoints) during the primal run; the reverse run then computes the required values by restarting the primal from the closest checkpoint. Checkpointing is also essential in large transient CFD simulations, where knowledge of the states at each time-step is necessary regardless of whether or not AD is used. Considerable efforts have been made, notably by Griewank and Walther [106, 107], in order to identify the theoretically ideal way of distributing checkpoints along the timeline of a program. Several other practical checkpointing strategies [123, 236] have also been proposed. Some AD tools can further optimise memory usage via *activity analysis* [118], i.e. by identifying those variables whose differentiated counterparts would

[8] http://www.autodiff.org

hold null or otherwise useless values.

- Iterative processes: primal codes often perform fixed-point iterations (FPI) to evaluate non-linear functions, starting from an initial guess until a convergence criterion is satisfied. It has been observed [55, 56, 96] that, in this case, a "brute-force" application of the AD concept (i.e. the line-by-line differentiation of each primal statement) can lead to erroneous results: the differentiated loop would be executed for as many iterations as the primal, which is an arbitrary number (since it depends on the initial guess) and therefore would not not guarantee convergence of derivative values. Besides, the reverse-AD code would need to store all intermediate values occurring at each cycle, which is unnecessary since FPI loops are path-independent processes (i.e. only the converged value, and thus its derivative, matters). State-of-the-art AD avoids brute-force differentiation of FPIs thanks to the strategy presented by Christianson [56]: if the primal features an FPI solving for $\mathbf{w}$ such that $\Psi(\mathbf{w}(\mathbf{y}), \mathbf{y}) = \mathbf{0}$, then graph construction and taping are switched off during the iterative procedure and, in the reverse-AD code, the loop is replaced by the adjoint equation: $\left(\frac{\partial \Psi}{\partial \mathbf{w}}\right)^T \mathbf{w}^* = \left(\frac{\partial J}{\partial \mathbf{w}}\right)^T$, with the Jacobian $\frac{\partial \Psi}{\partial \mathbf{w}}$ evaluated at the converged $\mathbf{w}$; it then follows $\frac{dJ}{d\mathbf{y}} = \frac{\partial J}{\partial \mathbf{y}} - (\mathbf{w}^*)^T \frac{\partial \Psi}{\partial \mathbf{y}}$, after which the conventional reverse-mode AD process is resumed. The article provides a recipe for building $\frac{\partial J}{\partial \mathbf{w}}$ and $\frac{dJ}{d\mathbf{y}}$ using AD, and points out that any suitable strategy (iterative or not) may be considered for solving the adjoint equation. One such strategy is suggested by the same author in an earlier publication [55] as a way of automating the adjoint solution as well: if $\mathbf{w} = \Phi(\mathbf{w}, \mathbf{y})$ is the primal FPI iterator used to solve $\Psi(\mathbf{w}, \mathbf{y}) = \mathbf{0}$, then an adjoint iterator $\frac{\partial \Phi}{\partial \mathbf{w}}$ may be obtained by reverse-differentiating a single primal iteration after convergence, to be used in an adjoint FPI of the form $\mathbf{w}^{*,n+1} = \left(\frac{\partial J}{\partial \mathbf{y}}\right)^T + \left(\frac{\partial \Phi}{\partial \mathbf{w}}\right)^T \mathbf{w}^{*,n}$. This approach essentially generates the adjoint version of the primal FPI solution algorithm. An important result shown by Christianson is that, if the primal iterator is *well behaved* (attractive), then the adjoint FPI is guaranteed to converge, and its asymptotic rate of convergence is the same as that of the primal.

- Closed-source function calls: a primal may make use of procedures from third-party libraries whose source code is not available and therefore cannot be submitted to AD tools. In this case, the user must gain knowledge of the corresponding discrete adjoint procedure and include it manually in the differentiated code. Efforts are being made by AD tool developers to automate this process, where possible. *Linear Solver Replacement* (LSR) is a typical example: some AD tools include directives allowing the user to identify a location where the primal calls an external linear algebra library to solve a linear system; the user needs only specify the relevant

inputs and outputs, and the tool automatically replaces the solve with its adjoint counterpart. It should also be mentioned that, even in the open-source case, it is often inefficient/illogical to reverse-differentiate through a linear solver; for instance, an iterative linear solver should be submitted instead to a treatment similar to the one for FPIs described above.

- Code preparation: the automatic nature of AD does not imply that the tool will always work "out of the box". This is notably the case when dealing with commercial legacy codes, which often contain a number of non-differentiable code statements and language features (data structures, memory management, input/output handling, etc.) unsupported by AD; these need to be manually rewritten or eliminated. Some human intervention is also required if the goal is to improve the efficiency of the AD-generated code - specifically in reverse mode - which in turn requires in-depth knowledge of the AD tool itself (to that end, several techniques are discussed by Müller and Cusdin [64, 168]).

2.3.3 Equational Differentiation

The remarks from Section 2.3.2 make it clear that applying reverse-mode AD indiscriminately to an entire CFD code (the "brute-force" approach) is not only impractical, but also highly inefficient - notably in terms of memory footprint - and not always logical. In practice, this is not how AD is applied today to full CFD codes. Isolated parts of the primal are submitted for AD treatment instead: the outputs of the resulting differentiated routines are then interpreted in terms of linear algebra [97], and these routines are assembled by a hand-written driver code to produce the full adjoint solver. Despite this, when using AD, one risks focusing on the algorithmic premise of the method (i.e. AD differentiates a computer program line-by-line, a process which is rule-based and thus automatable) and losing sight of the corresponding equations being solved. This may lead to consider the AD approach as a "special case" of discrete adjoint which operates at the code level and is agnostic with respect to what is being solved, or to view AD as the only viable way of obtaining a discrete adjoint code automatically.

To avoid that risk, the present work advocates the *Equational Differentiation* (ED) philosophy (as named by Pierrot [195]). ED suggests developing discrete adjoints within a framework where the distinction among equations, algorithms and coding tools is clear. In other words, it encourages the developer to maintain the separation between the *what* and the *how*: what the equations to be solved are, how they are solved and how the required terms are assembled in practice. It should be stressed that ED is not intended to introduce any new concepts: on the contrary, it imposes strict adherence to each step

outlined in the original definition of discrete adjont: obtain a discrete adjoint solution which satisfies the adjoint system (2.13), then compute sensitivity through (2.15). While ED demands these steps to be clearly distinguishable in the code, it does not impose any constraints on the strategies and tools used for the assembly of the adjoint system (which indeed may be done by AD, see below), on whether or not the process is automated, or on the selected solution algorithm. The advantage - specifically in a research setting - is that the user can switch and compare between strategies with relative ease: for instance, the freedom of choice on solution algorithms allows to investigate a number of strategies for the adjoint system (see Section 6.2) independently of how the primal is solved.

Any discrete adjoint code which exhibits this kind of flexibility fits within the ED philosophy; several instances can de facto be found in the literature. An exemplary implementation is the one presented by the DOLFIN-adjoint[9] project: the approach looks at the primal as a sequence of *equation solves*; in the FEniCS[10] development framework, this sequence is written in a high-level symbolic representation; DOLFIN-adjoint takes advantage of this abstraction to automatically derive the corresponding sequence of adjoint equations at the same level of abstraction (using AD-generated routines where necessary), with the declared objective of maintaining a "clean separation between mathematical intention and computer implementation" [85].

Three ED-compatible implementations are hereby named and described:

- ED1 - *hand-derived discrete adjoint*: arguably the most transparent strategy, it consists in deriving analytically the expressions corresponding to each entry of the Jacobian $\frac{\partial \mathbf{r}}{\partial \mathbf{w}}$, the adjoint right-hand side $\frac{\partial J}{\partial \mathbf{w}}$, the matrix $\frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}}$ and the direct sensitivity $\frac{\partial J}{\partial \boldsymbol{\alpha}}$, and implement routines that assemble them explicitly. The adjoint system is then solved via any suitable algorithm. Examples of hand-derived descrete adjoints include that presented by Giles et al. [98] and the NASA FUN3D[11] code. As mentioned before, the approach requires full knowledge of the residual expression, or at least access to the source code from which it can be deduced. Manual derivation is arguably the most efficient strategy in terms of both memory footprint and CPU time, but the derivation process is lengthy, error-prone and, as observed by Nadarajah [173], it quickly grows in complexity along with the underlying CFD schemes. It is also affected by maintenance issues because, each time a new scheme or feature is implemented in the primal which modifies the residual expression, its corresponding relevant derivatives have to be implemented in the adjoint assembly routines, which is problematic in an industrial context where several developers,

[9] http://www.dolfin-adjoint.org
[10] https://fenicsproject.org/
[11] https://fun3d.larc.nasa.gov/

not all necessarily aware of the details of adjoint functionalities, may be working on the same code.

- ED2 - *automatic reverse assembly*: similar to the hand-derived case the goal is to assemble the full Jacobian, adjoint right-hand side and the other sensitivity terms arising in (2.15), but in an automated way so that, once a driver code is in place for guiding the assembly and solution process, no further manual work is needed even if the primal is modified. The two main viable tools to achieve this are AD and Finite Differencing (FD). For example, assuming that the primal contains a routine `RESIDUAL`($\rightarrow\mathbf{w}$,$\rightarrow\boldsymbol{\alpha}$,$\leftarrow\mathbf{r}$) which computes the residual $\mathbf{r}(\mathbf{w},\boldsymbol{\alpha})$ for a given $\mathbf{w}$ and $\boldsymbol{\alpha}$, then the entries of the Jacobian $\frac{\partial \mathbf{r}}{\partial \mathbf{w}}$ can be computed either by: repeatedly calling the (forward-mode) AD routine `DRESIDUAL`, seeding each time for a different degree of freedom of $\mathbf{w}$; or repeatedly calling the original `RESIDUAL` routine, introducing each time a perturbation $\boldsymbol{\delta}\mathbf{w}$ on each degree of freedom of $\mathbf{w}$, and then computing the corresponding FD value. Either way, the cost of the process would scale in principle with the cardinality of $\mathbf{w}$, but this can be avoided thanks to colouring algorithms (see Section 6.1.3). An analogous procedure allows to assemble e.g. the adjoint right-hand side $\frac{\partial J}{\partial \mathbf{w}}$ based on a routine `COST`($\rightarrow\mathbf{w}$,$\rightarrow\boldsymbol{\alpha}$,$\leftarrow J$); in this case, if AD is employed, then reverse-mode is preferable since it requires only one call to the AD routine `COSTB`.

  An AD-based reverse assembly has the advantage of always computing exact derivative values, barring non-differentiable terms. Conversely, a FD-based assembly does in general introduce a truncation error depending on the chosen step-size (with some exceptions, see Section 6.1.6), but on the other hand it also introduces some smoothening, which may better represent how the Jacobian responds to a field perturbation in the vicinity of discontinuities in derivative values or small, non-physical numerical oscillations in the discretisation scheme. The FD approach also has the advantage of being non-intrusive: provided that interfaces such as `RESIDUAL` and `COST` are available (usually as *user subroutines* in commercial CFD solvers), then there is no need to modify - or indeed have access to - the source code within this function.

- ED3 - *AD reverse accumulation*: this is a specific implementation of Christianson's generic FPI treatment discussed in Section 2.3.2. Considering again the `RESIDUAL` procedure mentioned above, it has been shown [97] how its reverse-AD counterpart `RESIDUALB`($\rightarrow\mathbf{rb}$,$\leftrightarrow\mathbf{wb}$,$\leftrightarrow\boldsymbol{\alpha}\mathbf{b}$) can be exploited to compute $\left(\frac{\partial \mathbf{r}}{\partial \mathbf{w}}\right)^T\mathbf{v}$ (returned in $\mathbf{wb}$) for any vector $\mathbf{v}$, namely by seeding $\mathbf{rb}=\mathbf{v}$ and $\mathbf{wb}=\mathbf{0}$. Furthermore, as mentioned above, the adjoint right-hand side $\frac{\partial J}{\partial \mathbf{w}}$ is easily computed through the AD procedure `COSTB`($\rightarrow Jb$,$\leftrightarrow\mathbf{wb}$,$\leftrightarrow\boldsymbol{\alpha}\mathbf{b}$). The two results combined allow to compute

the adjoint residual for a given adjoint state $\mathbf{w}^*$, which can be used to solve the adjoint system via an iterative algorithm of the form

$$\mathbb{P}\left(\mathbf{w}^{*,n+1} - \mathbf{w}^{*,n}\right) = \left(\frac{\partial J}{\partial \mathbf{w}}\right)^T - \left(\frac{\partial \mathbf{r}}{\partial \mathbf{w}}\right)^T \mathbf{w}^{*,n} \ . \tag{2.27}$$

$\mathbb{P}$ can either be a diagonal matrix, thus giving rise to a matrix-free pseudo-timestepping algorithm, or any preconditioner one may deem suitable - a common choice is to reuse the (transposed) primal preconditioner, an option which shall be explored in the present work (Section 6.2). The approach is popular amongst AD users (see e.g. Giles et al. [98], Christakopoulos et al. [54], Courty et al. [61]). With respect to both ED1 and ED2, ED3 has the advantage of not requiring storage of the full Jacobian matrix. While it does limit the choice of the preconditioner $\mathbb{P}$ to one that does not need explicit knowledge of the exact Jacobian (unlike e.g. those shown in Sections 6.2.2 and 6.2.3), the range of choices is wide enough that the strategy is considered as fitting within the ED philosophy.

2.4 Challenges of discrete adjoints

In the context of discrete adjoint CFD, how strict the tolerance should be on the primal solution is a matter of debate. On one hand the Jacobian and adjoint right-hand side are evaluated at the converged state, hence a poorly converged primal field necessarily contributes to the error on the gradient; on the other hand it can be argued that, in practice, strict gradient exactness is not required by an optimisation algorithm (unless full convergence is sought) as long as the sensitivity is reasonably oriented towards the right direction, as is the case in the continuous approach or even in discrete settings, where it has been shown for some cases [74] how replacing the adjoint system with an approximation - at the expense of gradient consistency - can lead to overall satisfactory results in terms of optimisation.

However, in the case of a non-linear problem such as the steady-state Navier-Stokes, it has been observed [37] that while it is often possible to obtain a reasonably converged solution for the primal, difficulties arise when it comes to converging its adjoint equation. Let $\mathbf{r}(\mathbf{w}) = \mathbf{0}$ be a non-linear primal which is solved via a FPI of the form

$$\mathbf{w}^{n+1} = \mathbf{w}^n - \mathbb{P}^{-1,n}\mathbf{r}(\mathbf{w}^n) \tag{2.28}$$

where $\mathbb{P}$ denotes a generic, solution-dependent preconditioner, typically chosen such that $\mathbb{P}^{-1} \approx \left(\frac{\partial \mathbf{r}}{\partial \mathbf{w}}\right)^{-1}$. This is notably the case for iterative schemes for the steady-state, incom-

pressible Navier-Stokes (see Section 5.2). Giles et al. [99] show that, if the (transposed) primal iterator is inherited by the adjoint solver:

$$\mathbf{w}^{*,n+1} = \mathbf{w}^{*,n} - \mathbb{P}^{-T,N}\left(\left(\frac{\partial \mathbf{r}^N}{\partial \mathbf{w}}\right)^T \mathbf{w}^{*,n} - \left(\frac{\partial J^N}{\partial \mathbf{w}}\right)^T\right) \tag{2.29}$$

(where the superscript N denotes the last primal iteration, i.e. evaluation at convergence), then convergence is guaranteed at an asymptotically equivalent rate as long as the primal is contractive at convergence, in accordance with Christianson's theory of adjointed FPIs [56] discussed in Section 2.3.2. It is well-known [212] that contractivity of an operator depends on its spectral properties: it is contractive if all of its eigenvalues fall within the unit sphere, and since the adjoint iterator is the transpose of the primal one at convergence, their eigenvalues coincide.

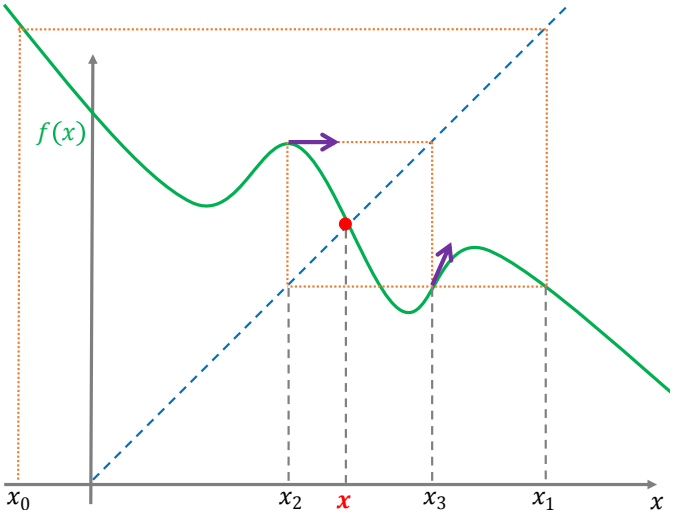


Figure 2.4: Example of a limit cycle occurring in the non-linear fixed-point iteration: $x = f(x)$.

Brezillon and Dwight [37] observe that, in many industrial cases, contractivity of the adjoint operator is affected by a number of factors, the most frequent being a) insufficient primal convergence, and b) the primal FPI not being asymptotically convergent itself. The first scenario implies that the computation has converged to the user-specified tolerance (which may be sufficiently small for mere engineering purposes) prior to the asymptotic regime; in this case, imposing a stricter tolerance solves the issue. The second scenario is more problematic: it occurs when the solver, rather than converging, stalls around *limit cycle oscillations* (LCO) [47] with unstable eigenvalues. This is illustrated in a simplified way in Figure 2.4, where a non-linear FPI solving $x = f(x)$ is plotted: starting from the initial guess x_0, the algorithm never reaches the exact solution $\widehat{x}$, stagnating instead between x_2 and x_3. The behaviour of the adjoint FPI $x^* = \frac{\partial J}{\partial x} + \frac{\partial f}{\partial x}x^*$ depends

on where the primal is stopped: if stopped at x_2 (zero slope), then it will immediately converge to a wrong solution; if stopped at x_3 (slope > 1, representative of a case with eigenvalues outside the unit sphere), then it will diverge. Considerable work has been done - mostly in the context of compressible RANS - to tackle adjoint convergence issues when this scenario occurs: some authors [141, 164] attribute the LCO phenomenon to an inherent (mild) flow unsteadiness and thus treat the problem as unsteady, with the cost function time-averaged over several cycles; others focus instead on enforcing FPI contractivity for both primal and adjoint by working on stabilisation methods for linear solvers, with popular choices being the Recursive Projection Method [47] or variants of preconditioned GMRES combined with multigrid [75, 144, 243].

As anticipated, the present thesis turns the attention to the primal discretisation schemes in an effort to eliminate features that may be partly responsible for the non-convergent behaviour. Classical Finite Volumes (FV) schemes [229] are currently the most widespread choice in the industry. The FV method is, in its basic formulation, a consistent and stable strategy; however, its practical implementation in commercial CFD software includes a number of numerical artefacts devised to allow tackling complex cases - in terms of physics and/or geometry and meshing - which are known to prevent convergence down to an arbitrarily small tolerance on residuals. Flux limiters (see Section 4.2.3) are a fitting example: the usage of flux/slope limiters, specifically in non-differentiable forms, has long been known to prevent convergence to steady-state solutions [230, 231], and researchers are actively investigating alternative limiter formulations to alleviate the issue [165, 246]. A similar example is the usage of so-called Non-Orthogonal Correctors (NOC, Section 3.3.4), quantities included in gradient approximations to take into account the non-orthogonality of the grid: NOCs are typically treated explicitly, i.e. relegated to the right-hand side of the primal system and updated at each iteration (*deferred correction* [86]); it has been observed [135, 171] how, in order to prevent the iterative process from becoming unbounded, some limiting on the correction is needed which can hamper convergence to steady-state similarly to flux limiters.

The main body of this thesis is devoted to the development of spatial discretisation strategies for CFD that are rid as much as possible of such numerical artefacts, under the hypothesis that this can improve primal convergence to steady-state and thus indirectly alleviate the instabilities arising in the discrete adjoint.

Chapter 3

Mixed Hybrid Finite Volumes

3.1 Mixed Virtual Elements

In the last two decades, research has produced a new family of discretisation schemes for CFD driven by the need for a method capable of dealing with generic settings in terms of both discretised geometry (generic grids, including non-conforming, distorted, skewed, or non-convex elements) and problem physics (anisotropic material properties, discontinuities) without loss of accuracy or stability. Hydrogeology, oil reservoir simulation, plasma physics, semiconductor modelling, biology are typically mentioned as potential industrial applications [82] as they often call for simulations of both complex geometries and/or anisotropic and heterogeneous material properties.

On of the earliest of these schemes, known as *Mimetic Finite Differences* (MFD), was pioneered by Brezzi, Lipnikov and Shashkov [41, 42] who focused on the *mimetic* nature of the method, i.e. the idea of constructing discrete operators that mimic key properties of their continuous counterparts. The second-order accurate MFD scheme has been extensively investigated and analysed since, and its power demonstrated initially on the pure anisotropic diffusion problem (and similar) over generic polyhedral meshes [41, 42, 49, 143, 151, 153]. A history of MFD is detailed in te recent book by Beirão da Veiga, Lipnikov and Manzini [20]. Current trends are overviewed by Lipnikov, Manzini and Shashkov [152]. Following the generalisation of MFD to higher orders of accuracy [19, 23, 150] the scheme was recast as *Mixed Virtual Element*[1] method (MVE) [15, 39], a name that highlights the link with *Mixed* and *Hybrid Finite Elements* (MFE/HFE) [40] and emphasises the *virtual* aspect of the scheme, i.e. the fact that MVE does not

[1]The scheme is referred to as either Mixed Virtual Element (MVE) or Virtual Element Method (VEM). The present work adopts the former nomenclature.

require explicit knowledge and construction of shape functions, thus allowing for more freedom in element shape. The MVE scheme has also been extended to general elliptic problems including convection-diffusion-reaction [17, 20, 50], Stokes [21, 48] and, very recently, steady incompressible Navier-Stokes [22].

A second community of researchers led primarily by Droniou and Eymard [71] has developed a similar family of schemes, named either *Hybrid Mimetic Mixed* method (HMM) [70] or *Mixed Finite Volumes* (MFV) [71], thus highlighting the link with classical Finite Volumes (FV). The development of HMM/MFV proceeded in parallel with that of MVE, focusing on anisotropic diffusion problems first [71, 82], then convection-diffusion [18, 70, 191], Stokes and Navier-Stokes [72]. The HMM community is also credited for its efforts to show the many similarities between HMM and MVE [18, 73], noticing that a) diffusion operators from each method are often different derivations of algebraically similar or coincident schemes, and b) it is possible to define a unified framework to include inherently different ways of handling convective terms. A recent doctoral thesis by J. Bonelle [35] also made a significant contribution towards comparison, classification and unification of both frameworks and others under the umbrella term of *Compatible Discrete Operator* (CDO) schemes.

3.2 Basic MVE concepts

In order to illustrate the basic MVE concepts it is convenient and customary [42] to start by discretising the *pure anisotropic diffusion equation*:

$$\nabla \cdot (-\mathbb{K}\nabla\phi) = f \quad \text{in } \Omega \tag{3.1}$$

where ϕ is the unknown scalar, f the source term and $\mathbb{K}$ a symmetric tensor describing the anisotropic diffusivity of the material. For ease of exposition the homogeneous Dirichlet boundary value problem is considered over a d-dimensional domain Ω. Equation (3.1) is first rewritten in *mixed formulation*:

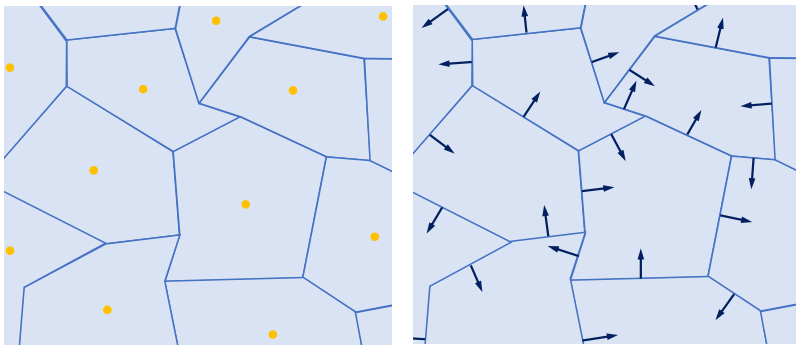
$$\begin{cases} \vec{V} = -\mathbb{K}\nabla\phi \\ \nabla \cdot \vec{V} = f \end{cases} \quad \text{in } \Omega \tag{3.2}$$

where the vector variable $\vec{V}$ is introduced as the negative gradient of ϕ scaled by $\mathbb{K}$.

It will be shown how, unlike classical FV, one of the key ideas common to all MFD/MVE philosophies is to replicate the mixed formulation (3.2) in the discrete setting, specifically by introducing a true independent discrete unknown to represent the vector variable $\vec{V}$.

3.2.1 Discrete spaces and scalar products

The four steps required for the construction of a MVE-like scheme are described in this section and the next as outlined by all main references on the subject (see e.g. [41, 42, 49, 143]).



(a) Q^h (b) X^h

Figure 3.1: Location of degrees of freedom for MVE spaces on a generic 2D polygonal mesh.

Step one: let Ω be a polyhedron, and Ω_h a non-overlapping partition of Ω into n_C polyhedral elements (or *cells*) with n_F planar faces, such as a typical FV mesh. Two discrete spaces are defined: Q^h, holding degrees of freedom at cells C, and X^h, holding degrees of freedom at faces F, as shown (in 2D) in Figure 3.1. Q^h shall be used to discretise scalar quantities via the following de Rham map [36]:

$$\phi_C = \frac{1}{|C|} \int_C \phi \, dV \quad \forall C \in \Omega_h \tag{3.3}$$

where $|C|$ is the cell volume (or area if $d = 2$). Hence scalar quantities are represented in Q^h via their cell-averaged values, which might be thought of as *d-forms*, or *d-cochains* [42]. The notation $\boldsymbol{\phi}$ (boldfaced) is also introduced to identify the vector (in Q^h) of unknowns across the grid, meaning that ϕ_C denotes the C-th degree of freedom of $\boldsymbol{\phi}$:

$$(\boldsymbol{\phi})_C = \phi_C \ . \tag{3.4}$$

Similarly, for X^h the following de Rham map is defined:

$$V_{FC} = \int_F \vec{V} \cdot \vec{n}_{FC} \, dS \quad \forall C \in \Omega_h, \forall F \in \partial C \tag{3.5}$$

where $\vec{n}_{FC}$ is the unit vector normal to face F outward with respect to cell C. Expression (3.5) entails that vector quantities are represented in X^h via their *fluxes* across faces

(which might be thought of as discrete $(d-1)$-*forms*). This would in principle imply that the number of degrees of freedom for any vector belonging to X^h is twice the number of internal faces in the mesh (plus the number of boundary faces if boundary conditions other than Dirichlet are applied, see Section 3.3.5). However, it is imposed that fluxes in X^h must be conservative across each face:

$$V_{FC+} + V_{FC-} = 0 \quad \text{, i.e.} \quad s_{FC+}V_F + s_{FC-}V_F = 0 \tag{3.6}$$

where F is the common face between cells $C+$ and $C-$, s_{FC} is the conventional sign (assumed fixed once and for all) defining the cell-face ordering between C and F, and V_F is an "unsigned" flux. This reduces the number of degrees of freedom in X^h to the number of faces. In order to define a convention: in the remainder of this work, the orientation of face F between cells $C+$ and $C-$ is set such that $s_{FC+} = 1$ and $s_{FC-} = -1$. As a consequence, the value of flux V_F in X^h shall be positive if outward with respect to $C+$. To conclude, notation $\mathbf{V}$ is introduced to represent a vector belonging to X^h, such that V_F corresponds to the F-th component of $\mathbf{V}$:

$$(\mathbf{V})_F = V_F = V_{FC+} \ . \tag{3.7}$$

Step two: spaces Q^h and X^h are equipped with scalar products, hereby defined as the summation over all cells of a scalar product defined locally on each cell, i.e.

$$\begin{aligned} \langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle_{Q^h} &= \sum_{C \in \Omega_h} \langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle_{C,Q^h} \qquad \text{for } Q^h \\ \langle \mathbf{V}, \mathbf{W} \rangle_{X^h} &= \sum_{C \in \Omega_h} \langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h} \qquad \text{for } X^h \ . \end{aligned} \tag{3.8}$$

The definition of the local scalar product for Q^h is fairly straightforward:

$$\langle \boldsymbol{\phi}, \boldsymbol{\psi} \rangle_{C,Q^h} = \phi_C \psi_C \left| C \right| \ . \tag{3.9}$$

It is less trivial for X^h. Denoting by m_C the number of faces delimiting cell C, Brezzi et al. [42] observe that the local scalar product definition on C implies the existence of a $m_C \times m_C$ symmetric positive-definite (SPD) matrix $\mathbb{M}_C$ which represents it:

$$\langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h} = \left(\mathbb{M}_C \left(\mathbf{V} \right)_{\partial C}, \left(\mathbf{W} \right)_{\partial C} \right) \tag{3.10}$$

where $(\cdot, \cdot)$ denotes the Euclidean inner product in $\mathbb{R}^{m_C}$, and $(\cdot)_{\partial C}$ denotes the restriction

of a vector in X^h to the faces delimiting cell C and oriented accordingly, i.e.

$$(\mathbf{V})_{\partial C} = \begin{pmatrix} s_{F_1 C} V_{F_1} \\ s_{F_2 C} V_{F_2} \\ \cdots \\ s_{F_{m_C} C} V_{F_{m_C}} \end{pmatrix} = \begin{pmatrix} V_{F_1 C} \\ V_{F_2 C} \\ \cdots \\ V_{F_{m_C} C} \end{pmatrix} \tag{3.11}$$

with $F_1, F_2, \cdots, F_{m_C}$ being the m_C faces delimiting cell C. The construction of a suitable $\mathbb{M}_C$ is arguably the most difficult task, the core of any MFD/MVE-like scheme and what differentiates each specific strategy from all others [38]. This will be discussed in Section 3.3, but it is specified from now that the scalar product (3.10) is constructed for convenience to be *material dependent*: once defined it will incorporate the material diffusivity, meaning that

$$\langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h} \approx \int_C \mathbb{K}^{-1} \vec{V} \cdot \vec{W} \, dV \tag{3.12}$$

(where, with an abuse of notation, "$\approx$" stands for "is a discrete approximation of").

3.2.2 Divergence and flux operators

Step three: a discrete divergence operator $\mathcal{D}$ is defined. Since the continuous divergence operator maps vector fields onto scalar fields, $\mathcal{D}$ is expected to do the same in the discrete spaces defined above, i.e. it should act on degrees of freedom in X^h and return values in Q^h. $\mathcal{D}$ is defined as

$$(\mathcal{D}\mathbf{V})_C = \frac{1}{|C|} \sum_{F \in \partial C} V_{FC} \ , \tag{3.13}$$

a direct application of the Gauss divergence theorem:

$$\int_C \nabla \cdot \vec{V} \, dV = \int_{\partial C} \vec{V} \cdot \vec{n}_{FC} \, dS \ . \tag{3.14}$$

A remark: $\mathcal{D}$ is an *exact* discrete differential operator, i.e. it does not involve any form of approximation. In differential geometry an operator like $\mathcal{D}$ is known as an *exterior derivative*, in this case mapping from $(d-1)$-forms to d-forms (for an exhaustive guide to exterior derivatives and their properties, see e.g. [35]). In practice, $\mathcal{D}$ is realised as a $n_C \times n_F$ matrix representing the signed face-to-cell incidence:

$$(\mathcal{D})_{CF} = \begin{cases} s_{FC} & \text{if } F \in \partial C \\ 0 & \text{otherwise} \end{cases} \ . \tag{3.15}$$

Hence $\mathcal{D}$ is a *topological* operator [35], i.e. it only requires knowledge of mesh connectivity between (in this case) faces and cells and face orientations. It does not make use of any mesh metrics or inertial quantities.

Step four: a *diffusive flux operator* $\mathcal{K}$ is derived by imposing that $\mathcal{K}$ and $\mathcal{D}$ be adjoint to each other with respect to the scalar products defined in Section 3.2.1:

$$\langle \mathcal{K}\boldsymbol{\phi}, \mathbf{W} \rangle_{X^h} = \langle \boldsymbol{\phi}, \mathcal{D}\mathbf{W} \rangle_{Q^h} \quad \forall \boldsymbol{\phi} \in Q^h \quad \forall \mathbf{W} \in X^h \ . \tag{3.16}$$

Constructing $\mathcal{K}$ such that it satisfies (3.16) is the essence of the *mimetic* nature of the method, as stated by e.g. Lipnikov et al. [153]: bearing in mind that the scalar product in X^h is material-dependent as in (3.12), equation (3.16) is in fact a discrete representation of the Gauss-Green theorem

$$\int_\Omega \mathbb{K}^{-1} \left(-\mathbb{K}\nabla\phi\right) \cdot \vec{W} \, dV = \int_\Omega \phi \nabla \cdot \vec{W} \, dV \tag{3.17}$$

(minus boundary values, which were set to zero for convenience), meaning that these discrete operators are constructed in order to mimic the behaviour of their continuous counterparts with respect to Gauss-Green. As explained by Bonelle [35], mimetic methods work by first constructing an exact *primary* discrete operator ($\mathcal{D}$ in this case) and use it to build a *derived* operator ($\mathcal{K}$) such that some key property of the continuous operator (Gauss-Green in this case) is satisfied at a discrete level with respect to the discrete scalar products (3.8). The derived operator is therefore a discrete co-differential to the primary operator, in this case mapping from d-forms to $(d-1)$-forms.

Assuming that a suitable formulation for the local scalar product matrix $\mathbb{M}_C$ is provided (see Section 3.3.1), then the global scalar product is assembled as

$$\langle \mathbf{V}, \mathbf{W} \rangle_{X^h} = \sum_{C \in \Omega_h} \left(\mathbb{M}_C \left(\mathbf{V} \right)_{\partial C}, \left(\mathbf{W} \right)_{\partial C} \right) = \left(\mathcal{H}\mathbf{V}, \mathbf{W} \right) \tag{3.18}$$

where $\mathcal{H}$ is a $n_F \times n_F$ SPD matrix. Therefore (3.16) can be rewritten equivalently as

$$\left(\mathcal{H}\mathcal{K}\boldsymbol{\phi}, \mathbf{W} \right) = \left(\mathcal{D}^T \boldsymbol{\phi}, \mathbf{W} \right) \ , \tag{3.19}$$

which yields the definition for the flux operator as $\mathcal{K} = \mathcal{H}^{-1}\mathcal{D}^T$. The specific form of $\mathcal{H}$ will depend directly on the definition of the local scalar product matrix $\mathbb{M}_C$. It will be shown in Section 3.3 how the construction of $\mathbb{M}_C$ relies on cell metrics such as lengths, areas, volumes and other inertial quantities of the mesh, meaning that $\mathcal{H}$, and therefore the flux operator $\mathcal{K}$, is a *metric* operator, as opposed to $\mathcal{D}$.

It is possible at this point to write a discrete mixed variational formulation of the homogeneous Dirichlet boundary value problem (3.2):

Find $\boldsymbol{\phi} \in Q^h$, $\mathbf{V} \in X^h$ *such that:*

$$\begin{aligned} \langle \mathbf{V}, \mathbf{W} \rangle_{X^h} - \langle \boldsymbol{\phi}, \mathcal{D}\mathbf{W} \rangle_{Q^h} &= \mathbf{0} \qquad && \forall \mathbf{W} \in X^h \\ \langle \mathcal{D}\mathbf{V}, \boldsymbol{\psi} \rangle_{Q^h} &= \langle \mathbf{f}, \boldsymbol{\psi} \rangle_{Q^h} && \forall \boldsymbol{\psi} \in Q^h \end{aligned} \tag{3.20}$$

where $\mathbf{V}$ is the flux $\vec{V} = -\mathbb{K}\nabla\phi$ discretised over X^h, $\boldsymbol{\phi}$ is the scalar solution field discretised over Q^h, and $\mathbf{f}$ is the de Rham map of the source term f onto Q^h. This ultimately leads to the saddle-point linear system

$$\begin{bmatrix} \mathcal{H} & -\mathcal{D}^T \\ \mathcal{D} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{V} \\ \boldsymbol{\phi} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f} \end{pmatrix} . \tag{3.21}$$

3.3 Mixed Hybrid Finite Volumes for pure anisotropic diffusion

This section shall outline the construction of the specific MVE-like scheme that was implemented specifically in the context of this thesis. The scheme is hereby named *Mixed Hybrid Finite Volumes* (MHFV):

- *Mixed*: different discrete forms exist in different spaces and all constitute problem unknowns, with operators mapping from one space to the other in a MFE spirit [40];
- *Hybrid*: a particular solution strategy will be chosen which is analogous to hybridisation in MFE (see Section 3.3.2);
- *Finite Volumes*: the ultimate goal of the method is to guarantee continuity at the discrete level over each control volume (cell), similar to classical FV.

3.3.1 MHFV local scalar product

At the core of MVE schemes lies the construction of a suitable, material-dependent local scalar product matrix $\mathbb{M}_C$ for the space X^h of discrete fluxes. The MHFV anisotropic diffusion operator is derived in this section based on the second-order accurate MFD scheme presented by Brezzi et al. [42]. It aims at satisfying the following two conditions:

- *Local consistency*: the Gauss-Green formula must hold at the discrete level on each

cell, and must be satisfied exactly for every linear function ψ^1 and for all $\mathbf{W} \in X^h$:

$$\left\langle \left(\mathbb{K}_C \nabla \psi^1\right)^{X^h}, \mathbf{W} \right\rangle_{C,X^h} + (\mathcal{D}\mathbf{W})_C \int_C \psi^1 \, dV = \sum_{F \in \partial C} W_{FC} \frac{1}{|F|} \int_F \psi^1 \, dS \quad (3.22)$$

where $\mathbb{K}_C$ is the cell-averaged diffusivity tensor, $|F|$ the face area, and $(\cdot)^{X^h}$ stands for "de Rham map onto X^h" via (3.5).

- *Stability*: there exist two positive constants s_* and S^* such that, for all $\mathbf{W} \in X^h$ and for every C:

$$s_* \sum_{F \in \partial C} |C| \, W_F^2 \leq \langle \mathbf{W}, \mathbf{W} \rangle_{C,X^h} \leq S^* \sum_{F \in \partial C} |C| \, W_F^2 \, . \quad (3.23)$$

Brezzi et al. [42] provide the following interpretation of these conditions: local consistency (3.22) imposes that operators are built such that the Gauss-Green formula is satisfied exactly at a discrete level for a polynomial scalar function of a given order (linear, in this case) - which is, again, a core concept of mimetic methods - hence establishing the order of accuracy of the scheme; stability (3.23) imposes that $\mathbb{M}_C$ be spectrally equivalent to the scalar matrix $|C|\,\mathbb{I}$ (where $\mathbb{I}$ is the $m_C \times m_C$ identity matrix), i.e. a mass matrix for cell C.

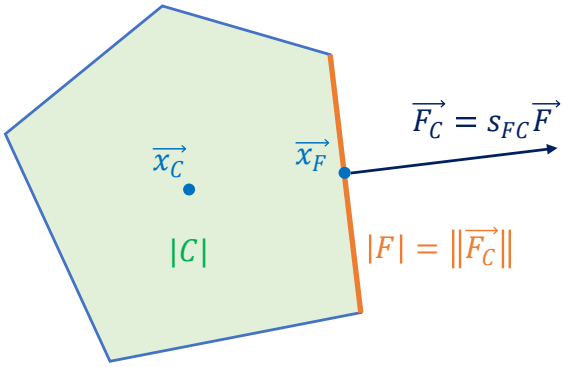


Figure 3.2: MHFV notation for main geometric/inertial quantities.

The scalar product $\mathbb{M}_C$ for MHFV is derived as follows. First a *cell-average* operator for X^h is introduced:

$$\vec{V}_C^{avg} = \sum_{F \in \partial C} \frac{V_{FC} \left(\vec{x}_F - \vec{x}_C\right)}{|C|} \quad (3.24)$$

where $\vec{x}_F$ and $\vec{x}_C$ are respectively the centre of gravity of face F and cell C (Figure 3.2). Justification of expression (3.24) comes from the Stokes formula: denoting by a

superscript i the i-th coordinate of vectors and coordinates, it can be stated that

$$\int_C V^i \, dV = \int_C \vec{V} \cdot \nabla x^i \, dV \ . \tag{3.25}$$

Integration by parts of the right-and side of (3.25) yields

$$\begin{aligned} \int_C \vec{V} \cdot \nabla x^i \, dV &= -\int_C \left(\nabla \cdot \vec{V} \right) x^i \, dV + \sum_{F \in \partial C} \int_F \left(\vec{V} \cdot \vec{n}_{FC} \right) x^i \, dS \\ &\approx -x_C^i \sum_{F \in \partial C} V_{FC} + \sum_{F \in \partial C} x_F^i V_{FC} \end{aligned} \tag{3.26}$$

where the second, approximate equation is exact up to order two, i.e. provided that $\nabla \cdot \vec{V}$ is uniform on a cell and $\vec{V} \cdot \vec{n}_{FC}$ is constant on each face. Definition (3.24) follows directly. A $\mathbb{K}$-dependent scalar product is then defined based on this average operator:

$$\langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h}^{avg} = |C| \left(\mathbb{K}_C^{-1} \vec{V}_C^{avg} \cdot \vec{W}_C^{avg} \right) \ . \tag{3.27}$$

Proof that (3.27) satisfies local consistency (3.22) is provided below.

Proof. Let $\vec{V}$ be the gradient of a linear function ψ^1, i.e. $\vec{V} = \nabla \psi^1$. It follows that $\vec{V}$ is constant and $\vec{V}_C^{avg} = \nabla \psi^1$. Replacing this result and, for $\vec{W}$, the cell-average operator (3.24) in the scalar product (3.27), yields

$$\left\langle \left(\mathbb{K}_C \nabla \psi^1 \right)^{X^h}, \mathbf{W} \right\rangle_{C,X^h}^{avg} = \left(\nabla \psi^1 \cdot \sum_{F \in \partial C} W_{FC} \left(\vec{x}_F - \vec{x}_C \right) \right) \ . \tag{3.28}$$

Furthermore, linearity of ψ^1 implies

$$\begin{aligned} \nabla \psi^1 \cdot \left(\vec{x}_F - \vec{x}_C \right) &= \psi^1 \left(\vec{x}_F \right) - \psi^1 \left(\vec{x}_C \right) \\ &= \frac{1}{|F|} \int_F \psi^1 \, dS - \frac{1}{|C|} \int_C \psi^1 \, dV \end{aligned} \tag{3.29}$$

which, re-injected in (3.28), gives

$$\left\langle \left(\mathbb{K}_C \nabla \psi^1 \right)^{X^h}, \mathbf{W} \right\rangle_{C,X^h}^{avg} = \sum_{F \in \partial C} W_{FC} \frac{1}{|F|} \int_F \psi^1 \, dS - \frac{1}{|C|} \int_C \psi^1 \, dV \sum_{F \in \partial C} W_{FC} \tag{3.30}$$

Considering the definition of the divergence operator (3.13), expression (3.30) corresponds to the local consistency condition (3.22). □

Formulation (3.27) alone is not sufficient to guarantee a stable scalar product. A gradient reconstruction based on (3.24) may vanish and give rise to checker-board modes

[191]. For a trivial example, consider a cubic cell: if W_{FC} happens to be identical on all six faces, then $\vec{W}_C^{avg} = 0$ and the inner product (3.27) goes to zero regardless of $\vec{V}_C^{avg}$. Hence the need for a *stabilisation term* of the form

$$\langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h}^{stab} = \sum_{F \in \partial C} \lambda_{FC} \left(V_{FC} - \vec{V}_C^{avg} \cdot \vec{F}_C \right) \left(W_{FC} - \vec{W}_C^{avg} \cdot \vec{F}_C \right) \tag{3.31}$$

with λ_{FC} being a weighting/scaling factor whose precise expression will be discussed in Section 3.3.4. In (3.31) a shorthand notation for the area vector was also introduced: $\vec{F}_C = \int_F \vec{n}_{FC}\, dS$. The specific form (3.31) for the stabilisation term comes from a *conformity defect* argument: the cell-averaged vector field $\vec{V}_C^{avg}$ is by definition a piecewise-constant reconstruction of $\vec{V}$ based on the discrete degrees of freedom available, i.e. the face fluxes V_{FC}; the de Rham map of such a reconstruction does not correspond to the original degrees of freedom, hence a penalisation term proportional to the defect $\left(V_{FC} - \vec{V}_C^{avg} \cdot \vec{F}_C \right)$ is introduced.

The full $\mathbb{K}$-scalar product thus takes the form

$$\langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h} = \langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h}^{avg} + \langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h}^{stab} \quad . \tag{3.32}$$

One can verify that the addition of the stabilisation term maintains consistency: if $\vec{V} = \nabla\psi^1$ with ψ^1 linear, then $\vec{V}_C^{avg} = \nabla\psi^1$ and therefore $V_{FC} = \int_F \nabla\psi^1 \cdot \vec{n}_{FC}\, dS = \vec{V}_C^{avg} \cdot \vec{F}_C$; hence the stabilisation term in (3.32) vanishes, and the remaining term is consistent as proved above. Concerning stability, several authors (see e.g. Beirão da Veiga et al. [15]) prove that a stabilisation term of the form (3.31) satisfies the stability condition (3.23).

The choice of weight λ_{FC} in (3.31), and therefore the choice of stabilisation term, is not unique: the stabilization is defined up to a multiplicative constant. As a consequence, as Brezzi et al. [42] observe, there is not a unique admissible $\mathbb{M}_C$. Brezzi et al. [41] (and Beirão da Veiga et al. [15, 17] in a more generalised framework) show that, under certain assumptions on the magnitude of the scaling factor in the stabilisation term, and provided that $\mathbb{M}_C$ satisfies consistency and stability, the value obtained via an expression of type (3.32) is equivalent to that coming from a scalar product based on a MFE-like lifting of $\mathbf{V}$ and $\mathbf{W}$ defined on C. In other words, MFD/MVE methods can be interpreted as defining a scalar product (or better, a family of admissible scalar products [42]) between two functions lifted from a discrete space without having to explicitly compute any shape functions. For this very reason these methods are referred to as *virtual*: it can be shown [15, 39] that, if the stabilisation term is large enough, then shape functions virtually exist, but one does not need to define them. As a consequence there are fewer constraints on element shape.

3.3.2 Hybrid pure anisotropic diffusion operator

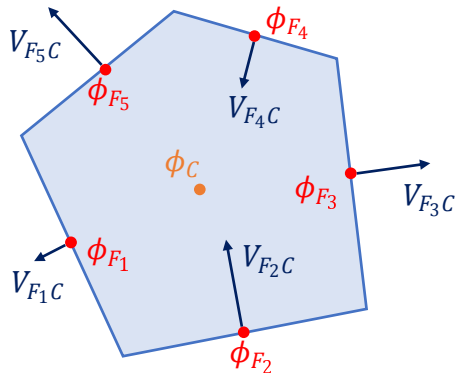


Figure 3.3: Location of MHFV variables in a cell.

An explicit recipe for building the MHFV local scalar product $\mathbb{M}_C$ (minus the choice of weights λ_{FC}, which shall be discussed in Section 3.3.4) was provided in Section 3.3.1, allowing to assemble the discrete pure anisotropic diffusion equation (3.21). In order to circumvent the saddle-point form of (3.21) and recover a SPD system which can be solved more efficiently, it is chosen to follow the MFE *hybridisation* approach as suggested by many [42, 71, 82, 143]. The process begins by noticing that the flux field $\mathbf{V}$ satisfying the weak formulation of the constitutive equation (first equation in (3.20)) is the solution to the optimisation problem

$$\begin{aligned} \mathbf{V} &= \underset{\mathbf{V}\in X^h}{\operatorname{argmin}} \left(\frac{1}{2} \langle \mathbf{V}, \mathbf{V} \rangle_{X^h} - \langle \boldsymbol{\phi}, \mathcal{D}\mathbf{V} \rangle_{Q^h} \right) \\ &= \underset{\mathbf{V}\in X^h}{\operatorname{argmin}} \left(\sum_{C\in\Omega_h} \frac{1}{2} \langle \mathbf{V}, \mathbf{V} \rangle_{C,X^h} - \sum_{C\in\Omega_h} \langle \boldsymbol{\phi}, \mathcal{D}\mathbf{V} \rangle_{C,Q^h} \right) . \end{aligned} \tag{3.33}$$

A *broken space* $\widehat{X^h}$ is then introduced, defined such that a field $\widehat{\mathbf{V}}$ belonging to $\widehat{X^h}$ does not necessarily satisfy flux conservation (3.6) and thus holds two degrees of freedom per face, namely V_{FC+} and V_{FC-}. A formulation equivalent to (3.33) in $\widehat{X^h}$ can be written as

$$\begin{aligned} \widehat{\mathbf{V}} &= \underset{\widehat{\mathbf{V}}\in\widehat{X^h}}{\operatorname{argmin}} \left(\sum_{C\in\Omega_h} \frac{1}{2} \left\langle \widehat{\mathbf{V}}, \widehat{\mathbf{V}} \right\rangle_{C,\widehat{X^h}} - \sum_{C\in\Omega_h} \left\langle \boldsymbol{\phi}, \mathcal{D}\widehat{\mathbf{V}} \right\rangle_{C,Q^h} \right) \\ &\quad s.t. \quad V_{FC+} + V_{FC-} = 0 \quad \forall F \in \Omega_h \end{aligned} \tag{3.34}$$

and adding the flux conservation constraint via a Lagrange multiplier σ_F at each face

leads to the Lagrangian

$$\begin{aligned}\mathcal{L} &= \sum_{C\in\Omega_h}\frac{1}{2}\left\langle\widehat{\mathbf{V}},\widehat{\mathbf{V}}\right\rangle_{C,\widehat{X^h}} - \sum_{C\in\Omega_h}\left\langle\boldsymbol{\phi},\mathcal{D}\widehat{\mathbf{V}}\right\rangle_{C,Q^h} + \sum_{F\in\Omega^h}\sigma_F\left(V_{FC+}+V_{FC-}\right)\\ &= \sum_{C\in\Omega_h}\frac{1}{2}\left\langle\widehat{\mathbf{V}},\widehat{\mathbf{V}}\right\rangle_{C,\widehat{X^h}} - \sum_{C\in\Omega_h}\left\langle\boldsymbol{\phi},\mathcal{D}\widehat{\mathbf{V}}\right\rangle_{C,Q^h} + \sum_{C\in\Omega^h}\sum_{F\in\partial C}\sigma_F V_{FC}\ .\end{aligned} \tag{3.35}$$

Optimality conditions on (3.35) lead to the normal equation

$$\left\langle\widehat{\mathbf{V}},\widehat{\mathbf{W}}\right\rangle_{C,\widehat{X^h}} - \left\langle\boldsymbol{\phi},\mathcal{D}\widehat{\mathbf{W}}\right\rangle_{C,Q^h} + \sum_{F\in\partial C}\sigma_F W_{FC} = 0 \quad \forall\widehat{\mathbf{W}}\in\widehat{X^h},\,\forall C\in\Omega_h\ , \tag{3.36}$$

i.e.

$$\left(\mathbb{M}_C\left(\mathbf{V}\right)_{\partial C},\left(\mathbf{W}\right)_{\partial C}\right) - \phi_C\sum_{F\in\partial C}W_{FC} = -\sum_{F\in\partial C}\sigma_F W_{FC}\ . \tag{3.37}$$

There is an evident analogy between (3.37) and a discrete Gauss-Green formula (or equivalently, a discrete integration by parts), suggesting an interpretation of the Lagrange multiplier σ_F in (3.37) as the face-averaged scalar quantity ϕ_F, or *hybrid variable*:

$$\phi_F = \frac{1}{|F|}\int_F \phi\, dS \quad \forall F\in\Omega_h\ . \tag{3.38}$$

In particular, if φ is linear, then replacing σ_F with ϕ_F in (3.37) yields the exact Gauss-Green formula (3.22). The notation $\widetilde{\boldsymbol{\phi}}$ is hereby introduced to identify the vector holding all face-based hybrid degrees of freedom, such that ϕ_F is the F-th element of $\widetilde{\boldsymbol{\phi}}$:

$$\left(\widetilde{\boldsymbol{\phi}}\right)_F = \phi_F\ , \tag{3.39}$$

while the mapping of a hybrid vector to the faces delimiting cell C shall be denoted as $\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}$. Expression (3.37) allows to define a local discrete constitutive law (first equation in (3.2)) as

$$\left(\mathbf{V}\right)_{\partial C} = \mathbb{M}_C^{-1}\left(\phi_C - \phi_F\right)_{\partial C} \tag{3.40}$$

where

$$\left(\phi_C-\phi_F\right)_{\partial C} = \begin{pmatrix}\phi_C-\phi_{F_1}\\ \phi_C-\phi_{F_2}\\ \cdots\\ \phi_C-\phi_{F_{m_C}}\end{pmatrix}\ . \tag{3.41}$$

Equation (3.40) highlights how $\mathbb{M}_C$ may be interpreted as a local flux operator acting on all MHFV scalar variables belonging to C (Figure 3.3), providing in particular the value of the diffusive flux across each face in function of ϕ_C and $\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}$. By applying the divergence operator (3.13) a local continuity law (second equation in (3.2)) can also be

written as

$$\sum_{F \in \partial C} V_{FC} = |C|\, f_C \ . \tag{3.42}$$

Replacing the expression for fluxes V_{FC} from (3.40) in (3.42) allows to express the cell-averaged ϕ_C in function of the local hybrid variables $\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}$:

$$\phi_C = \frac{\left(\mathbb{M}_C^{-T}\mathbb{1}, \left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}\right) + |C|\, f_C}{\left(\mathbb{M}_C^{-1}\mathbb{1}, \mathbb{1}\right)} \tag{3.43}$$

where $\mathbb{1}$ is the identity vector of cardinality m_C. Re-injecting this in (3.40) yields an expression for all local fluxes $(\mathbf{V})_{\partial C}$ dependent on the hybrid variables only:

$$(\mathbf{V})_{\partial C} = \frac{\left(\mathbb{M}_C^{-T}\mathbb{1}, \left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}\right) + |C|\, f_C}{\left(\mathbb{M}_C^{-1}\mathbb{1}, \mathbb{1}\right)} \mathbb{M}_C^{-1}\mathbb{1} - \mathbb{M}_C^{-1}\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C} \ . \tag{3.44}$$

Finally, the flux variable is eliminated by imposing flux conservation (3.6) over each face. The process - also known as *static condensation* [40] - leads to the linear system

$$\mathcal{F}_{\mathbb{K}} \widetilde{\boldsymbol{\phi}} = \widetilde{\mathbf{f}} \tag{3.45}$$

where: $\widetilde{\boldsymbol{\phi}}$ is the aforementioned hybrid variable (3.38); $\mathcal{F}_{\mathbb{K}}$ is a matrix - shown to be SPD [35] - representing the MHFV *hybrid anisotropic diffusion operator*:

$$(\mathcal{F}_{\mathbb{K}})_{FF'} = \sum_{C \ni F,F'} \left(\frac{\left(\mathbb{M}_C^{-1}\mathbb{1}\right)_F \left(\mathbb{M}_C^{-T}\mathbb{1}\right)_{F'}}{\left(\mathbb{M}_C^{-1}\mathbb{1}, \mathbb{1}\right)} - \left(\mathbb{M}_C^{-1}\right)_{FF'} \right) \ ; \tag{3.46}$$

$\widetilde{\mathbf{f}}$ is the right-hand side arising from the hybridisation process:

$$\left(\widetilde{\mathbf{f}}\right)_F = -f_{C-}\,|C_-|\, \frac{\left(\mathbb{M}_{C-}^{-1}\mathbb{1}\right)_F}{\left(\mathbb{M}_{C-}^{-1}\mathbb{1}, \mathbb{1}\right)} - f_{C+}\,|C_+|\, \frac{\left(\mathbb{M}_{C+}^{-1}\mathbb{1}\right)_F}{\left(\mathbb{M}_{C+}^{-1}\mathbb{1}, \mathbb{1}\right)} \ . \tag{3.47}$$

Operator $\mathcal{F}_{\mathbb{K}}$ acts on stencils based on a non-standard face-to-face connectivity (Figure 3.4), which is in general more complex than a (first-order) FV cell-to-cell one, and it scales with the number of faces in the mesh n_F, which is larger than the number of cells n_C. A larger linear system is arguably the main drawback of MHFV compared to classical FV for pure diffusion.

Once (3.45) is solved for the hybrid field $\widetilde{\boldsymbol{\phi}}$, the solution is used to reconstruct $\boldsymbol{\phi}$ and $\mathbf{V}$ via (3.43) and (3.44) respectively. Notice that the reconstruction of face fluxes through (3.44) will in principle return two values, V_{FC+} and V_{FC-}, since reconstruction

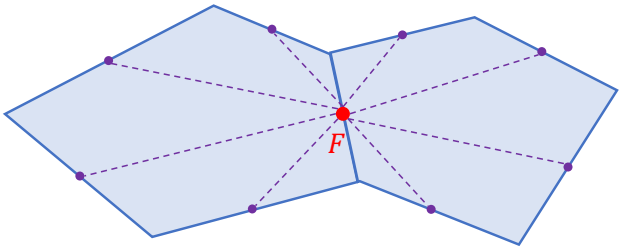


Figure 3.4: Face-to-face stencil for a generic 2D poligonal mesh.

is possible from either side of F. Given that flux conservation is imposed in (3.45), the two should theoretically be equal and opposite, and thus the unsigned flux V_F should be inferable from either one of them. In practice, too large a tolerance on the linear solver might result in slightly different values; in this case V_F is computed as an average between the two.

3.3.3 Inversion of the local scalar product matrix

Some solution strategies for MVE/MFV pure diffusion problems require explicit knowledge of the inverse of $\mathbb{M}_C$ for each cell. In fact, some require knowledge of $\mathbb{M}_C^{-1}$ only, which is notably the case for the MHFV hybrid strategy presented above. Some authors from the MFD community [42, 82] have devised ways of computing $\mathbb{M}_C^{-1}$ directly, while others [71] invert $\mathbb{M}_C$ via direct methods. In the MHFV framework the former approach is feasible. More specifically, in this section it is proven that:

$$\begin{aligned} V_{FC} &= \left(\mathbb{M}_C^{-1}\left(\phi_C - \phi_F\right)_{\partial C}\right)_F \\ &= -\mathbb{K}_C \nabla_C^{\mathcal{G}}\phi \cdot \vec{F}_C - \frac{1}{\lambda_{FC}}\left(\phi_F - \phi_C - \nabla_C^{\mathcal{L},\lambda}\phi \cdot (\vec{x}_F - \vec{x}_C)\right) \ , \end{aligned} \tag{3.48}$$

i.e. the reconstructed flux V_{FC} is a combination of two linearly consistent approximate gradients, $\nabla_C^{\mathcal{G}}$ and $\nabla_C^{\mathcal{L},\lambda}$, respectively based on the Gauss formula and the λ_{FC}-weighted least-squares (LSQ) approach:

$$\nabla_C^{\mathcal{G}}\phi = \frac{1}{|C|}\sum_{F\in\partial C}\phi_F \vec{F}_C \ ; \tag{3.49}$$

$$\nabla_C^{\mathcal{L},\lambda}\phi = \underset{\vec{A}\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{F\in\partial C}\frac{1}{\lambda_{FC}}\left(\phi_F - \phi_C - \vec{A}\cdot(\vec{x}_F - \vec{x}_C)\right)^2 \ . \tag{3.50}$$

As a remarkable practical consequence, the only direct inversion involved in the assembly of $\mathbb{M}_C^{-1}$ is that of a $d \times d$ matrix required to compute the LSQ gradient (3.50), which makes the computational cost of the inversion fully independent of element complexity.

Proof. Let $\overline{\mathbf{W}}$ denote a field such that

$$\left(\overline{\mathbf{W}}\right)_{\partial C} = (\mathbf{W})_{\partial C} - \left(\vec{W}_C^{avg} \cdot \vec{F}_C\right)_{\partial C} \ . \tag{3.51}$$

Then it can be stated that

$$\langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h} = |C| \left(\mathbb{K}_C^{-1} \vec{V}_C^{avg} \cdot \vec{W}_C^{avg}\right) + \left(\left(\overline{\mathbf{V}}\right)_{\partial C}, \left(\overline{\mathbf{W}}\right)_{\partial C}\right)^{\lambda} \tag{3.52}$$

where $(\cdot,\cdot)^{\lambda}$ denotes the λ-weighted canonical inner product in $\mathbb{R}^{m_C}$. From (3.40) it follows that

$$\langle \mathbf{V}, \mathbf{W} \rangle_{C,X^h} = \left(\left(\frac{\phi_C - \phi_F}{\lambda_{FC}}\right)_{\partial C}, (\mathbf{W})_{\partial C}\right)^{\lambda} \ . \tag{3.53}$$

Based on definition (3.51), two subspaces of X^h can be defined. The first, S_L, is a *low frequency* space:

$$S_L = \left\{ (\mathbf{W})_{\partial C} \mid \exists \vec{W}_C^{avg} \in \mathbb{R}^d \ , (\mathbf{W})_{\partial C} = \left(\vec{W}_C^{avg} \cdot \vec{F}_C\right)_{\partial C} \right\} \ , \tag{3.54}$$

i.e. if $(\mathbf{W})_{\partial C}$ belongs to S_L then its high frequency component is null: $\left(\overline{\mathbf{W}}\right)_{\partial C} = \mathbf{0}$, and it holds

$$(\mathbf{W})_{\partial C} = \left(\vec{W}_C^{avg} \cdot \vec{F}_C\right)_{\partial C} \quad \forall \mathbf{W} \in S_L \ . \tag{3.55}$$

The second subspace, S_H, is a *high frequency* space:

$$\begin{aligned} S_H &= \left\{ (\mathbf{W})_{\partial C} \mid \left((\mathbf{W})_{\partial C}, \left(\frac{x_F^i - x_C^i}{\lambda_{FC}}\right)_{\partial C}\right)^{\lambda} = \mathbf{0} \quad \forall i = 1 \cdots d \right\} \\ &= \left(\text{span}\left\{ \left(\frac{x_F^i - x_C^i}{\lambda_{FC}}\right)_{\partial C} , \, i = 1 \cdots d \right\}\right)^{\perp,\lambda} \ ; \end{aligned} \tag{3.56}$$

therefore, considering the cell-average definition (3.24), if $(\mathbf{W})_{\partial C}$ belongs to S_H then its low frequency component is null: $\left(\vec{W}_C^{avg} \cdot \vec{F}_C\right)_{\partial C} = \mathbf{0}$, and it holds:

$$(\mathbf{W})_{\partial C} = \left(\overline{\mathbf{W}}\right)_{\partial C} \quad \forall \mathbf{W} \in S_H \ . \tag{3.57}$$

First a vector $(\mathbf{W})_{\partial C} \in S_L$, i.e. one that satisfies (3.55), is considered. From (3.52) and (3.53) it comes immediately that

$$|C| \left(\mathbb{K}_C^{-1} \vec{V}_C^{avg} \cdot \vec{W}_C^{avg}\right) = \left(\left(\frac{\phi_C - \phi_F}{\lambda_{FC}}\right)_{\partial C}, (\mathbf{W})_{\partial C}\right)^{\lambda} \ . \tag{3.58}$$

Then, using the fact that the average (3.24) is constant, it can be deduced that

$$\begin{aligned} |C|\left(\mathbb{K}_C^{-1}\vec{V}_C^{avg}\cdot\vec{W}_C^{avg}\right) &= \sum_{F\in\partial C}\left(\phi_C-\phi_F\right)\vec{W}_C^{avg}\cdot\vec{F}_C \\ &= -\left(\sum_{F\in\partial C}\phi_F\vec{F}_C\right)\cdot\vec{W}_C^{avg} \end{aligned} \tag{3.59}$$

and thus, by definition of Gauss gradient:

$$\vec{V}_C^{avg} = -\mathbb{K}_C\nabla_C^{\mathcal{G}}\phi\ . \tag{3.60}$$

Secondly, a vector $(\mathbf{W})_{\partial C}\in S_H$, i.e. one that satisfies (3.57), is considered. From (3.52) and (3.53) it comes

$$\begin{aligned} \left((\overline{\mathbf{V}})_{\partial C},(\overline{\mathbf{W}})_{\partial C}\right)^{\lambda} &= \left(\left(\frac{\phi_C-\phi_F}{\lambda_{FC}}\right)_{\partial C},(\mathbf{W})_{\partial C}\right)^{\lambda} \\ &= \left(\left(\frac{\phi_C-\phi_F}{\lambda_{FC}}\right)_{\partial C},(\overline{\mathbf{W}})_{\partial C}\right)^{\lambda} \end{aligned} \tag{3.61}$$

and therefore it must hold

$$(\overline{\mathbf{V}})_{\partial C} = \mathbb{P}_{S_H}^{\perp,\lambda}\left(\frac{\phi_C-\phi_F}{\lambda_{FC}}\right)_{\partial C} \tag{3.62}$$

where $\mathbb{P}_{S_H}^{\perp,\lambda}$ denotes the orthogonal projection on S_H according to the λ-weighted inner product of $\mathbb{R}^{m_C}$. Hence it can be stated that

$$(\overline{\mathbf{V}})_{\partial C} = \left(\frac{\phi_C-\phi_F}{\lambda_{FC}}\right)_{\partial C} - \mathbb{P}_{S_H^{\perp,\lambda}}^{\perp,\lambda}\left(\frac{\phi_C-\phi_F}{\lambda_{FC}}\right)_{\partial C}\ . \tag{3.63}$$

Noticing that

$$S_H^{\perp,\lambda} = \operatorname{span}\left\{\left(\frac{x_F^i-x_C^i}{\lambda_{FC}}\right)_{\partial C}\ ,\ i=1\cdots d\right\} \tag{3.64}$$

which is a consequence of (3.56), the last term in (3.63) may be written in the form

$$\mathbb{P}_{S_H^{\perp,\lambda}}^{\perp,\lambda}\left(\frac{\phi_C-\phi_F}{\lambda_{FC}}\right)_{\partial C} = \left(\vec{B}\cdot\frac{\vec{x}_F-\vec{x}_C}{\lambda_{FC}}\right)_{\partial C}\ . \tag{3.65}$$

Considering the definition of orthogonal projection and that of LSQ gradient (3.50), it

holds

$$\begin{aligned}\vec{B} &= \underset{\vec{A}\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{F\in\partial C} \lambda_{FC}\left(\frac{\phi_C-\phi_F}{\lambda_{FC}} - \vec{A}\cdot\left(\frac{\vec{x}_F-\vec{x}_C}{\lambda_{FC}}\right)\right)^2 \\ &= \underset{\vec{A}\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{F\in\partial C} \frac{1}{\lambda_{FC}}\left(\phi_C-\phi_F-\vec{A}\cdot(\vec{x}_F-\vec{x}_C)\right)^2 \\ &= -\nabla_C^{\mathcal{L},\lambda}\phi\ ,\end{aligned} \tag{3.66}$$

and combining (3.66), (3.65) and (3.63) yields the full expression for the high frequency component:

$$\overline{V}_{FC} = -\frac{1}{\lambda_{FC}}\left(\phi_F-\phi_C-\nabla_C^{\mathcal{L},\lambda}\phi\cdot(\vec{x}_F-\vec{x}_C)\right)\ . \tag{3.67}$$

Finally, combining (3.67), (3.60) and (3.51) leads to the complete one-sided flux expression (3.48). The $\lambda_F C$-weighted LSQ gradient is computed as

$$\nabla_C^{\mathcal{L},\lambda} = \left(\mathbb{X}_C^\lambda\right)^{-1}\vec{b}_{\phi,\lambda} \tag{3.68}$$

where $\mathbb{X}_C^\lambda$ is the $d\times d$ LSQ matrix

$$\left(\mathbb{X}_C^\lambda\right)_{ij} = \sum_{F\in\partial C}\frac{\left(x_F^i-x_C^i\right)\left(x_F^j-x_C^j\right)}{\lambda_{FC}} \tag{3.69}$$

and $\vec{b}_{\phi,\lambda}$ is the LSQ right-hand side

$$\vec{b}_{\phi,\lambda} = \sum_{F\in\partial C}\frac{(\vec{x}_F-\vec{x}_C)(\phi_F-\phi_C)}{\lambda_{FC}}\ . \tag{3.70}$$

This implies that, as anticipated, the only matrix inversion required to construct $\mathbb{M}_C^{-1}$ is that of $\mathbb{X}_C^\lambda$. □

The MHFV one-sided flux definition as expressed in (3.48) highlights a striking similarity with that presented by Eymard et al. [82], who directly define the reconstruction of a stabilised MFV flux - with the option of hybridising - and subsequently show the similarities between their method and the MFD family. In their case, however, they do not make use of a LSQ gradient, using instead the same Gauss gradient for the stabilisation term.

3.3.4 Link with classical Finite Volumes

Result (3.48) also allows to draw a parallelism between MHFV and traditional FV; this is exploited in this section to derive the MHFV-specific definition for the weights λ_{FC} in

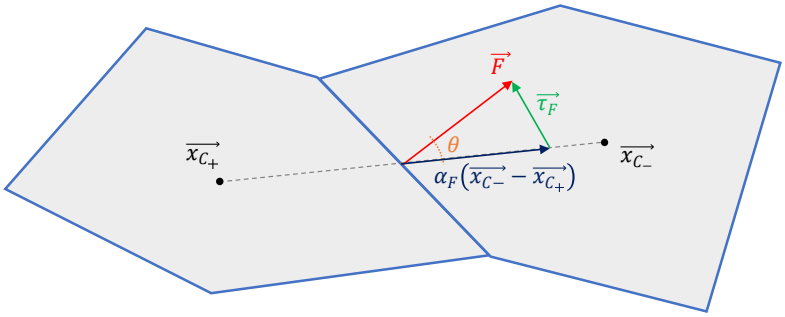


Figure 3.5: Non-orthogonal cells and NOC decomposition of $\vec{F}$.

(3.31). To do so, the starting point is to write the MHFV flux reconstruction (3.48) in a form comparable with a classical FV face flux. In the general case of a non-orthogonal mesh, FV build face fluxes by means of the *Non-Orthogonal Corrector* (NOC) approach [171] as follows. First the face vector $\vec{F}$ is split (Figure 3.5) as

$$\vec{F} = \alpha_F \left(\vec{x}_{C-} - \vec{x}_{C+}\right) + \vec{\tau}_F \tag{3.71}$$

where α_F is a positive coefficient. Then, considering for ease of exposition the isotropic diffusion case (i.e. $\mathbb{K}_C = k_C \mathbb{I}$ where k_C is the cell-averaged diffusivity), the diffusive flux across F is computed as

$$V_F^{FV} = k_F \left(\underbrace{\alpha_F \left(\phi_{C+} - \phi_{C-}\right)}_{\text{orthogonal term}} - \underbrace{\text{“}\nabla_F \phi \cdot \vec{\tau}_F\text{”}}_{\text{NOC}} \right) \tag{3.72}$$

with k_F being some form of face-averaged diffusivity, typically as simple as a weighted two-point arithmetic average between k_{C+} and k_{C-}. The gradient for the NOC in (3.72) may be computed via interpolated nodal values, as in diamond schemes [60], or via interpolated cell gradients [135].

It is now assumed that the LSQ weights in (3.50) can be written in the form

$$\lambda_{FC} = \frac{\mu_F}{k_C} \tag{3.73}$$

where μ_F is a weighting factor for face F. Then, under the same assumption of isotropic diffusivity, the MHFV one-sided flux reconstruction (3.48) simplifies to

$$V_{FC} = k_C \left(-\nabla_C^{\mathcal{G}} \phi \cdot \vec{F}_C - \frac{1}{\mu_F} \left(\phi_F - \phi_C - \nabla_C^{\mathcal{L},\lambda} \phi \cdot \left(\vec{x}_F - \vec{x}_C\right) \right) \right) \ . \tag{3.74}$$

Applying flux conservation (3.6) to (3.74) leads to

$$\begin{aligned}\phi_F &= \frac{k_{C+}\phi_{C+} + k_{C-}\phi_{C-}}{k_{C+} + k_{C-}} \\ &- \mu_F \frac{\left(\nabla^{\mathcal{G}}_{C+}\phi \cdot \vec{F}_{C+} + \nabla^{\mathcal{G}}_{C-}\phi \cdot \vec{F}_{C-}\right)}{k_{C+} + k_{C-}} \\ &+ \frac{k_{C+}\nabla^{\mathcal{L},\lambda}_{C+}\phi \cdot (\vec{x}_F - \vec{x}_{C+}) + k_{C-}\nabla^{\mathcal{L},\lambda}_{C-}\phi \cdot (\vec{x}_F - \vec{x}_{C-})}{k_{C+} + k_{C-}} .\end{aligned} \tag{3.75}$$

A remark: the leading term in (3.75) is an average value between ϕ_{C+} and ϕ_{C-} weighted by the respective cell-averaged k; there is a remarkable resemblance with *Roe's average interface state* used in linearised *Riemann solvers* [205] for compressible flows, where the weighting is based instead on a cell-averaged $\sqrt{\rho}$ (ρ being the density) and is devised specifically to capture correctly the propagation of discontinuities and/or shock waves in the solution field. This parallelism further highlights how MVE-like methods are suitable for cases where material properties are discontinuous across faces.

Re-injecting now (3.75) in (3.74), yields

$$V_F = V_{FC+} = \frac{2k_{C+}k_{C-}}{k_{C+} + k_{C-}} \left(\frac{1}{2\mu_F} (\phi_{C+} - \phi_{C-}) - NOC_F \right) \tag{3.76}$$

with

$$\begin{aligned}NOC_F &= \frac{\nabla^{\mathcal{G}}_{C+}\phi + \nabla^{\mathcal{G}}_{C-}\phi}{2} \cdot \vec{F} \\ &- \frac{1}{2\mu_F} \left(\nabla^{\mathcal{L},\lambda}_{C+}\phi \cdot (\vec{x}_F - \vec{x}_{C+}) + \nabla^{\mathcal{L},\lambda}_{C-}\phi \cdot (\vec{x}_{C-} - \vec{x}_F) \right)\end{aligned} \tag{3.77}$$

(expression (3.77) makes use of the trivial identity: $\vec{F}_{C-} = -\vec{F}_{C+} = -\vec{F}$). There are evident similarities between the MHFV flux (3.76) and the FV flux (3.72). In particular, one can identify $k_F = \frac{2k_{C+}k_{C-}}{k_{C+}+k_{C-}}$ (harmonic average of the diffusion coefficient) and $\alpha_F = \frac{1}{2\mu_F}$. This suggests a FV-inspired expression for the LSQ weights, namely by defining μ_F - and thus λ_{FC} through (3.73) - such that it reproduces an existing FV formulation of the NOC factor α_F. For example, the NOC approach known as *over-relaxed* [135] gives:

$$\alpha_F = \frac{|F|^2}{(\vec{x}_{C-} - \vec{x}_{C+}) \cdot \vec{F}} \quad \rightarrow \quad \lambda_{FC} = \frac{(\vec{x}_{C-} - \vec{x}_{C+}) \cdot \vec{F}}{2k_C |F|^2} . \tag{3.78}$$

One may easily verify that λ_{FC} as defined in (3.78) is dimensionally consistent when replaced in (3.48). Notice also that, when λ_{FC} is defined this way, the MHFV stabilisation term scales with the inverse of the diffusivity: this mirrors the argument typically brought forward by MFD/MVE scholars (see e.g. [42, 49]) of scaling the stabilisation term by a characteristic value of $\mathbb{K}_C^{-1}$ - typically trace $\left(\mathbb{K}_C^{-1}\right)$ - in order to obtain the desired spectral properties of $\mathbb{M}_C$. In fact, one could have alternatively pre-multiplied the stabilisation

term (3.31) by some $\mathbb{K}_C^{-1}$-scaling factor and then derived a purely geometric λ_{FC}. The definition of a weight that allows to retrieve under specific conditions a classical FV scheme is a strategy already known by the MFV community - see e.g. Piar et al. [191] and Eymard et al. [82], who retrieve the classical two-point flux scheme for 2D triangular and rectangular grids. However, the idea of defining NOC-inspired weights is an original aspect of the MHFV diffusion scheme presented in this thesis.

The following symmetric weight types - where by symmetry is intended $\lambda_{FC+} = \lambda_{FC-}$ for a constant diffusivity field - are implemented in MHFV. They are derived from and named after the NOC formulations proposed by Jasak [135]:

- *Orthogonal Symmetric* (ORTS):

$$\lambda_{FC} = \gamma_F \zeta_C \frac{\|\vec{x}_{C-} - \vec{x}_{C+}\|}{|F|} \tag{3.79}$$

- *Over-Relaxed Symmetric* (OVRS):

$$\lambda_{FC} = \gamma_F \zeta_C \frac{\left|(\vec{x}_{C-} - \vec{x}_{C+}) \cdot \vec{F}\right|}{|F|^2} \tag{3.80}$$

- *Minimal Symmetric* (MINS):

$$\lambda_{FC} = \gamma_F \zeta_C \frac{\|\vec{x}_{C-} - \vec{x}_{C+}\|^2}{\left|(\vec{x}_{C-} - \vec{x}_{C+}) \cdot \vec{F}\right|} \tag{3.81}$$

The γ_F coefficient is set to $\frac{1}{2}$ for internal faces and 1 for boundary faces, since at boundaries $\vec{x}_{C-}$ is replaced by $\vec{x}_F$. The scaling coefficient ζ_C is defined as

$$\zeta_C = \frac{\text{trace}\left(\mathbb{K}_C^{-1}\right)}{d} \tag{3.82}$$

so that one retrieves $\zeta_C = \frac{1}{k_C}$ in case of isotropic diffusivity. Where dot products are present, their absolute value is taken in order to avoid negative weights in case of non-convex cells. Special care must be taken when using the OVRS formulation (3.80), as it vanishes when $(\vec{x}_{C-} - \vec{x}_{C+})$ and $\vec{F}_C$ are orthogonal (which could be the case in a non-convex cell) thus introducing a singularity in (3.48). A similar observation holds for the MINS formulation (3.81): in this case the singularity is in the weight expression itself. These weights should not be used in combination with non-convex grids.

The following additional non-symmetric versions are also implemented. They use

face-to-cell centre distances (rather than cell-to-cell) to attain a better weighting scheme on grids featuring large jumps in cell width:

- *Orthogonal Non-Symmetric* (ORTN):

$$\lambda_{FC} = \zeta_C \frac{\|\vec{x_F} - \vec{x_C}\|}{|F|} \tag{3.83}$$

- *Over-Relaxed Non-Symmetric* (OVRN):

$$\lambda_{FC} = \zeta_C \frac{\left|(\vec{x}_F - \vec{x}_C) \cdot \vec{F}_C\right|}{|F|^2} \tag{3.84}$$

Finally, a second version of the non-symmetric over-relaxed weight (3.84) is derived which takes into account the anisotropy of $\mathbb{K}$, namely by giving more weight to faces across which diffusion is more important:

- *Over-Relaxed Non-Symmetric with Anisotropy* (OVRNA):

$$\lambda_{FC} = \frac{\left|(\vec{x}_F - \vec{x}_C) \cdot \vec{F}_C\right|}{\left|\mathbb{K}_C \vec{F} \cdot \vec{F}\right|} \tag{3.85}$$

In case of Cartesian isotropic mesh and isotropic diffusivity all weight formulations are identical and reduce to ORTS (3.79), and additional symmetry properties entail that $\nabla_C^{\mathcal{G}} = \nabla_C^{\mathcal{L}}$; as a consequence, a classical two-point flux is retrieved through (3.76).

A link between MHFV and FV has been highlighted and exploited for deriving the weighting schemes above. However this does not imply that the drawbacks of FV NOCs (Section 2.4) will affect the MHFV scheme: a) the *deferred correction* approach [86] and subsequent limiting [171] do not apply in this context, since in MHFV both the consistency and stabilisation components of fluxes are treated implicitly; b) the SPD property of the global hybrid operator (3.45) is not affected since the local scalar products are tailored to that end in the mimetic spirit, regardless of how the λ_{FC} weights are computed. This is not always the case in classical FV: if the local flux reconstructions used for the NOC are not symmetric on general grids (such as Multipoint Flux Approximations (MPFA) [1]), then the resulting system is not symmetric - thus violating a fundamental property of the continuous problem - and convergence properties are impaired [140].

3.3.5 Boundary conditions

In the hybridised MHFV framework, non-homogeneous Dirichlet boundaries are implemented by face-averaging the forced boundary value f_D and imposing it to the corresponding hybrid variable. Hence the hybrid solution field $\widetilde{\boldsymbol{\phi}}$ may be reduced to internal faces only, with boundary faces extracted from the system and boundary values included in the right-hand side where present. In practice, the degrees of freedom corresponding to Dirichlet boundary faces are kept in the system - for the practical purpose of avoiding resizing/renumbering matrices and arrays - and their corresponding equation is set to $\phi_F = f_{F,D}$, where

$$f_{F,D} = \frac{1}{|F|} \int_F f_D \, dS \ , \tag{3.86}$$

although the equation is typically scaled in order to maintain the corresponding matrix entries within their original order of magnitude. Boundary values are moved to the right-hand side of any other equation where they appear, and their corresponding entries in $\mathcal{F}_{\mathbb{K}}$ zeroed, which ensures that the symmetry of the operator is maintained. To formalise: introducing Ω_h^I as the subspace of all internal + non-Dirichlet boundary faces and $\partial\Omega_h^D$ as that of Dirichlet faces, the hybrid matrix and right-hand side are modified such that the following equations are solved:

$$(\mathcal{F}_{\mathbb{K}})_{FF} \, \phi_F = (\mathcal{F}_{\mathbb{K}})_{FF} \, f_{F,D} \quad \forall F \in \partial\Omega_h^D \tag{3.87}$$

and

$$\sum_{F' \in \Omega_h^I} (\mathcal{F}_{\mathbb{K}})_{FF'} \, \phi_{F'} = \left(\widetilde{\mathbf{f}}\right)_F - \sum_{F' \in \partial\Omega_h^D} (\mathcal{F}_{\mathbb{K}})_{FF'} \, f_{F',D} \quad \forall F \in \Omega_h^I \ . \tag{3.88}$$

Notice that this is a case of *hard Dirichlet* boundary condition; therefore, all considerations from Section 2.2.4 on the matter do apply to the MHFV scheme.

Neumann boundary conditions impose that $-\mathbb{K}\nabla\phi \cdot \vec{n} = f_N$ across the Neumann boundary $\partial\Omega_h^N$. By computing the imposed flux through a Neumann face as

$$f_{F,N} = \int_F f_N \, dS \tag{3.89}$$

and making use of the one-sided flux expression (3.44), this amounts to imposing:

$$\left(\frac{\left(\mathbb{M}_C^{-1^T}\mathbb{1}, \left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}\right) + f_C \, |C|}{\left(\mathbb{M}_C^{-1}\mathbb{1}, \mathbb{1}\right)} \mathbb{M}_C^{-1}\mathbb{1} - \mathbb{M}_C^{-1}\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C} \right)_F = f_{F,N} \quad \forall F \in \partial\Omega_h^N \ . \tag{3.90}$$

Finally, Robin-type boundary conditions impose that $a\phi - \mathbb{K}\nabla\phi \cdot \vec{n} = f_R$, where a is a

scalar coefficient, over the Robin boundary $\partial\Omega_h^R$. By defining

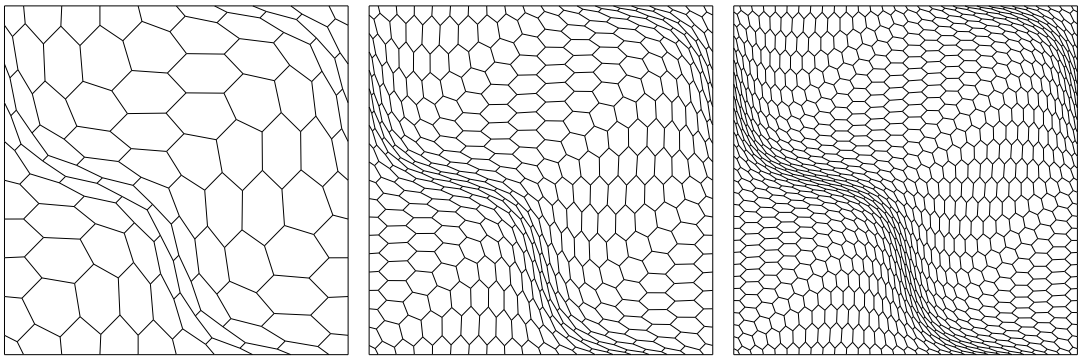
$$f_{F,R} = \int_F f_R \, dS \quad \text{and} \quad a_F = \int_F a \, dS \ , \tag{3.91}$$

the resulting equation is analogous to the Neumann case with the additional quantity a_F on the central coefficient:

$$a_F \phi_F + \left(\frac{\left(\mathbb{M}_C^{-1^T} \mathbb{1}, \left(\widetilde{\boldsymbol{\phi}} \right)_{\partial C} \right) + f_C \, |C|}{\left(\mathbb{M}_C^{-1} \mathbb{1}, \mathbb{1} \right)} \mathbb{M}_C^{-1} \mathbb{1} - \mathbb{M}_C^{-1} \left(\widetilde{\boldsymbol{\phi}} \right)_{\partial C} \right)_F = f_{F,R} \quad \forall F \in \partial\Omega_h^R \ . \tag{3.92}$$

3.4 Validation of MHFV for pure anisotropic diffusion problems

3.4.1 h-convergence for pure anisotropic diffusion



(a) $h \approx 0.1$ (b) $h \approx 0.05$ (c) $h \approx 0.03$

Figure 3.6: Refinement sequence for a 2D polygonal distorted mesh.

A first validation of the MHFV scheme is carried out by verifying h-convergence on a 2D pure anisotropic diffusion test case. The benchmark case proposed by Kuznetsov et al. [143] is considered: the domain is the unit square $\Omega =]0,1[\times]0,1[$; the diffusion tensor is taken as

$$\mathbb{K}(x,y) = \begin{pmatrix} (x+1)^2 + y^2 & -xy \\ -xy & (x+1)^2 \end{pmatrix} \ ; \tag{3.93}$$

the source term is calculated such that the exact solution is:

$$\phi_{ex}(x,y) = x^3 y^2 + x \sin(2\pi xy) \sin(2\pi y) \ . \tag{3.94}$$

Dirichlet boundary conditions are applied throughout. The OVRNA formulation (3.85) is selected for the stabilisation weights λ_{FC}. The literature suggests testing on a sequence of progressively refined polygonal unstructured meshes based on the dual to a Voronoi tessellation, generated via the algorithm described by Beirão da Veiga et al. [18] and shown in Figure 3.6 for three different values of refinement h. Such meshes feature strongly skewed/non-orthogonal cells, making them suitable to test the capabilities of mimetic methods. Testing over distorted meshes is deemed relevant for two reasons. Firstly, while it is true that modern commercial software is capable of generating high quality meshes under most circumstances, Eymard et al. [82] observe that in some industrial applications (e.g. hydrogeology, oil engineering) the inherently complex geometry of the domain implies that mesh quality in a FV-sense is not trivial to achieve. Secondly - and in correlation with the context of this thesis - a mesh-independent scheme means that, when performing shape optimisation cycles, the quality requirements imposed on mesh morphers (see Section 6.3) can be relaxed. For the sake of completeness, the same test is also run on a sequence of fully Cartesian meshes. All linear systems are solved with a direct solver so as to eliminate any effect due to incomplete linear solves.

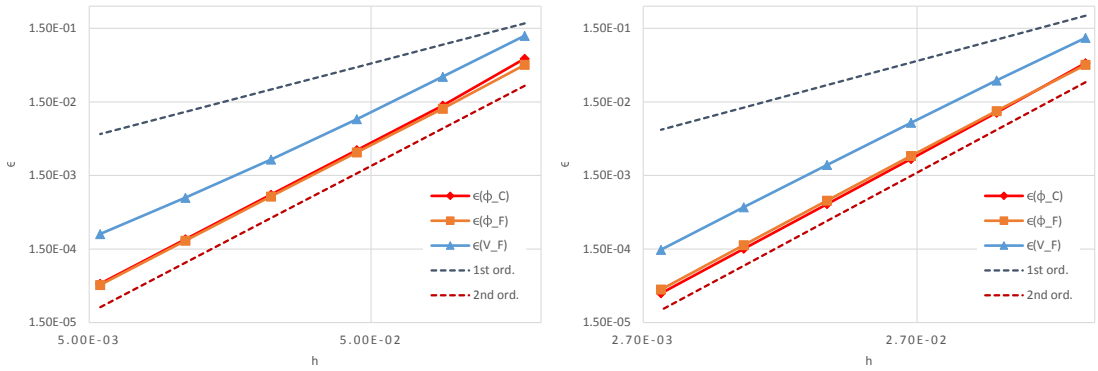
The error ϵ on each MHFV variable is estimated via the following L^2 norms scaled by the exact solution ϕ_{ex}:

$$\epsilon\left(\phi_C\right) = \sqrt{\frac{\sum\limits_{C\in\Omega_h} |C|\left(\boldsymbol{\phi}-\boldsymbol{\phi}_{ex}\right)_C^2}{\sum\limits_{C\in\Omega_h} |C|\left(\boldsymbol{\phi}_{ex}\right)_C^2}} \quad \text{where } \left(\boldsymbol{\phi}_{ex}\right)_C = \phi_{ex}\left(\vec{x}_C\right) \ ; \tag{3.95}$$

$$\epsilon\left(\phi_F\right) = \sqrt{\frac{\sum\limits_{F\in\Omega_h} |F|\left(\widetilde{\boldsymbol{\phi}}-\widetilde{\boldsymbol{\phi}}_{ex}\right)_F^2}{\sum\limits_{F\in\Omega_h} |F|\left(\widetilde{\boldsymbol{\phi}}_{ex}\right)_F^2}} \quad \text{where } \left(\widetilde{\boldsymbol{\phi}}_{ex}\right)_F = \phi_{ex}\left(\vec{x}_F\right) \ ; \tag{3.96}$$

$$\begin{aligned} \epsilon\left(V_F\right) &= \sqrt{\frac{\sum\limits_{C\in\Omega_h} \left(\mathbb{M}_C\left(\mathbf{V}-\mathbf{V}_{ex}\right)_{\partial C}, \left(\mathbf{V}-\mathbf{V}_{ex}\right)_{\partial C}\right)}{\sum\limits_{C\in\Omega_h} \left(\mathbb{M}_C\left(\mathbf{V}_{ex}\right)_{\partial C}, \left(\mathbf{V}_{ex}\right)_{\partial C}\right)}} \\ \text{where } \left(\mathbf{V}_{ex}\right)_{\partial C} &= \left(-\left(\mathbb{K}\left(\vec{x}_F\right)\nabla\phi_{ex}\left(\vec{x}_F\right)\right)\cdot\vec{F}_C\right)_{\partial C} \ . \end{aligned} \tag{3.97}$$

Results are plotted in Figure 3.7 against a value h indicative of mesh coarseness, hereby taken as the maximum cell-to-cell centre distance found in each mesh. Solution field contours for selected h values are shown in Figure 3.8. Error values and correspond-



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 3.7: Pure anisotropic diffusion: h-convergence.

ing convergence rates are reported in Table 3-A and 3-B for polygonal and Cartesian meshes, respectively. Here, and in the remainder of this work, the convergence rate at the n-th refinement level is computed via the formula

$$\text{Rate}^n = \frac{\log \frac{\epsilon^n}{\epsilon^{n-1}}}{\log \frac{h^n}{h^{n-1}}} \,. \tag{3.98}$$

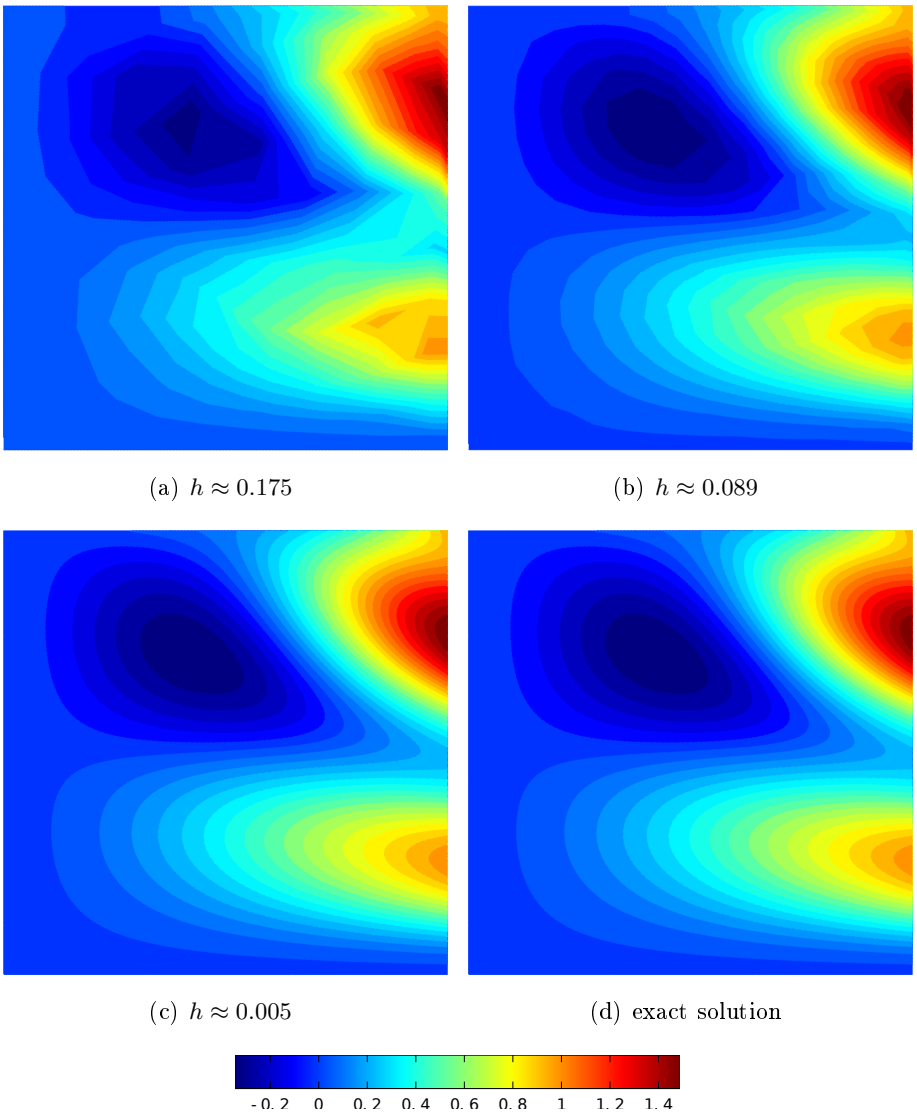
As expected, both mesh types display second-order convergence for scalar values (both cell-averaged and hybrid). Super-convergence is also observed on the flux variable, despite it being formally first-order accurate. This is especially evident on the Cartesian mesh sequence, which is attributable to the cancellation of truncation error induced by the additional symmetries on such regular grids. Results are in perfect agreement with the theoretical findings reported by Brezzi et al. [41].

Table 3-A: Pure anisotropic diffusion - polygonal distorted mesh: errors and convergence rates.

h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
1.751 E^{-1}	5.797 E^{-2}	–	4.777 E^{-2}	–	1.188 E^{-1}	–
8.967 E^{-2}	1.337 E^{-2}	2.192	1.210 E^{-2}	2.052	3.301 E^{-2}	1.914
4.449 E^{-2}	3.313 E^{-3}	1.991	3.089 E^{-3}	1.948	8.725 E^{-3}	1.899
2.206 E^{-2}	8.202 E^{-4}	1.990	7.773 E^{-4}	1.967	2.466 E^{-3}	1.801
1.097 E^{-2}	2.035 E^{-4}	1.995	1.945 E^{-4}	1.983	7.472 E^{-4}	1.709
5.471 E^{-3}	5.064 E^{-5}	1.999	4.864 E^{-5}	1.992	2.398 E^{-4}	1.634

Table 3-B: Pure anisotropic diffusion - Cartesian mesh: errors and convergence rates.

h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
1.111 E^{-1}	5.039 E^{-3}	–	4.778 E^{-3}	–	1.112 E^{-1}	–
5.263 E^{-3}	1.071 E^{-3}	2.073	1.127 E^{-3}	1.933	2.932 E^{-3}	1.784
2.564 E^{-3}	2.518 E^{-3}	2.013	2.762 E^{-3}	1.955	7.818 E^{-3}	1.838
1.266 E^{-3}	6.129 E^{-4}	2.002	6.830 E^{-4}	1.980	2.086 E^{-3}	1.872
6.289 E^{-3}	1.513 E^{-4}	1.999	1.697 E^{-4}	1.990	5.546 E^{-4}	1.893
3.135 E^{-3}	3.759 E^{-5}	2.000	4.227 E^{-5}	1.997	1.468 E^{-4}	1.909

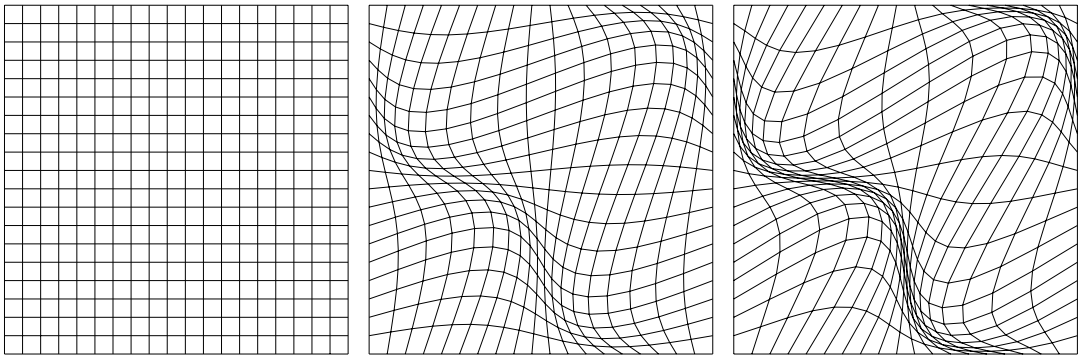


(a) $h \approx 0.175$ (b) $h \approx 0.089$

(c) $h \approx 0.005$ (d) exact solution

Figure 3.8: Pure anisotropic diffusion: solution field ϕ for different refinement values - polygonal distorted mesh.

3.4.2 Comparison of weight types



(a) $\min(\cos\theta) = 1$ (b) $\min(\cos\theta) \approx 0.16$ (c) $\min(\cos\theta) \approx 0.02$

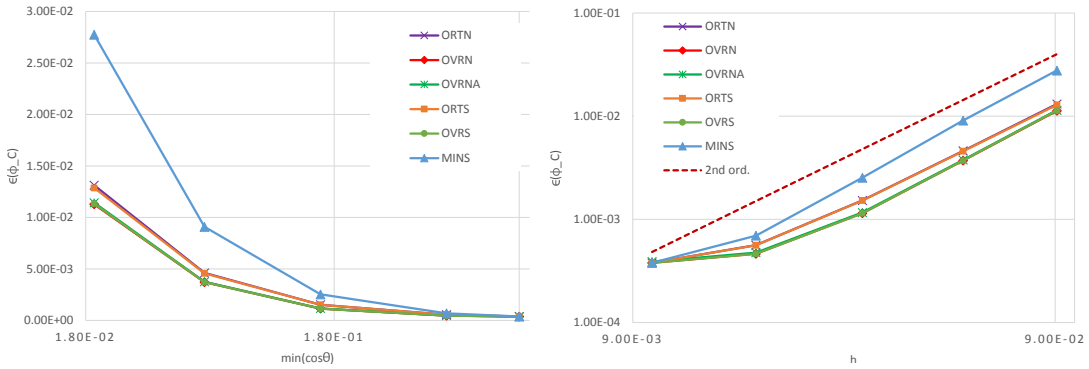
Figure 3.9: Distortion sequence for a 2D quadrilateral mesh.

In this section the various formulations for the the λ_{FC} weights, specific to MHFV and derived in Section 3.3.4, are tested. Since they are inspired by NOC formulations from classical FV, it is interesting to compare how they perform on progressively non-orthogonal meshes. The test case is the same as in the previous section, solved over an initially Cartesian 100×100 mesh to which a progressive distortion is applied as shown in Figure 3.9. All linear systems are solved with a direct solver.

The minimum value of $\cos\theta$ found in the mesh, with θ being the angle between the face normal $\vec{F}$ and the distance between the centres of the two cells adjacent to F (see Figure 3.5), is taken as non-orthogonality indicator: smaller values indicate higher non-orthogonality, with $\min(\cos\theta) = 1$ being a perfectly orthogonal mesh.

Results are reported in Figure 3.10(a), where the normalised L^2 norm of the error on the cell-averaged scalar ϕ_C is plotted against $\min(\cos\theta)$. It can be observed that all weights perform comparably: the over-relaxed formulations (OVRS, OVRN, OVRNA) show a slight superiority (in agreement with the findings of Jasak [135] in FV), while the minimal symmetric (MINS) exhibits a more visible deteriorating effect over more distorted meshes. For each formulation there is no significant difference between the symmetric and non-symmetric version.

At a first glance the plot appears to reveal a rather strong correlation between error and mesh orthogonality, which would negate the benefits of mimetic formulations, but the effect is not due to mesh distortion alone. It can be seen in Figure 3.9 how the algorithm used to distort the mesh doesn't only affect mesh orthogonality, but also local mesh width: the most distorted meshes in the sequence feature a significant expansion ratio (largest-to-smallest cell volume) which is likely to have an impact on the error. For a



(a) error vs. orthogonality

(b) error vs. h

Figure 3.10: Pure anisotropic diffusion: errors for different weight types on a sequence of progressively distorted meshes.

more fair comparison the h-convergence over the same mesh sequence is analysed in Figure 3.10(b): it can be observed how all weight formulations maintain the nominal second-order accuracy of MHFV up until the last mesh in the sequence, where $\min(\cos\theta) \approx 1.95\ \mathrm{E}^{-2}$ corresponds to $\max(\theta) \approx 88.88°$, indicating the presence of near-degenerate cells in the grid. This suggests that the trends observed in Figure 3.10(a) are solely due to mesh size and independent of grid distortion.

Chapter 4

MHFV Convection-Diffusion-Reaction

4.1 Addition of convective fluxes

The next step towards Navier-Stokes is the discretisation of the *convection-diffusion-reaction equation* for a scalar ϕ:

$$\nabla \cdot \left(-\mathbb{K} \nabla \phi + \vec{U} \phi \right) + \eta \phi = f \quad \text{in } \Omega \tag{4.1}$$

where $\vec{U}$ is the convecting velocity field and η the scalar reaction coefficient. As mentioned in Section 3.1, while the investigation of mimetic schemes has focused extensively on the pure diffusion problem, solutions for general elliptic problems including convection and reaction are more recent. From the the MFD/MVE community a second-order accurate convection-reaction scheme for low-Peclet (diffusion-dominated) problems was first proposed by Cangiani et al. [50] and later extended to high-Peclet (convection-dominated) regimes by Beirão da Veiga et al. [18] and Droniou [70]. Very recently, SUPG-like (see Section 4.2.1) stabilisation techniques for high-Peclet conditions have surfaced [24, 159]. From the MFV/HMM community, an interesting implementation based on the FV MUSCL scheme (*Monotonic Upwind Scheme for Conservation Laws* [227]) has been proposed by Piar et al. [191], while adaptations of other classical FV strategies - such as first-order upwinding and the Scharfetter-Gummel scheme [208] - have been investigated by Droniou and Eymard [72] and Droniou [70]. A synergy between the two communities has led to a unified approach [18, 73] for the handling of convection terms, an option that shall fit the MHFV framework as well (Section 4.1.4).

The starting point is common to all methods and, as for the pure diffusion case, it consists in rewriting (4.1) in mixed formulation:

$$\begin{cases} \vec{V} = -\mathbb{K}\nabla\phi + \vec{U}\phi \\ \nabla \cdot \vec{V} + \eta\phi = f \end{cases} \quad \text{in } \Omega \ . \tag{4.2}$$

Intuitively, the discrete counterpart of $\vec{V}$ in X^h shall be a flux which accounts for both diffusive and convective components. Different approaches to how the convective term is included in the discrete problem give rise to different convection schemes. The main approaches are discussed in the following sections and adapted to MHFV.

4.1.1 Centred schemes

The scheme that most naturally fits in a MVE context is based on the discrete variational formulation of the mixed convection-diffusion-reaction equation (4.2) as presented by e.g. [17, 50, 70]:

Find $\boldsymbol{\phi} \in Q^h$, $\mathbf{V} \in X^h$ *such that:*

$$\begin{aligned} &\langle \mathbf{V}, \mathbf{W} \rangle_{X^h} - \langle \boldsymbol{\phi}, \mathcal{D}\mathbf{W} \rangle_{Q^h} - a^{cnv}(\mathbf{U}, \mathbf{W}, \boldsymbol{\phi}) = \mathbf{0} && \forall \mathbf{W} \in X^h \\ &\langle \mathcal{D}\mathbf{V}, \boldsymbol{\psi} \rangle_{Q^h} + \langle \boldsymbol{\eta\phi}, \boldsymbol{\psi} \rangle_{Q^h} = \langle \mathbf{f}, \boldsymbol{\psi} \rangle_{Q^h} && \forall \boldsymbol{\psi} \in Q^h \end{aligned} \tag{4.3}$$

where $\boldsymbol{\eta\phi}$ identifies a vector in Q^h defined such that $(\boldsymbol{\eta\phi})_C = \eta_C \phi_C$, with η_C being the de Rham map of η onto Q^h:

$$\eta_C = \frac{1}{|C|} \int_C \eta \, dV \quad \forall C \in \Omega_h \ , \tag{4.4}$$

and $\mathbf{U}$ denotes the de Rham map of $\vec{U}$ onto X^h:

$$(\mathbf{U})_F = U_{FC} = \int_F \vec{U} \cdot \vec{n}_{FC} \, dS \quad \forall F \in \Omega_h \ . \tag{4.5}$$

In (4.3), $a^{cnv}(\mathbf{U}, \mathbf{W}, \boldsymbol{\phi})$ is a trilinear form representing the contribution of the convective term to the global scalar product. It must be defined as a consistent discrete representation of $\int_\Omega \phi \left(\mathbb{K}^{-1}\vec{U} \cdot \vec{W} \right) dV$. As Droniou [70] observes, the MVE scheme proposed by

Cangiani et al. [50] can be interpreted as being based on the approximation:

$$\begin{aligned}\int_\Omega \phi \left(\mathbb{K}^{-1}\vec{U}\cdot\vec{W}\right) dV &= \sum_{C\in\Omega_h}\int_C \phi \left(\mathbb{K}^{-1}\vec{U}\cdot\vec{W}\right) dV \\ &\approx \sum_{C\in\Omega_h} \phi_C \int_C \left(\mathbb{K}^{-1}\vec{U}\cdot\vec{W}\right) dV \\ &\approx \sum_{C\in\Omega_h} \phi_C \left\langle \mathbf{U},\mathbf{W}\right\rangle_{C,X^h} \\ &\approx \sum_{C\in\Omega_h} \left\langle \widehat{\mathbf{V}^{cnv}},\widehat{\mathbf{W}}\right\rangle_{C,\widehat{X^h}}\end{aligned} \tag{4.6}$$

where $\widehat{\mathbf{V}^{cnv}}$ denotes a field representing the convective flux in the broken space $\widehat{X^h}$ such that

$$V_{FC}^{cnv} = \phi_C U_{FC} \ . \tag{4.7}$$

Imposing flux conservation to the total flux $\mathbf{V}$ (convective + diffusive) through the same Lagrange multiplier approach from Section 3.3.2, optimality conditions on the constitutive equation in (4.3) yield the normal equation

$$\left\langle \widehat{\mathbf{V}},\widehat{\mathbf{W}}\right\rangle_{C,\widehat{X^h}} - \left\langle \widehat{\mathbf{V}^{cnv}},\widehat{\mathbf{W}}\right\rangle_{C,\widehat{X^h}} - \left\langle \boldsymbol{\phi},\mathcal{D}\widehat{\mathbf{W}}\right\rangle_{C,\mathcal{Q}^h} + \sum_{F\in\partial C}\phi_F W_{FC} = 0 \quad \forall\widehat{\mathbf{W}}\in\widehat{X^h} \tag{4.8}$$

with the hybrid variable ϕ_F acting as Lagrange multiplier as in the pure diffusion case. A local flux reconstruction can be deduced from (4.8):

$$(\mathbf{V})_{\partial C} = \mathbb{M}_C^{-1}\left(\phi_C - \phi_F\right)_{\partial C} + \varphi_C\left(\mathbf{U}\right)_{\partial C} \tag{4.9}$$

i.e.

$$V_{FC} = V_{FC}^{diff} + V_{FC}^{cnv} \tag{4.10}$$

where V_{FC}^{diff} is the diffusive flux (3.40) derived in Section 3.3.2, and V_{FC}^{cnv} is the convective flux defined as in (4.7). Notice that V_{FC}^{cnv} is not conservative across faces; conservation (3.6) will be imposed instead on the total flux (4.10). Similar to the pure diffusion case, an analogy between (4.8) and a discrete integration by parts allows to maintain the interpretation of the hybrid variable ϕ_F as the face-averaged scalar; however, in this case its exactness property for linear fields is lost. Formulation (4.9) is hereby named the *Mixed Centred* (MIXC) scheme: *mixed*, because it maintains the separation between d-forms for scalars and $(d-1)$-forms for vectors (hybridisation via static condensation is still possible as will be shown in Section 4.1.4, but the hybrid variable is not involved in the discrete convective flux definition); *centred* because, unlike upwinding techniques (Section 4.1.2), the convective flux definition does not depend on the direction of the convecting flow.

Alternatively, as suggested by e.g. [18, 70, 191], one may bypass the variational formulation argument and, in a spirit closer to classical FV, define directly a local flux reconstruction which includes both the convective and diffusive term, of the form

$$(\mathbf{V})_{\partial C} = \underbrace{\mathbb{M}_C^{-1}\left(\phi_C - \phi_F\right)_{\partial C}}_{\text{diffusion}} + \underbrace{\left(U_{FC}\phi_{FC}^{cnv}\right)_{\partial C}}_{\text{convection}} \ . \tag{4.11}$$

The goal is to define ϕ_{FC}^{cnv}, the convected scalar at the face, such that $U_{FC}\phi_{FC}^{cnv}$ results in a suitable approximation of $\int_F \phi\vec{U}\cdot\vec{n}_{FC}\,dS$. In classical FV the simplest choice would lead to a centred scheme where ϕ_{FC}^{cnv} comes from some form of face-averaging procedure based on cell-centred values. The MHFV framework has the advantage of already featuring a face-based degree of freedom naturally in the form of the hybrid variable, hence it can be set $\phi_{FC}^{cnv} = \phi_F$ which yields

$$(\mathbf{V})_{\partial C} = \mathbb{M}_C^{-1}\left(\phi_C - \phi_F\right)_{\partial C} + \left(U_{FC}\phi_F\right)_{\partial C} \ . \tag{4.12}$$

Formulation (4.12) is named the *Hybrid Centred* (HYBC) scheme. Contrary to MIXC, in HYBC the convective flux is conservative by definition, meaning that imposing flux conservation on the total flux (4.12) reverts to imposing conservation on its diffusive component.

Both centred strategies MIXC and HYBC lead to a second-order accurate scheme that is stable for low-Peclet problems (see results in Section 4.3.1). However, centred schemes in a MVE-like context suffer from the same stability issues as in FV and FE, namely they give rise to non-physical oscillations in the solution field which may become unbounded in strongly convection-dominated problems [18]. To support that statement, MIXC and HYBC are proven below to be convectively unstable via stability estimates.

Stability estimate for MIXC and HYBC. The following assumptions are made: a) the reaction term is not considered, i.e. $\eta = 0$, b) for simplicity, the homogeneous Dirichlet boundary value problem is considered, and c) $\nabla\cdot\vec{U} \geq 0$ (in particular this thesis considers incompressible applications where the convecting velocity field $\vec{U}$ is solenoidal, i.e. $\nabla\cdot\vec{U} = 0$). In the identity:

$$\sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\left(\phi_C - \phi_F\right) = \sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\phi_C - \underbrace{\sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\phi_F}_{=0} \tag{4.13}$$

the last term vanishes by virtue of flux conservation and the homogeneous Dirichlet boundary assumption. The continuity equation in (4.2) is discretised locally via the

divergence operator (3.13), i.e. $\sum_{F\in\partial C} V_{FC} = |C|\, f_C$. Injecting this into (4.13) yields

$$\sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\left(\phi_C - \phi_F\right) = \sum_{C\in\Omega_h} |C|\, f_C \phi_C \ . \tag{4.14}$$

In the specific case of the MIXC scheme (4.9) it also holds

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\left(\phi_C - \phi_F\right) &= \sum_{C\in\Omega_h} \left(\mathbb{M}_C^{-1}\left(\phi_C - \phi_F\right)_{\partial C}, \left(\phi_C - \phi_F\right)_{\partial C}\right) \\ &+ \sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C\left(\phi_C - \phi_F\right) \ .\end{aligned} \tag{4.15}$$

The last term in (4.15) can be further decomposed as

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C\left(\phi_C - \phi_F\right) &= \sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C^2 \\ &- \sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C\phi_F \\ &= \sum_{C\in\Omega_h}\left(\int_C \nabla\cdot\vec{U}\, dV\right)\phi_C^2 \\ &- \sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C\phi_F\end{aligned} \tag{4.16}$$

and the last term in (4.16) rewritten as

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C\phi_F &= \frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C^2 \\ &+ \underbrace{\frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_F^2}_{=0} \\ &- \frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\left(\phi_C - \phi_F\right)^2 \\ &= \frac{1}{2}\sum_{C\in\Omega_h}\left(\int_C \nabla\cdot\vec{U}\, dV\right)\phi_C^2 \\ &- \frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\left(\phi_C - \phi_F\right)^2\end{aligned} \tag{4.17}$$

where conservation of the convecting flux $\mathbf{U}$ was taken into account. The stability esti-

mate for MIXC is obtained by combining (4.17), (4.16), (4.15) and (4.14):

$$\begin{aligned}\sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C},\left(\phi_C-\phi_F\right)_{\partial C}\right) = \\ \underbrace{-\frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C}U_{FC}\left(\phi_C-\phi_F\right)^2}_{?}\underbrace{-\frac{1}{2}\sum_{C\in\Omega_h}\left(\int_C\nabla\cdot\vec{U}\,dV\right)\phi_C^2}_{\geq 0}+\sum_{C\in\Omega_h}|C|\,f_C\phi_C\ .\end{aligned} \tag{4.18}$$

The first term on the right-hand side of (4.18) is troublesome: since U_{FC} has no definite sign, this term can be either positive or negative. Therefore this estimate cannot provide a conclusive proof of the boundedness of the term on the left-hand side of (4.18), which is representative of the (squared) magnitude of the discrete gradient. For low-Peclet problems this is not an issue, since $\mathbb{M}_C^{-1}=\mathcal{O}\left(h\right)$ and $U_{FC}=\mathcal{O}\left(h^2\right)$ and thus the diffusive term ultimately dominates. Conversely, in convection-dominated problems the only way of maintaining the convective term small enough is by refining the mesh, and for particularly high Peclet numbers the tipping point is delayed to refinement levels that cannot be afforded in practice.

For the HYBC scheme (4.12) the estimate is very similar. Starting from

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C}V_{FC}\left(\phi_C-\phi_F\right) &= \sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C},\left(\phi_C-\phi_F\right)_{\partial C}\right) \\ &+ \sum_{C\in\Omega_h}\sum_{F\in\partial C}U_{FC}\phi_F\left(\phi_C-\phi_F\right)\ ,\end{aligned} \tag{4.19}$$

the last term in (4.19) is decomposed as

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C}U_{FC}\phi_F\left(\phi_C-\phi_F\right) &= \sum_{C\in\Omega_h}\sum_{F\in\partial C}U_{FC}\phi_C\phi_F \\ &- \underbrace{\sum_{C\in\Omega_h}\sum_{F\in\partial C}U_{FC}\phi_F^2}_{=0}\end{aligned} \tag{4.20}$$

and the remaining term on the right-hand side of (4.20) is rewritten as already shown in (4.17). Combining (4.17), (4.20), (4.19) and (4.14) yields

$$\begin{aligned}\sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C},\left(\phi_C-\phi_F\right)_{\partial C}\right) = \\ \underbrace{\frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C}U_{FC}\left(\phi_C-\phi_F\right)^2}_{?}\underbrace{-\frac{1}{2}\sum_{C\in\Omega_h}\left(\int_C\nabla\cdot\vec{U}\,dV\right)\phi_C^2}_{\geq 0}+\sum_{C\in\Omega_h}|C|\,f_C\phi_C\ ,\end{aligned} \tag{4.21}$$

i.e. an estimate nearly identical to that for the MIXC case (4.18), the sole difference

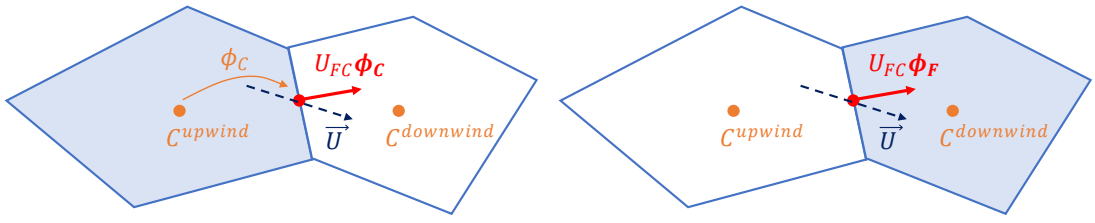
being that the problematic term on the right-hand side - whose sign is undefined - is added rather than subtracted. The conclusion is analogous, namely the boundedness of the discrete gradient is only guaranteed if the diffusive term dominates. □

4.1.2 First-order upwinding

In order to tackle the stability issues of centred schemes described above, one well-known solution in both classical FV [229] and FE [147] is *upwinding*: in (4.11), ϕ_{FC}^{cnv} is set to be the cell-centred variable of whichever of the two cells $C+$ and $C-$ is upwind of the other with respect to the convecting flow across F. The same can be done in the MHFV context, but in this case the choice is not limited to cell-averaged degrees of freedom only: one can take advantage of the existence of the hybrid variable and set

$$\phi_{FC}^{cnv} = \begin{cases} \phi_C & \text{if } U_{FC} \geq 0 \\ \phi_F & \text{if } U_{FC} < 0 \end{cases} . \tag{4.22}$$

Formulation (4.22), represented in Figure 4.1, is hereby named *Hybrid First-Order Upwind* (HUPW1), highlighting the fact that it reverts accuracy back to first-order via the phenomenon known as *numerical diffusion*. It has been observed however [18, 70] that using the hybrid variable as in (4.22) is likely to be less diffusive than traditional upwinding while maintaining stability (see estimate below). More importantly, the strategy allows to fully hybridise the scheme via static condensation as in the pure diffusion case, whereas a traditional cell-based upwind scheme would imply the presence of neighbouring cell values $\phi_{C'}$ in the local continuity equation (3.42), thus making it impossible to express ϕ_C in function of ϕ_F only. This property will be of primary importance when devising a unified framework for convective schemes (Section 4.1.4) and assembling the full hybrid convection-diffusion-reaction operator (Section 4.1.5).



(a) upwind cell (left) (b) downwind cell (right)

Figure 4.1: Hybrid first-order upwinding: convective flux across F as seen from an upwind and downwind cell.

A stability estimate for the HUPW1 scheme is provided below.

Stability estimate for HUPW1. Under the same assumptions as in Section 4.1.1, for the HUPW1 scheme (4.22) it holds

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\left(\phi_C-\phi_F\right) &= \sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C},\left(\phi_C-\phi_F\right)_{\partial C}\right)\\ &+ \sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}\geq 0} U_{FC}\phi_C\left(\phi_C-\phi_F\right)\\ &+ \sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}<0} U_{FC}\phi_F\left(\phi_C-\phi_F\right)\ .\end{aligned} \tag{4.23}$$

Breaking down each term similarly to what described in Section 4.1.1, and observing that conservation of $\mathbf{U}$ implies

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}\geq 0} U_{FC}\phi_F^2+\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}<0} U_{FC}\phi_F^2 &= \sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_F^2\\ &= 0\end{aligned} \tag{4.24}$$

and

$$\begin{aligned}\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}\geq 0} U_{FC}\phi_C^2+\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}<0} U_{FC}\phi_C^2 &= \sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}\phi_C^2\\ &= \sum_{C\in\Omega_h}\left(\int_C \nabla\cdot\vec{U}\,dV\right)\phi_C^2\ ,\end{aligned} \tag{4.25}$$

the following estimate is obtained:

$$\begin{aligned}&\sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C},\left(\phi_C-\phi_F\right)_{\partial C}\right) =\\ &\underbrace{\frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}<0} U_{FC}\left(\phi_C-\phi_F\right)^2}_{\leq 0}-\underbrace{\frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}\geq 0} U_{FC}\left(\phi_C-\phi_F\right)^2}_{\geq 0}\\ &\underbrace{-\frac{1}{2}\sum_{C\in\Omega_h}\left(\int_C \nabla\cdot\vec{U}\,dV\right)\phi_C^2}_{\geq 0}+\sum_{C\in\Omega_h}|C|\,f_C\phi_C\ .\end{aligned} \tag{4.26}$$

The estimate is evidently more favourable compared to the centred schemes: (4.26) shows HUPW1 to be stable since it holds

$$\sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C},\left(\phi_C-\phi_F\right)_{\partial C}\right)\leq\sum_{C\in\Omega_h}|C|\,f_C\phi_C\ , \tag{4.27}$$

as is the case in the pure diffusion scenario. □

4.1.3 θ-Scheme

The stability estimate for HUPW1 shows that the strategy leaves some room to reduce numerical dissipation. In classical FV one of the options is to take ϕ_{FC}^{cnv} as an intermediate value, weighted by a factor θ, between the face-averaged ϕ_F and its corresponding cell-averaged degree of freedom ϕ_C. Such a strategy is easily adapted to MHFV and improved upon by making use of the hybrid variable. The resulting scheme, hereby named *Hybrid θ-Scheme* (HTHE), sets:

$$\phi_{FC}^{cnv} = \begin{cases} \theta\phi_F + (1-\theta)\,\phi_C & \text{if } U_{FC} \geq 0 \\ \phi_F & \text{if } U_{FC} < 0 \end{cases} \qquad \text{with } 0 \leq \theta \leq 1 \ . \tag{4.28}$$

The θ-Scheme is an attempt at curbing the loss of accuracy caused by numerical diffusion while still benefiting from its stabilising effect. It is easy to verify how it is in fact a linear combination between HYBC and HUPW1, as it reverts to either scheme for $\theta = 1$ and $\theta = 0$, respectively. It is possible to derive a stability condition on the value of θ via the following estimate.

Stability estimate for HTHE. Under the same assumptions as in Section 4.1.1, the HTHE scheme (4.28) yields the identity

$$\begin{aligned} \sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\left(\phi_C - \phi_F\right) &= \sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C}, \left(\phi_C-\phi_F\right)_{\partial C}\right) \\ &+ \sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}\geq 0} U_{FC}\,(1-\theta)\,\phi_C\left(\phi_C-\phi_F\right) \\ &+ \sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}\geq 0} U_{FC}\theta\phi_F\left(\phi_C-\phi_F\right) \\ &+ \sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}<0} U_{FC}\phi_F\left(\phi_C-\phi_F\right) \ . \end{aligned} \tag{4.29}$$

Breaking down each term as usual, and considering again remarks (4.24) and (4.25), leads to the following expression:

$$\begin{aligned} \sum_{C\in\Omega_h}\left(\mathbb{M}_C^{-1}\left(\phi_C-\phi_F\right)_{\partial C}, \left(\phi_C-\phi_F\right)_{\partial C}\right) = \\ \underbrace{\frac{1}{2}\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}<0} U_{FC}\left(\phi_C-\phi_F\right)^2}_{\leq 0} - \frac{1-2\theta}{2}\underbrace{\sum_{C\in\Omega_h}\sum_{F\in\partial C/U_{FC}\geq 0} U_{FC}\left(\phi_C-\phi_F\right)^2}_{\geq 0} \\ \underbrace{-\frac{1}{2}\sum_{C\in\Omega_h}\left(\int_C \nabla\cdot\vec{U}\,dV\right)\phi_C^2}_{\geq 0} + \sum_{C\in\Omega_h}|C|\,f_C\phi_C \ . \end{aligned} \tag{4.30}$$

Hence, in order to the satisfy the stability condition (4.27), it is sufficient to impose

$$\frac{1-2\theta}{2} \geq 0 \quad \rightarrow \quad \theta \leq \frac{1}{2} \,, \tag{4.31}$$

meaning that HTHE is guaranteed to be stable for any value of θ below 0.5. A similar result in a MFV context is reported by Brezzi et al. [43]. □

4.1.4 Unified framework for convective schemes

Following the ideas put forward by Beirão da Veiga, Droniou and Manzini [18], it is useful to define a unified framework in MHFV in order to include all of the convection strategies from Sections 4.1.1, 4.1.2 and 4.1.3 by using a single, common notation system. Irrespective of the specific convection scheme, the local convective-diffusive flux operator can be written in the generic form

$$(\mathbf{V})_{\partial C} = \mathbb{N}_C \left(\phi_C - \phi_F\right)_{\partial C} + (\mathbf{U})_{\partial C}\, \phi_C \tag{4.32}$$

with

$$\mathbb{N}_C = \mathbb{M}_C^{-1} - \mathbb{U}_C \quad , \quad \mathbb{U}_C = \operatorname{diag}\left(U_{FC}^{dw}\right)_{\partial C} \,. \tag{4.33}$$

The specific definition of the "downwind flux" U_{FC}^{dw} differentiates each scheme:

$$U_{FC}^{dw} = \begin{cases} 0 & \text{MIXC} & \text{(Mixed Centred)} \\ U_{FC} & \text{HYBC} & \text{(Hybrid Centred)} \\ \min\left(0, U_{FC}\right) & \text{HUPW1} & \text{(Hybrid First-Order Upwind)} \\ \min\left(\theta U_{FC}, U_{FC}\right) & \text{HTHE} & \text{(Hybrid } \theta\text{-Scheme)} \end{cases} . \tag{4.34}$$

The unified framework is a way of emphasising similarities amongst different strategies and bridging gaps between approaches coming from different philosophies such as FE and FV. It also has the considerable practical advantage of facilitating the implementation of MHFV operators, since each strategy does not need to be treated separately at a high level. More importantly, it shows as anticipated that any of the schemes presented above can be hybridised thanks to the fact that, regardless of the chosen strategy, the local flux reconstruction (4.32) only hinges on face values ϕ_F and the cell-averaged ϕ_C over the cell being considered, hence ϕ_C can still be algebraically eliminated by expressing it as a function of $\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}$; the process shall be detailed in Section 4.1.5.

4.1.5 Hybrid convection-diffusion-reaction operator

The the generic unified formulation (4.32) for the convective-diffusive flux facilitates the assembly of the global convection-diffusion-reaction operator. Local continuity in (4.2) is discretised by applying the divergence operator (3.13) to the total flux $\mathbf{V}$:

$$\sum_{F\in\partial C} V_{FC} + |C|\,\eta_C\phi_C = |C|\,f_C \ . \tag{4.35}$$

Then the static condensation procedure [40] is applied as in the pure diffusion case (Section 3.3.2), namely by replacing (4.32) in (4.35) which allows to express ϕ_C as a function of the hybrid variable:

$$\phi_C = \frac{\left({\mathbb{N}_C}^T\mathbb{1}, \left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}\right) + |C|\,f_C}{(\mathbb{N}_C\mathbb{1},\mathbb{1}) + ((\mathbf{U})_{\partial C},\mathbb{1}) + |C|\,\eta_C} \tag{4.36}$$

and thus, re-injecting in (4.32), the local fluxes in function of $\widetilde{\boldsymbol{\phi}}$ are obtained:

$$(\mathbf{V})_{\partial C} = \frac{\left({\mathbb{N}_C}^T\mathbb{1}, \left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}\right) + |C|\,f_C}{(\mathbb{N}_C\mathbb{1},\mathbb{1}) + ((\mathbf{U})_{\partial C},\mathbb{1}) + |C|\,\eta_C}\left(\mathbb{N}_C\mathbb{1} + (\mathbf{U})_{\partial C}\right) - \mathbb{N}_C\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C} \ . \tag{4.37}$$

Notice that, when an upwinding convective strategy is deployed (HUPW1 or HTHE), the difference between traditional and hybrid upwinding becomes crucial: with a classical FV cell-based strategy, cell-averaged values $\phi_{C'}$ from some of the neighbouring cells C' would have appeared on the right-hand side of (4.36) (more specifically, those coming from cells upwind with respect to C), and it would have been impossible to eliminate them and express all fluxes in function of $\widetilde{\boldsymbol{\phi}}$ only, as in (4.37).

Imposing flux conservation (3.6) over each face yields the linear system

$$\mathcal{F}_{\mathbb{K},\vec{U},\eta}\widetilde{\boldsymbol{\phi}} = \widetilde{\mathbf{f}} \tag{4.38}$$

where $\mathcal{F}_{\mathbb{K},\vec{U},\eta}$ denotes the MHFV *hybrid convection-diffusion-reaction operator*. A remark: the hybrid right-hand side $\widetilde{\mathbf{f}}$ differs from the one for pure diffusion in (3.45), although the same notation is used for the sake of readability. For a convection-diffusion-reaction problem it is computed as

$$\begin{aligned}\left(\widetilde{\mathbf{f}}\right)_F \ = \ & -f_{C-}\,|C_-|\,\frac{\left(\mathbb{N}_{C-}\mathbb{1} + (\mathbf{U})_{\partial C-}\right)_F}{(\mathbb{N}_{C-}\mathbb{1},\mathbb{1}) + \left((\mathbf{U})_{\partial C-},\mathbb{1}\right) + |C_-|\,\eta_{C-}} \\ & -f_{C+}\,|C_+|\,\frac{\left(\mathbb{N}_{C+}\mathbb{1} + (\mathbf{U})_{\partial C+}\right)_F}{(\mathbb{N}_{C+}\mathbb{1},\mathbb{1}) + \left((\mathbf{U})_{\partial C+},\mathbb{1}\right) + |C_+|\,\eta_{C+}} \ .\end{aligned} \tag{4.39}$$

4.1.6 Boundary conditions

A typical scalar transport equation requires three basic types of boundary conditions: inlet, wall and outlet. The Dirichlet-type may be used to enforce the first two. Strong Dirichlet boundary conditions can be implemented for the convection-diffusion-reaction operator in the same, straightforward way as in the pure diffusion case (Section 3.3.5). The same applies to Neumann boundary conditions, where the imposed boundary value corresponds to the total diffusive + convective flux at each face.

For the outlet, however, a simple Neumann-type is not sufficient since the total flux value is usually not known. One of the most common solutions [229] is to define the domain geometry such that outlet planes are a) "distant enough" from any expected relevant physical phenomenon and b) oriented such that it is reasonable to not expect any changes in their normal direction $\vec{n}$, i.e. $\frac{\partial \phi}{\partial \vec{n}} = 0$. This is often referred to as "fully developed flow" condition, a typical basic example being the outlet of a flow through a pipe (Poiseuille flow).

In MHFV this translates to imposing zero normal gradient or, equivalently, zero diffusive flux across outlets. Denoting by $\partial\Omega_h^O$ the outlet boundary, this amounts to imposing:

$$\left(\mathbb{M}_C^{-1}\left(\phi_C - \phi_F\right)_{\partial C}\right)_F = 0 \quad \forall F \in \partial\Omega_h^O \ . \tag{4.40}$$

Hybridisation is then possible as usual by injecting (4.36) in (4.40), leading to the following equation for outlet faces:

$$\begin{aligned} &\left(\mathbb{M}_C^{-1}\mathbb{1}\frac{\left(\mathbb{N}_C^T\mathbb{1}, \left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}\right)}{\left(\mathbb{N}_C\mathbb{1}, \mathbb{1}\right) + \left(\left(\mathbf{U}\right)_{\partial C}, \mathbb{1}\right) + |C|\,\eta_C} - \mathbb{M}_C^{-1}\left(\widetilde{\boldsymbol{\phi}}\right)_{\partial C}\right)_F = \\ &-\left(\mathbb{M}_C^{-1}\mathbb{1}\frac{|C|\,f_C}{\left(\mathbb{N}_C\mathbb{1}, \mathbb{1}\right) + \left(\left(\mathbf{U}\right)_{\partial C}, \mathbb{1}\right) + |C|\,\eta_C}\right)_F \quad \forall F \in \partial\Omega_h^O \ . \end{aligned} \tag{4.41}$$

4.2 Stabilised second-order convection schemes

It was shown how hybrid upwinding (HUPW1) from Section 4.1.2 guarantees stability for any Peclet number at the cost of reverting accuracy to first-order for the scalar variable ϕ, and how the θ-Scheme (HTHE) from Section 4.1.3 maintains such stability for $\theta \leq 0.5$ while limiting to an extent the degrading effects of numerical diffusion.

One of the strengths of MVE-like methods in the pure diffusion case is the ability to construct operators up to an arbitrary order of accuracy (by adding degrees of

freedom within each element, see e.g. [19, 23, 150]). In particular, the MHFV diffusion scheme presented in Chapter 3 is conceived to be second-order accurate. It is desirable to preserve accuracy when convective terms are added, namely by deriving operators that are both stable in convection-dominated regimes and formally second-order accurate. So-called *Petrov-Galerkin* approaches [88], which aim at introducing artificial diffusion where necessary while minimising the loss of accuracy, are common stabilisation techniques in FE; examples of such methods include *Streamline-Upwind Petrov-Galerkin* (SUPG) [45, 126], *Pressure-Stabilizing/Petrov-Galerkin* (PSPG) [127] and *Galerkin/Least-Squares* (GLS) [128]. Concerning FV, a number of strategies have been devised: examples include traditional *flux limiting* [165, 166, 230, 231, 246] to enforce monotonicity of high-order flux reconstructions (e.g. MUSCL [227]), *(Weighted) Essentially Non-Oscillatory* (ENO/WENO) schemes [116, 154, 213], and the *Weighted Least-Squares* (WLSQR) approach [89, 90]. The development of order-preserving stabilisation techniques has received little attention from MFV and HMM scholars (a notable exception being the work of Piar et al. [191] on a MUSCL-like scheme) until very recently [3, 24, 129, 159]. In the following sections some solutions inspired by both the FE and FV frameworks are adapted to MHFV.

4.2.1 Streamline-Upwind Petrov-Galerkin

The *Streamline-Upwind Petrov-Galerkin* (SUPG) method comes from the FE community and it was first introduced by Brooks and Hughes [45, 126]. Fries and Matthies [88] interpret SUPG as the introduction of artificial diffusion in a "smart" way in order to benefit of its stabilising effects (the smoothing of unbounded non-physical oscillations) without sacrificing accuracy where this is not necessary: SUPG operates by adding to a centred scheme a certain amount of artificial diffusion in the streamline direction only (hence the name), thus ensuring that no diffusion perpendicular to the direction of the convecting flow (*crosswind*) is introduced, which is the reason for excessive numerical diffusion in first-order upwind schemes.

Based on this interpretation, SUPG implies increasing diffusivity anisotropically and thus can be implemented within any scheme capable of dealing with anisotropic diffusion. MHFV lends itself well to a SUPG implementation. More specifically, this corresponds to a HYBC formulation (4.12) where the cell-averaged diffusivity tensor $\mathbb{K}_C$ is augmented as follows:

$$\mathbb{K}_C^{SUPG} = \mathbb{K}_C + \tau_C \left(\vec{U}_C^{avg} \otimes \vec{U}_C^{avg} \right) \tag{4.42}$$

where $\vec{U}_C^{avg}$ is a cell-averaged convecting flow (this can be evaluated via e.g. the cell-average operator (3.24)) and τ_C the SUPG stabilisation parameter. The value of τ_C is

to be fine-tuned in order to obtain a suitable amount of numerical diffusivity, i.e. one that smoothens oscillations sufficiently but maintains the order of magnitude of $\mathbb{K}_C^{SUPG}$ within a reasonable range of the physical quantities characterising the overall problem. This is important especially for MHFV, since the value of $\mathbb{K}_C^{SUPG}$ affects that of the λ_{FC} weights in the stabilisation term (3.31) which, as pointed out in Section 3.3.4, should fall within reasonably well scaled values so that the desired spectral properties of the scalar product matrix $\mathbb{M}_C$ are not affected.

The specific formulation of τ_C, at least for generic meshes, is rather heuristic in nature [221]. Since the original work presented in this thesis, Manzini et al. [159] have proposed an adaptation of SUPG to MVE schemes which has been extended to an order-preserving version by Benedetto et al. [24]; both formulations rely on a traditional FE definition of τ_C. In the present work, the following MHFV-specific definition is proposed:

$$\tau_C = \frac{\sum_{F\in\partial C}\left(U_{FC}^{uw}\left\|\vec{x}_F - \vec{x}_C\right\|^2\right)}{|C|\left\|\vec{U}_C^{avg}\right\|^2} \tag{4.43}$$

where

$$U_{FC}^{uw} = \max\left(0, U_{FC}\right) \tag{4.44}$$

is the "upwind flux" (as opposed to the "downwind flux" in (4.34)). It is shown below how formulation (4.43) is intended to introduce an amount of streamline dissipation roughly equivalent to that caused by first-order upwinding.

Proof. Let E define the energy associated with the total flux, i.e.

$$E = \sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\phi_C \ . \tag{4.45}$$

Considering for ease of exposition the homogeneous Dirichlet boundary value problem, and taking into account flux conservation, (4.45) may be rewritten as

$$E = \sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}\left(\phi_C - \phi_F\right) \tag{4.46}$$

and the diffusive and convective contributions (E^{diff} and E^{cnv}) split as

$$\begin{aligned} E &= E^{diff} + E^{cnv} \\ &= \sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}^{diff}\left(\phi_C - \phi_F\right) + \sum_{C\in\Omega_h}\sum_{F\in\partial C} V_{FC}^{cnv}\left(\phi_C - \phi_F\right) \ . \end{aligned} \tag{4.47}$$

If V_{FC}^{cnv} is expressed according to the HUPW1 strategy (4.22), then it holds

$$\begin{aligned} E^{cnv} &= \sum_{C\in\Omega_h}\sum_{F\in\partial C}\left(U_{FC}^{uw}\phi_C + U_{FC}^{dw}\phi_F\right)(\phi_C-\phi_F) \\ &= \sum_{C\in\Omega_h}\sum_{F\in\partial C}\left(U_{FC}^{uw}(\phi_F+\phi_C-\phi_F) + U_{FC}^{dw}\phi_F\right)(\phi_C-\phi_F) \\ &= \sum_{C\in\Omega_h}\sum_{F\in\partial C}\left(U_{FC}\phi_F(\phi_C-\phi_F) + U_{FC}^{uw}(\phi_C-\phi_F)^2\right) \end{aligned} \tag{4.48}$$

where the trivial relationship $U_{FC}^{uw} + U_{FC}^{dw} = U_{FC}$ was used. As previously shown in (4.19), the term $\sum_{F\in\partial C} U_{FC}\phi_F(\phi_C-\phi_F)$ in (4.48) corresponds to the HYBC scheme (4.12) and is therefore non-dissipative. The remaining term represents the dissipation E^{diss} caused by upwinding:

$$E^{diss} = \sum_{C\in\Omega_h}\sum_{F\in\partial C} U_{FC}^{uw}(\phi_C-\phi_F)^2 \ . \tag{4.49}$$

Noticing that $(\phi_C-\phi_F)$ is first-order equivalent to $\nabla_C\phi\cdot(\vec{x}_F-\vec{x}_C)$ with $\nabla_C\phi$ some first-order approximation of the cell-averaged gradient of ϕ, (4.49) may be manipulated into the form

$$E^{diss} \approx \sum_{C\in\Omega_h} \mathbb{Q}_C\nabla_C\phi\cdot\nabla_C\phi \tag{4.50}$$

where $\mathbb{Q}_C$ is the $d\times d$ matrix:

$$(\mathbb{Q}_C)_{ij} = \sum_{F\in\partial C} U_{FC}^{uw}\left(x_F^i - x_C^i\right)\left(x_F^j - x_C^j\right) \ . \tag{4.51}$$

Expression (4.50) allows to see the added dissipation as a diffusion-like term, and it can therefore be taken as the term that the SUPG parameter τ_C should scale with in order to emulate upwind-like numerical diffusion. It makes sense to take the quantity

$$\text{trace}\left(\mathbb{Q}_C\right) = \sum_{F\in\partial C} U_{FC}^{uw}\left\|\vec{x}_F-\vec{x}_C\right\|^2 \tag{4.52}$$

as an appropriate indicator of the magnitude of the added diffusivity. Lastly, since SUPG augments diffusivity by a term of the form $\tau_C\left(\vec{U}_C^{avg}\otimes\vec{U}_C^{avg}\right)$, (4.52) is scaled by $|C|\left\|\vec{U}_C^{avg}\right\|^2$ in order to be dimensionally consistent with the diffusivity tensor $\mathbb{K}_C$ itself, thus yielding expression (4.43). □

4.2.2 Second-order upwinding

Classical FV also put forward a class of solutions to obtain a formally second-order accurate convection scheme that is stable even for convection-dominated problems: the well-known second-order upwind schemes. These are all based on the idea of reconstructing the value of ϕ at a face F via a piecewise-linear reconstruction in C, where C is the cell upwind of F. The same can be done in MHFV, once again with the further advantage of being able to use the hybrid variable for downwind values. This gives rise to the (unlimited) *Hybrid Second-Order Upwind* (HUPW2) scheme:

$$\phi_{FC}^{cnv} = \begin{cases} \phi_C + \nabla_C^{\mathcal{L},\lambda}\phi \cdot (\vec{x}_F - \vec{x}_C) & \text{if } U_{FC} \geq 0 \\ \phi_F & \text{if } U_{FC} < 0 \end{cases} . \tag{4.53}$$

Notice that, in order to attain a second-order accurate scheme, it is sufficient to reconstruct the face value of ϕ via any linearly consistent gradient approximation. In (4.53) the λ_{FC}-weighted LSQ gradient $\nabla_C^{\mathcal{L},\lambda}\phi$ is deliberately chosen for practical reasons: computation of the coefficients related to this specific gradient approximation is already required by the assembly of the local diffusive flux operator, as was shown in (3.48) and subsequent proof, while the possibility of modifying the LSQ weights allows to implement the stabilisation techniques discussed later in Sections 4.2.4 and 4.2.5. This choice allows to write a one-sided total flux expression where contributions from both diffusion and second-order upwinding are lumped together:

$$\begin{aligned} V_{FC} &= -\mathbb{K}_C \nabla_C^{\mathcal{G}}\phi \cdot \vec{F}_C - \frac{\phi_F - \phi_C}{\lambda_{FC}} + U_{FC}^{dw}\phi_F + U_{FC}^{uw}\phi_C \\ &+ \left(U_{FC}^{uw} + \frac{1}{\lambda_{FC}}\right) \nabla_C^{\mathcal{L},\lambda}\phi \cdot (\vec{x}_F - \vec{x}_C) \end{aligned} \tag{4.54}$$

where U_{FC}^{dw} is defined as in the HUPW1 case in (4.34). Based on (4.54), the full local flux operator takes the form

$$\begin{aligned} (\mathbf{V})_{\partial C} &= \mathbb{N}_C \left(\phi_C - \phi_F\right)_{\partial C} + (\mathbf{U})_{\partial C}\,\phi_C + \left(U_{FC}^{uw}\nabla_C^{\mathcal{L},\lambda}\phi \cdot (\vec{x}_F - \vec{x}_C)\right)_{\partial C} \\ &= \left(\mathbb{M}_C^{-1} - \mathbb{U}_C\right)\left(\phi_C - \phi_F\right)_{\partial C} + (\mathbf{U})_{\partial C}\,\phi_C + \left(U_{FC}^{uw}\nabla_C^{\mathcal{L},\lambda}\phi \cdot (\vec{x}_F - \vec{x}_C)\right)_{\partial C} \end{aligned} \tag{4.55}$$

where $\mathbb{U}_C$ is assembled according to the same rules as for HUPW1 in (4.33)-(4.34); hence, remembering that construction of $\mathbb{M}_C^{-1}$ requires inverting the $d \times d$ LSQ matrix, using the same LSQ gradient for the second-order upwind scheme implies that the first and last term in (4.55) can be assembled at the same time, and no further matrix inversions are required.

Classical FV literature [229] reports that second-order upwinding suffers from the

same instability issues affecting centred schemes: it can give rise to instabilities (spurious oscillations in the solution field) if no special care is taken to bound the gradient, especially for high-Peclet problems and/or in the presence of steep gradients/near-discontinuities in the scalar solution field. In MHFV, the HUPW2 scheme is not exempt from the very same issues. A series of gradient control/moderation techniques is investigated in the following sections.

4.2.3 Flux limiters

Arguably the most popular strategy in FV is the use of so-called *flux limiters*, where - for a second-order scheme - the reconstructed face value is expressed as

$$\phi_{FC}^{cnv} = \phi_C + \underbrace{\gamma_C}_{\text{limiter}} \nabla_C^{\mathcal{L},\lambda}\phi \cdot (\vec{x}_F - \vec{x}_C) \tag{4.56}$$

with γ_C being a limiting coefficient computed in order to adjust the predicted face value such that the reconstruction does not give rise to, or at least bounds, new local extrema. More specifically, those limiting procedures that aim at suppressing oscillations altogether, thus producing a solution field that strictly preserves monotonicity, are known as *Total Variation Diminishing* (TVD) [229]. Flux limiting on unstructured meshes was first introduced by Barth and Jespersen [11], whose strategy (described below) is found to suffer from two drawbacks: a) excessive numerical diffusion and subsequent degradation of accuracy in regions around smoothened extrema [213], and b) difficulties - due to the non-differentiability of the formulation - in converging the iterative solution algorithm to steady-state [230]. Since then, several authors have developed variants of limiters to both reduce diffusivity (e.g. [13, 187]) and improve convergence behaviour (e.g. [230, 231] and, more recently, [31, 165, 166]).

Considering the motivating factors for the present work (Section 2.4), these drawbacks strongly discourage the usage of limiters in MHFV. However, given that adaptation to MHFV is fairly trivial, some well-known limiters are implemented for the sake of comparison. The first formulation considered is the *Barth-Jespersen* [11] strategy, which belongs to the TVD family [166]. In classical FV the scheme limits the gradient slope such that, for each cell, the reconstructed face value at each face does not exceed the cell-averaged one in the respective adjacent cell (Figure 4.2). The MHFV framework allows to implement two versions of this: *Cell-Based Barth-Jespersen* (BJC), which limits by the value of ϕ_C (as in the original version); *Face-Based Barth-Jespersen* (BJF), which uses the hybrid variable ϕ_F instead. The limiter is computed for each cell as follows:

1. using current solution values, find $\delta\phi_C^{min}$ and $\delta\phi_C^{max}$, respectively the largest negative and positive values of either: $(\phi_{C'} - \phi_C)_{\partial C}$ with C' being neighbour cells of C (BJC), or $(\phi_F - \phi_C)_{\partial C}$ (BJF);

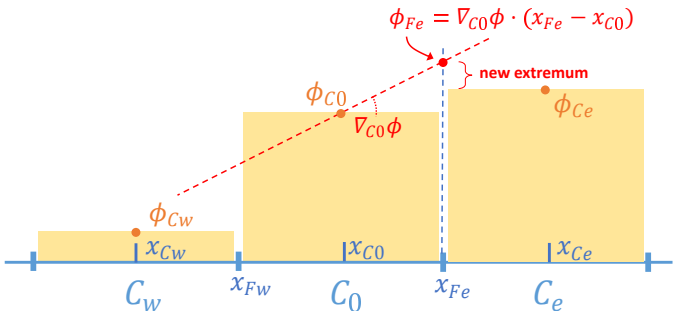
2. using the λ_{FC}-weighted LSQ gradient, compute the unlimited reconstructed value at each face of C:

$$\phi_F^{unlm} = \phi_C + \nabla_C^{\mathcal{L},\lambda}\phi \cdot (\vec{x}_F - \vec{x}_C) \quad \forall F \in \partial C \tag{4.57}$$

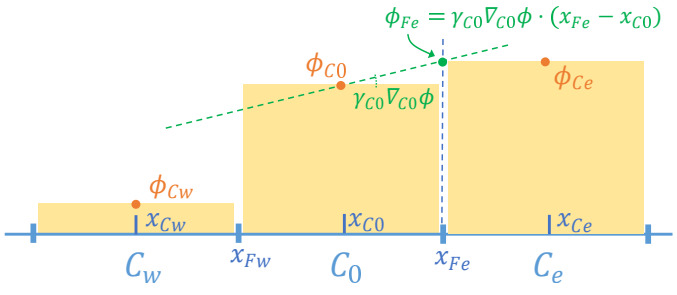
3. compute the maximum allowable γ_F for each face:

$$\gamma_F = \begin{cases} \min\left(1, \dfrac{\delta\phi_C^{max}}{\phi_F^{unlm} - \phi_C}\right) & \text{if} \phi_F^{unlm} - \phi_C > 0 \\ \min\left(1, \dfrac{\delta\phi_C^{min}}{\phi_F^{unlm} - \phi_C}\right) & \text{if} \phi_F^{unlm} - \phi_C < 0 \\ 1 & \text{if} \phi_F^{unlm} - \phi_C \approx 0 \end{cases} \quad \forall F \in \partial C \ ; \tag{4.58}$$

4. set $\gamma_C = \min(\gamma_F)_{\partial C}$.



(a) unlimited



(b) limited

Figure 4.2: Barth-Jespersen limiter (cell-based) illustrated on a 1D domain: reconstructed value ϕ_{Fe} at face F_e from cell C_0, without and with limiting.

As mentioned above, it has been observed [166] that the non-differentiability of the Barth-Jespersen limiter can hamper convergence to the steady-state solution. One of the earliest improvements in that sense was presented by Venkatakrishnan [230], who proposed replacing in step 3 the non-differentiable function $\min(1, y)$ with

$$f(y) = \frac{y^2 + 2y}{y^2 + y + 2} \ . \tag{4.59}$$

This is the second formulation implemented in this work. Again, in MHFV two versions of the limiter are implemented: *Cell-Based Venkatakrishnan* (VNKC) corresponding to the original FV formulation, and *Face-Based Venkatakrishnan* (VNKF) relying on the hybrid variable. The Venkatakrishnan limiter does resolve the issue of non-differentiability, thus facilitating convergence at the cost of limiting the gradient even in regions where no new extrema are formed.

All limiters presented above (BJC,BJF,VNKC,VNKF) highlight a further disadvantage of the limiting approach: they cause the MHFV scheme to lose its fully implicit nature. The value of γ_C, required to assemble coefficients of the local convective flux operator, depends on the solution field itself ($\boldsymbol{\phi}$ as well as $\widetilde{\boldsymbol{\phi}}$ for the face-based versions); the overall scheme thus becomes solution-dependent. This is bothersome not only from a philosophical standpoint - the continuous convection operator is linear and thus, in the mimetic spirit, its discrete counterpart should be too - but also in practice, since it requires an iterative solution process unlike all other schemes presented so far, where operator linearity was preserved. The last remark however becomes less of an issue when deriving the Navier-Stokes operator (Chapter 5), since the non-linearity of the problem necessarily calls for an iterative process.

4.2.4 Weighted Least-Squares

An alternative to limiters is the family of so-called *Essentially Non-Oscillatory* (ENO) schemes, first published by Harten, Engquist, Osher and Chakravarthy [116] in a Finite Differencing framework. When extended to classical FV, ENO methods consist in selecting for each cell an "ad hoc" stencil for gradient approximation such that the resulting gradient is biased towards preserving local smoothness, i.e. a stencil that does not cross shocks or near-discontinuities in the solution field. It has been observed [213] how ENO schemes do not totally suppress oscillations, but they succeed in having them decay as $\mathcal{O}(h^2)$ (for second-order schemes), hence the name: *essentially* non-oscillatory. Variants of ENO schemes have been developed for unstructured grids [2, 176].

One of the disadvantages of ENO schemes is that implementation is not trivial, since

they operate by selecting for each cell a range of candidate stencils, and then evaluating some local smoothness parameter for each stencil in order to select the most suitable one. The *Weighted Essentially Non-Oscillatory* schemes (WENO), first presented by Liu, Osher and Chan [154], constitute an improvement in that sense: WENO schemes reconstruct gradients over all candidate stencils, and then perform a weighted convex combination of these, with weights favouring those stencils lying on smoother areas. Several WENO variants are found in the literature [87, 125].

A similar concept gives rise to the *Weighted Least-Squares* (WLSQR) approach proposed by Fürst [89, 90] and implemented in the present work for MHFV. The scheme is applicable when the reconstruction of the scalar face value is done via a LSQ gradient, which is the case for the HUPW2 strategy (4.53). WLSQR suggests constructing each local LSQ gradient with weights ω_{FC} designed to strongly favour those components that would constitute the corresponding ENO sub-stencil. The method is thus similar to WENO in its essence, but easier to implement for generic unstructured meshes since local stencils need not be modified.

Adapting the formulation by Fürst [89] to the MHFV face-based LSQ gradient yields weights of the form

$$\omega_{FC} = \sqrt{\frac{h^{-r}}{\left|\frac{\phi_F - \phi_C}{h}\right|^p + h^q}} \quad \text{where } h = \|\vec{x}_F - \vec{x}_C\| \ , \tag{4.60}$$

with p, q and r being constants to be determined empirically; Fürst [90] suggests $p = 4$, $q = -2$ and $r = 3$ for 2D problems, although these might not necessarily be the best choices here since the MHFV LSQ gradient uses face-to-cell centre distances, as opposed to the traditional FV cell-to-cell.

A weight defined as in (4.60) is intended to be biased towards areas where the solution is smooth, i.e. where $|\phi_F - \phi_C|$ is small. A way of including this bias in the MHFV gradient $\nabla_C^{\mathcal{L},\lambda}$ is to augment the already existing LSQ weights λ_{FC} (defined in Section 3.3.4) by ω_{FC}, thus obtaining a modified set of weights λ_{FC}^{WLSQR} to be used throughout the assembly of the operator. Two precautions must be taken: a) the LSQ gradient (3.50) is in fact weighted by the inverse of λ_{FC}, hence λ_{FC}^{-1} is the quantity that should be augmented; b) as already highlighted for the SUPG approach (Section 4.2.1), care must be taken not to alter the dimensionality of λ_{FC}, lest incur the risk of altering the dimensions of the stabilisation term (3.31) and jeopardise stability of the MHFV diffusion

scheme. An unscaled augmentation would be of the form

$$\begin{aligned} \frac{1}{\lambda_{FC}^{WLSQR}} &= \frac{1}{\lambda_{FC}} + \omega_{FC} \\ &= \frac{1 + \lambda_{FC}\omega_{FC}}{\lambda_{FC}} \end{aligned} \quad .$$

Rescaling this by a factor $(1 + \lambda_{FC}\omega_{FC})_{\max} = 1 + \max(\lambda_{FC}\omega_{FC})_{\partial C}$ leads to the following expression for the augmented weights:

$$\lambda_{FC}^{WLSQR} = \frac{\lambda_{FC}\,(1 + \lambda_{FC}\omega_{FC})_{\max}}{1 + \lambda_{FC}\omega_{FC}} \quad . \tag{4.61}$$

It can be verified how definition (4.61) produces augmented weights that are close to the original λ_{FC} in areas where the local solution field is smooth, and biased towards the smoother side of cells lying across steep gradients/near-discontinuities. For example, over an isotropic Cartesian grid the parameter h in (4.60) and the λ_{FC} weights are identical for all faces of C, hence $\max(\lambda_{FC}\omega_{FC})_{\partial C}$ is given by the face with the smallest $|\phi_F - \phi_C|$. The augmented weight λ_{FC}^{WLSQR} (4.61) for this face will be identical to the unmodified λ_{FC}. For all other faces, λ_{FC}^{WLSQR} will be larger - and thus $\left(\lambda_{FC}^{WLSQR}\right)^{-1}$ penalised - for those faces with larger jumps $|\phi_F - \phi_C|$ in the solution field. If the cell lies in a smooth area, then the solution jumps for all of its faces are roughly equivalent which implies $\lambda_{FC}^{WLSQR} \approx \lambda_{FC} \quad \forall F \in \partial C$, i.e. the weighting scheme reverts to the unmodified version from Section 3.3.4.

There are two evident disadvantages connected with the WLSQR approach. The first is its semi-empirical nature: parameters p, q and r in (4.60) are to be determined based on experimental data, and their fine-tuning is found to be case-dependent [89]; furthermore, there are at present no clear guidelines on how to extend them to 3D cases [90]. The second is that, as in the case of flux limiters (Section 4.2.3), WLSQR is also a solution-dependent approach subject to the same drawbacks, namely a) the violation at the discrete level of the linearity of the continuous convection operator, and b) the need for iterative solution techniques.

4.2.5 Upwind Least-Squares

A stabilisation technique native to the MHFV scheme developed in this thesis is now introduced: the *Upwind Least-Squares* (ULSQR) method. ULSQR combines the basic principles of both SUPG and WLSQR: similarly to SUPG it aims at a streamline stabilisation depending on the local convecting flow but, instead of altering the diffusivity tensor $\mathbb{K}_C$, it deploys second-order upwinding through the HUPW2 scheme (4.53) where,

in the WLSQR spirit, the LSQ weights are modified in a way that takes into account the direction of the convecting flux.

More specifically, the λ_{FC} weights in (4.54) - and therefore in the LSQ gradient (3.50) - are replaced with a set of modified weights defined as

$$\lambda_{FC}^{ULSQR} = \frac{\lambda_{FC}}{1 + \lambda_{FC} \left| U_{FC}^{dw} \right|} \tag{4.62}$$

which is obtained via the augmentation: $\left(\lambda_{FC}^{ULSQR}\right)^{-1} = \lambda_{FC}^{-1} + \left| U_{FC}^{dw} \right|$. The ULSQR modification only affects weights related to faces which are upwind with respect to C, since in the opposite case it holds $U_{FC}^{dw} = 0$. Similarly, faces that are perpendicular (or almost) to the crosswind direction are not affected, since in that case $U_{FC} \approx 0$. Hence ULSQR yields a LSQ gradient that is both streamline-biased and upwind-biased. In that sense, ULSQR shares some of its premises with the QUICK scheme (*Quadratic Upstream Interpolation for Convective Kinematics*, [145]), which is upwind-biased in the choice of values for interpolation.

It is interesting to observe that, where active (i.e. for upwind faces), weight augmentation (4.62) introduces a bias proportional to the dimensionless quantity

$$Pe_{FC} = \lambda_{FC} \left| U_{FC} \right| \tag{4.63}$$

which can be naturally interpreted as a *local Peclet number*. Definition (4.63) is justified by considering the weight definitions from Section 3.3.4. Taking for example the ORTN formulation (3.83) with isotropic diffusivity k_C, it leads to

$$Pe_{FC} = \frac{\left\| \vec{x}_F - \vec{x}_C \right\| \left| U_{FC} \right|}{k_C \left| F \right|} \tag{4.64}$$

which is indeed non-dimensional and grows larger as the problem becomes locally convection-dominated, leading to a stronger upwind bias. It is argued that, compared to WLSQR (Section 4.2.4), ULSQR may be just as heuristic but it fits more elegantly within the MHFV scheme, namely because a) since Pe_{FC} is dimensionless, expression (4.62) does not affect the dimension of the weights, meaning that no further weight normalisation is required, b) ULSQR is less empirical as it is solely based on quantities directly related to the physics of the problem (conversely, as mentioned, WLSQR relies on empirical fine-tuning of parameters based on case-dependent numerical results), and c) ULSQR, unlike WLSQR and flux limiters, is not solution-dependent, thus allowing to maintain a linear and fully implicit MHFV scheme.

4.3 Validation of MHFV for convection-diffusion-reaction problems

4.3.1 Low-Pe h-convergence for basic convective schemes

A manufactured solution test case analogous to the one used for pure diffusion (Section 3.4.1) is proposed here in order to validate the centred convective schemes (MIXC and HYBC) form Section 4.1.1 and first-order upwinding (HUPW1) from Section 4.1.2. The domain is the unit square $\Omega =]0,1[\times]0,1[$ and the anisotropic diffusion tensor is defined as in (3.93). The divergence-free convecting field $\vec{U}$ is defined as

$$\vec{U}(x,y) = \begin{pmatrix} \beta x(2y-1)(x-1) \\ -\beta y(2x-1)(y-1) \end{pmatrix}, \tag{4.65}$$

where setting parameter $\beta = 10^3$ leads to a rather low macroscopic Peclet number $Pe \approx 57$ (based on averaged velocity and diffusivity fields and the side of the square domain), hence maintaining the problem within a range where centred convection schemes can be expected to be stable even on coarser meshes. The reaction coefficient is set to

$$\eta(x,y) = 10(x+y) \tag{4.66}$$

and the source term computed such that the exact solution remains the same as in the pure diffusion test case (3.94). Dirichlet boundary conditions are applied throughout.

Tests are run on two mesh sequences: one polygonal distorted (Figure 3.6), one fully Cartesian. Errors in L^2 norm for each MHFV variable for the MIXC, HYBC and HUPW1 strategies are plotted in Figure 4.3, 4.4 and 4.5 respectively. Corresponding values and convergence rates are reported in Table 4-A through 4-F.

Table 4-A: Mixed Centred (MIXC) - polygonal distorted mesh: errors and convergence rates.

h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
1.751 E^{-1}	9.882 E^{-1}	–	3.324 E^{+0}	–	1.38 E^{+0}	–
8.967 E^{-2}	5.572 E^{-2}	4.297	1.330 E^{-1}	4.809	7.94 E^{-2}	4.261
4.449 E^{-2}	1.342 E^{-2}	2.031	3.193 E^{-2}	2.036	2.04 E^{-2}	1.938
2.206 E^{-2}	3.323 E^{-3}	1.990	7.926 E^{-3}	1.986	5.38 E^{-3}	1.900
1.097 E^{-2}	8.285 E^{-4}	1.988	1.979 E^{-3}	1.986	1.49 E^{-3}	1.839
5.471 E^{-3}	2.070 E^{-4}	1.994	4.946 E^{-4}	1.993	4.39 E^{-4}	1.758

Table 4-B: Hybrid Centred (HYBC) - polygonal distorted mesh: errors and convergence rates.

h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
1.751 E^{-1}	1.546 E^{-1}	–	8.678 E^{-2}	–	1.450 E^{-1}	–
8.967 E^{-2}	2.917 E^{-2}	2.492	2.315 E^{-2}	1.975	3.344 E^{-2}	2.192
4.449 E^{-2}	7.072 E^{-3}	2.022	6.230 E^{-3}	1.873	8.164 E^{-3}	2.012
2.206 E^{-2}	1.765 E^{-3}	1.979	1.609 E^{-3}	1.930	2.021 E^{-3}	1.990
1.097 E^{-2}	4.416 E^{-4}	1.983	4.079 E^{-4}	1.964	5.083 E^{-4}	1.976
5.471 E^{-3}	1.105 E^{-4}	1.991	1.027 E^{-4}	1.982	1.302 E^{-4}	1.958

Table 4-C: Hybrid First-Order Upwind (HUPW1) - polygonal distorted mesh: errors and convergence rates.

h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
1.751 E^{-1}	2.481 E^{-1}	–	1.629 E^{-1}	–	2.70 E^{-1}	–
8.967 E^{-2}	1.189 E^{-1}	1.099	9.096 E^{-2}	0.871	1.30 E^{-1}	1.096
4.449 E^{-2}	5.808 E^{-2}	1.022	4.833 E^{-2}	0.902	6.19 E^{-2}	1.054
2.206 E^{-2}	2.871 E^{-2}	1.004	2.515 E^{-2}	0.931	3.03 E^{-2}	1.020
1.097 E^{-2}	1.429 E^{-2}	0.999	1.289 E^{-2}	0.957	1.50 E^{-2}	1.003
5.471 E^{-3}	7.122 E^{-3}	1.001	6.528 E^{-3}	0.978	7.48 E^{-3}	1.002

Table 4-D: Mixed Centred (MIXC) - Cartesian mesh: errors and convergence rates.

h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
1.111 E^{-1}	1.238 E^{-1}	–	5.113 E^{-1}	–	8.134 E^{-2}	–
5.263 E^{-2}	2.312 E^{-2}	2.246	1.080 E^{-1}	2.081	1.909 E^{-2}	1.940
2.564 E^{-2}	5.253 E^{-3}	2.061	2.577 E^{-2}	1.993	4.642 E^{-3}	1.966
1.266 E^{-2}	1.268 E^{-3}	2.014	6.324 E^{-3}	1.991	1.144 E^{-3}	1.985
6.289 E^{-3}	3.122 E^{-4}	2.003	1.568 E^{-3}	1.993	2.846 E^{-4}	1.988
3.135 E^{-3}	7.751 E^{-5}	2.001	3.903 E^{-4}	1.998	7.111 E^{-5}	1.992

Table 4-E: Hybrid Centred (HYBC) - Cartesian mesh: errors and convergence rates.

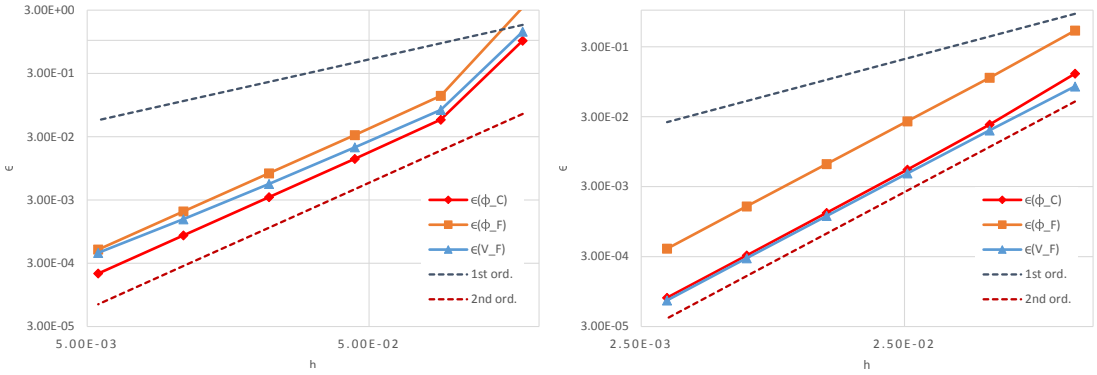
h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
$1.111\ \mathrm{E}^{-1}$	$1.274\ \mathrm{E}^{-1}$	–	$5.752\ \mathrm{E}^{-1}$	–	$7.884\ \mathrm{E}^{-2}$	–
$5.263\ \mathrm{E}^{-2}$	$2.383\ \mathrm{E}^{-2}$	2.244	$1.271\ \mathrm{E}^{-1}$	1.994	$1.883\ \mathrm{E}^{-2}$	1.917
$2.564\ \mathrm{E}^{-2}$	$5.432\ \mathrm{E}^{-3}$	2.056	$3.012\ \mathrm{E}^{-2}$	1.973	$4.533\ \mathrm{E}^{-3}$	1.980
$1.266\ \mathrm{E}^{-2}$	$1.312\ \mathrm{E}^{-3}$	2.013	$7.348\ \mathrm{E}^{-3}$	1.986	$1.114\ \mathrm{E}^{-3}$	1.989
$6.289\ \mathrm{E}^{-3}$	$3.232\ \mathrm{E}^{-4}$	2.002	$1.817\ \mathrm{E}^{-3}$	1.992	$2.770\ \mathrm{E}^{-4}$	1.989
$3.135\ \mathrm{E}^{-3}$	$8.027\ \mathrm{E}^{-5}$	2.001	$4.514\ \mathrm{E}^{-4}$	1.997	$6.922\ \mathrm{E}^{-5}$	1.992

Table 4-F: Hybrid First-Order Upwind (HUPW1) - Cartesian mesh: errors and convergence rates.

h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_F)$	Rate	$\epsilon(V_F)$	Rate
$1.111\ \mathrm{E}^{-1}$	$2.639\ \mathrm{E}^{-1}$	–	$2.149\ \mathrm{E}^{-1}$	–	$1.735\ \mathrm{E}^{-1}$	–
$5.263\ \mathrm{E}^{-2}$	$1.290\ \mathrm{E}^{-1}$	0.958	$1.171\ \mathrm{E}^{-1}$	0.813	$8.816\ \mathrm{E}^{-2}$	0.906
$2.564\ \mathrm{E}^{-2}$	$6.176\ \mathrm{E}^{-2}$	1.024	$5.849\ \mathrm{E}^{-2}$	0.965	$4.277\ \mathrm{E}^{-2}$	1.006
$1.266\ \mathrm{E}^{-2}$	$2.964\ \mathrm{E}^{-2}$	1.040	$2.873\ \mathrm{E}^{-2}$	1.007	$2.046\ \mathrm{E}^{-2}$	1.045
$6.289\ \mathrm{E}^{-3}$	$1.440\ \mathrm{E}^{-2}$	1.032	$1.416\ \mathrm{E}^{-2}$	1.011	$9.907\ \mathrm{E}^{-3}$	1.037
$3.135\ \mathrm{E}^{-3}$	$7.081\ \mathrm{E}^{-3}$	1.020	$7.016\ \mathrm{E}^{-3}$	1.009	$4.859\ \mathrm{E}^{-3}$	1.023

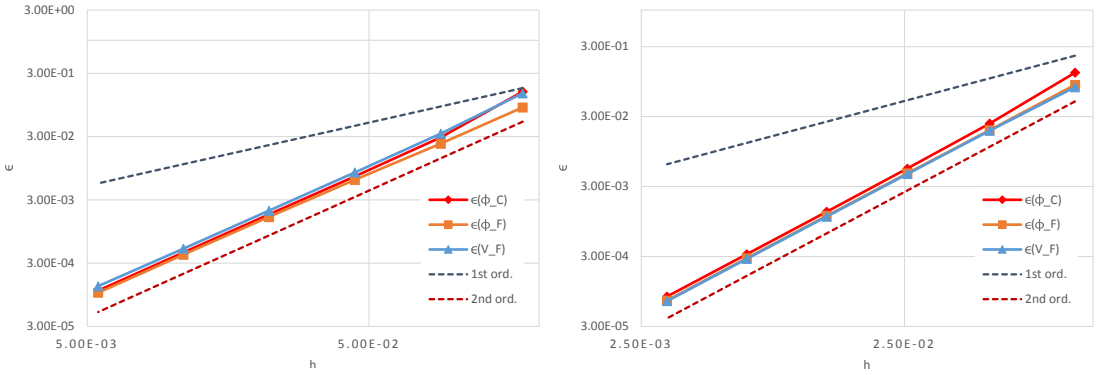
Results are overall in line with what expected: second-order accuracy is observed on the scalar variable for both centred schemes MIXC and HYBC on both mesh types, whereas HUPW1 reverts to first-order accuracy. Similar observations apply to the flux variable, which also exhibits second-order (or near) accuracy with centred schemes.

It is also observed that MIXC (Figure 4.3) produces a slightly higher error on the hybrid variable. This can be explained by the fact that, as mentioned before, while hybrid schemes impose strong consistency on convective fluxes, in the MIXC case these fluxes are weakly consistent and they do not involve ϕ_F in their definition; the hybrid variable therefore loses its exactness property as face-averaged scalar for linear solution fields. This however does not degrade the scheme itself, since in the MHFV philosophy the hybrid variable may be thought of as a "working" variable, used to solve the system and then reconstruct the solution fields: cell-averaged scalars $\boldsymbol{\phi}$ and total face fluxes $\mathbf{V}$.



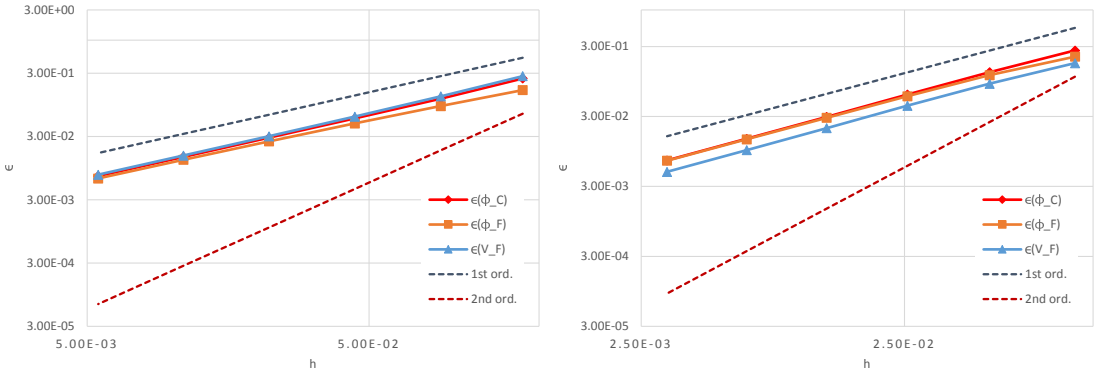
(a) polygonal distorted mesh (b) Cartesian mesh

Figure 4.3: Mixed Centred (MIXC): h-convergence.



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 4.4: Hybrid Centred (HYBC): h-convergence.



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 4.5: Hybrid First-Order Upwind (HUPW1): h-convergence.

4.3.2 Low-Pe validation of the Hybrid θ-Scheme

It is also interesting to analyse numerical results for the Hybrid θ-Scheme (HTHE) from Section 4.1.3. This is done on the same test case as in the previous section, which is run on a polygonal distorted mesh for different values of θ. Convergence is plotted in Figure 4.6 - on the L^2 norm of the error for ϕ_C only, for the sake of readability. Error values and convergence rates are also reported in Table 4-G.

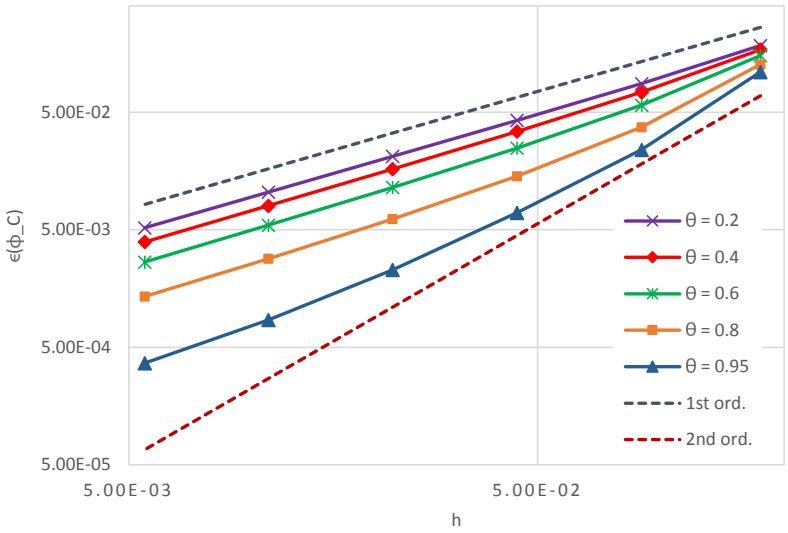


Figure 4.6: Hybrid θ-Scheme (HTHE): h-convergence on a polygonal distorted mesh for different values of θ.

Table 4-G: Hybrid θ-Scheme (HTHE) - polygonal distorted mesh: errors and convergence rates for at different values of θ.

	$\theta = 0.2$		$\theta = 0.6$		$\theta = 0.95$	
h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate
1.751 E^{-1}	1.841 E^{-1}	–	1.517 E^{-1}	–	1.098 E^{-1}	–
8.967 E^{-2}	8.745 E^{-2}	1.112	5.737 E^{-2}	1.453	2.393 E^{-2}	2.277
4.449 E^{-2}	4.262 E^{-2}	1.025	2.477 E^{-2}	1.198	6.973 E^{-3}	1.759
2.206 E^{-2}	2.103 E^{-2}	1.007	1.144 E^{-2}	1.101	2.274 E^{-3}	1.597
1.097 E^{-2}	1.043 E^{-2}	1.004	5.458 E^{-3}	1.059	8.551 E^{-4}	1.400
5.471 E^{-3}	5.187 E^{-3}	1.004	2.655 E^{-3}	1.036	3.672 E^{-4}	1.215

As expected, increasing the value of θ leads to an increase in the order of accuracy from first to second-order: this is because, as explained in Section 4.1.3, HTHE is a linear combination between hybrid HUPW1 ($\theta = 0$) and HYBC ($\theta = 1$), with the goal of curbing artificial diffusion introduced by the former without incurring in stability issues caused by the latter.

Numerical results show that the benefits are especially evident on coarser meshes, whereas on finer ones the gain on accuracy remains rather modest even for high values of θ ($\theta = 0.95$). A possible improvement could be done by implementing the FV *Scharfetter-Gummel scheme* (FVSG): originally developed for classical FV [208], FVSG can be interpreted as a θ-Scheme where the value of θ is locally adjusted according to a local Peclet number, thus avoiding numerical diffusion where it is not necessary - such as e.g. in areas where the mesh is sufficiently refined. An adaptation of FVSG to a HMM scheme has been investigated by Beirão da Veiga et al. [18].

4.3.3 High-Pe h-convergence for Hybrid First and Second-Order Upwinding

Table 4-H: Behaviour of MHFV centred and upwind convective schemes on a coarse mesh for an increasingly convection-dominated problem.

		$\epsilon(\phi_C)$			
k	Pe	MIXC	HYBC	HUPW1	HUPW2
$1\ \mathrm{E}^{-1}$	$8.53\ \mathrm{E}^{+1}$	$3.924\ \mathrm{E}^{-1}$	$8.877\ \mathrm{E}^{-2}$	$1.699\ \mathrm{E}^{-1}$	$5.819\ \mathrm{E}^{-2}$
$1\ \mathrm{E}^{-2}$	$8.53\ \mathrm{E}^{+2}$	$4.109\ \mathrm{E}^{+0}$	$3.772\ \mathrm{E}^{-1}$	$2.064\ \mathrm{E}^{-1}$	$5.920\ \mathrm{E}^{-2}$
$1\ \mathrm{E}^{-3}$	$8.53\ \mathrm{E}^{+3}$	$3.377\ \mathrm{E}^{+1}$	$5.033\ \mathrm{E}^{+0}$	$2.114\ \mathrm{E}^{-1}$	$5.724\ \mathrm{E}^{-2}$
$1\ \mathrm{E}^{-4}$	$8.53\ \mathrm{E}^{+2}$	$3.068\ \mathrm{E}^{+4}$	$6.002\ \mathrm{E}^{+1}$	$2.120\ \mathrm{E}^{-1}$	$5.685\ \mathrm{E}^{-2}$
$1\ \mathrm{E}^{-5}$	$8.53\ \mathrm{E}^{+5}$	$3.046\ \mathrm{E}^{+3}$	$6.114\ \mathrm{E}^{+2}$	$2.120\ \mathrm{E}^{-1}$	$5.681\ \mathrm{E}^{-2}$

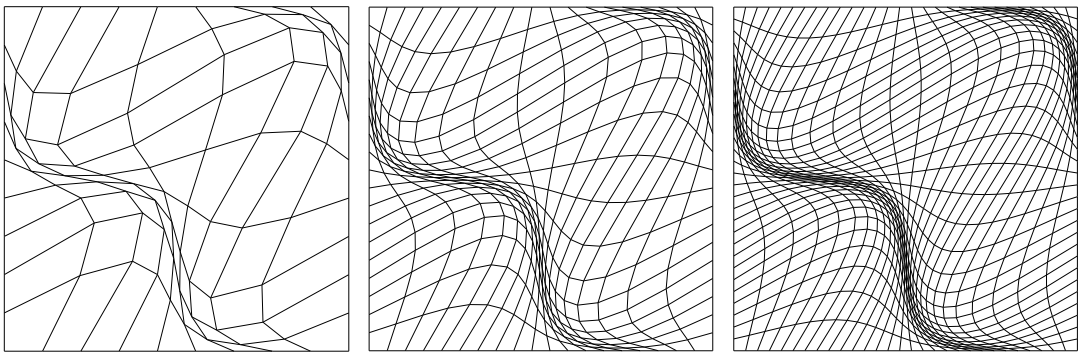
A test case is set up to validate the Hybrid First and Second-Order Upwinding schemes (HUPW1 and HUPW2) in a convection-dominated regime. The problem considered is a pure convection-diffusion one (hence without reaction: $\eta = 0$) over the unit square domain $\Omega =]0,1[\times]0,1[$. Diffusivity k is chosen to be constant and isotropic, which is achieved in MHFV by using a diagonal matrix for the diffusivity tensor, i.e. $\mathbb{K} = k\mathbb{I}$. The solenoidal convecting field is defined as

$$\vec{U}(x,y) = \begin{pmatrix} 10x + 2 \\ 3x - 10y \end{pmatrix} . \tag{4.67}$$

The source term is computed such that the exact solution is

$$\phi_{ex}(x,y) = 2x^2 + \cos\left(2\pi x y^2\right) . \tag{4.68}$$

Firstly, the need for upwinding schemes is illustrated by solving over a rather coarse mesh ($h \approx 0.26$) for an increasing macroscopic Peclet number, which is achieved by gradually decreasing the diffusivity k. The effects are shown in Table 4-H, where the L^2 norm of



(a) $h \approx 0.1$ (b) $h \approx 0.05$ (c) $h \approx 0.03$

Figure 4.7: Refinement sequence for a 2D quadrilateral distorted mesh.

the error on ϕ_C is reported for centred and upwind schemes. The error steadily increases with Pe for both centred schemes and becomes unbounded. Conversely, when upwinding is employed, the error remains bounded below a certain value even at high Pe numbers. The decreased order of accuracy due to first-order upwinding leads to a larger error bound for HUPW1 than for HUPW2. For this specific case, results show that HUPW2 is stable even on coarser meshes and does not require any further limiting/stabilising procedures.

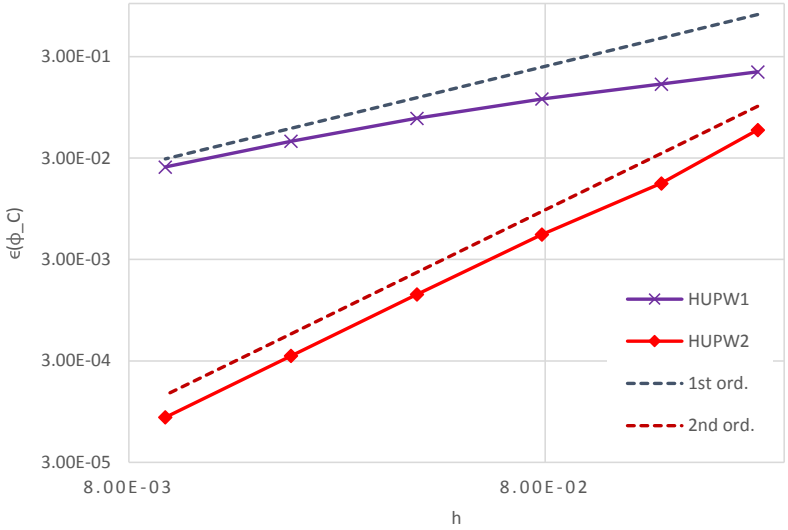


Figure 4.8: Hybrid First and (unlimited) Second-Order Upwinding (HUPW1 and HUPW2): h-convergence on a quadrilateral distorted mesh for a convection-dominated problem.

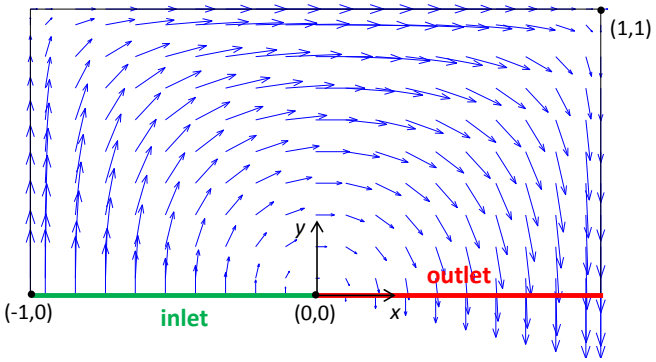
Secondly, a h-convergence study is conducted for the HUPW1 and HUPW2 schemes. The lowest value $k = 10^{-5}$ from Table 4-H is selected, corresponding to the rather high $Pe \approx 8.53\ \mathrm{E}^5$. The mesh is of type quadrilateral distorted (Figure 4.7). Results are reported in Figure 4.8 and Table 4-I (on ϕ_C only for better readability).

Table 4-I: Hybrid First and Second-Order Upwind (HUPW1 and HUPW2) - quadrilateral distorted mesh: errors and convergence rates.

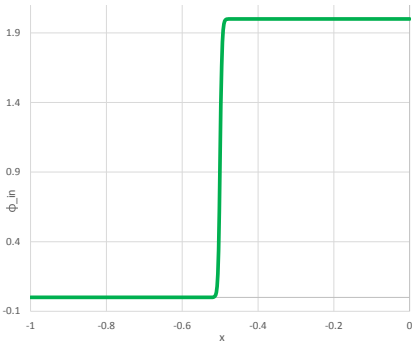
	HUPW1		HUPW2	
h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate
2.593 E^{-1}	2.120 E^{-1}	–	5.681 E^{-2}	–
1.521 E^{-1}	1.607 E^{-1}	0.519	1.688 E^{-2}	2.275
7.856 E^{-2}	1.146 E^{-1}	0.512	5.292 E^{-3}	1.756
3.935 E^{-2}	7.404 E^{-2}	0.632	1.358 E^{-3}	1.967
1.962 E^{-2}	4.394 E^{-2}	0.750	3.366 E^{-4}	2.004
9.789 E^{-3}	2.444 E^{-2}	0.844	8.358 E^{-5}	2.004

Both schemes behave as expected and, especially for finer meshes in the sequence, they converge according to their nominal order of accuracy (HUPW1 shows a pre-asymptotic behaviour on coarser girds). Results are deemed sufficient to validate the accuracy of MHFV hybrid upwinding techniques in high-Pe conditions.

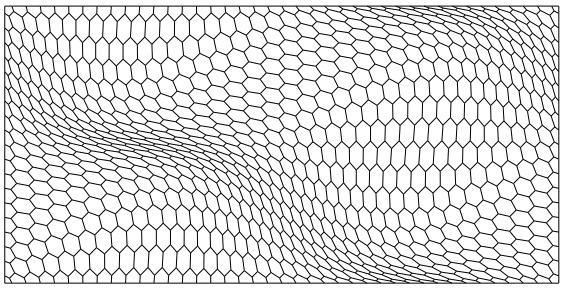
4.3.4 Comparison of stabilisation techniques



(a) domain, convecting field, boundary conditions



(b) inlet profile ϕ_{in}



(c) mesh type

Figure 4.9: Smith-Hutton test case setup.

In this section the Smith-Hutton test case [215] is used to test the various stabilisation strategies from Section 4.2 and analyse how they compare against each other. The test case is a 2D convection-diffusion problem specifically designed for this kind of analysis. The setup is shown in Figure 4.9. The domain is the rectangle $\Omega =]-1,1[\times]0,1[$, and the divergence-free convecting field is

$$\vec{U}(x,y) = \begin{pmatrix} 2y\left(1-x^2\right) \\ -2x\left(1-y^2\right) \end{pmatrix} \tag{4.69}$$

which corresponds to the stream function

$$\psi = -\left(1-x^2\right)\left(1-y^2\right) \ ; \tag{4.70}$$

the pattern of this flow field is shown in Figure 4.9(a). On the bottom side of Ω an inlet distribution is imposed over the interval $-1 \leq x \leq 0$:

$$\phi_{in}(x) = 1 + \tanh\left(\beta\left(1+2x\right)\right) \tag{4.71}$$

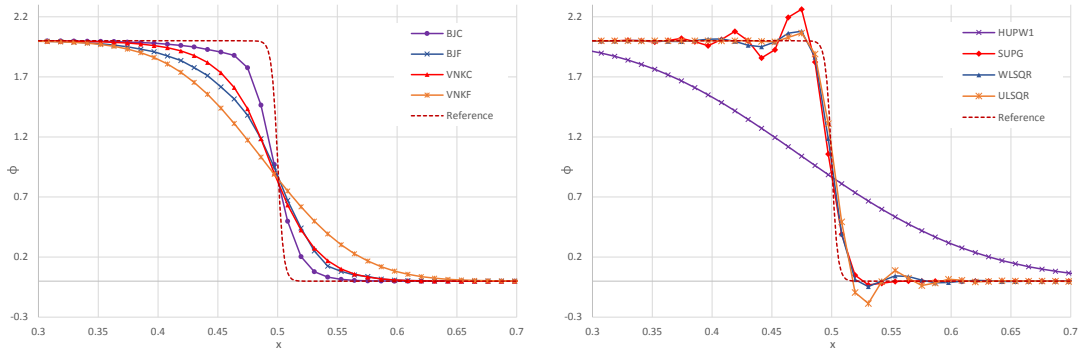
which, as shown in Figure 4.9(b), corresponds to a transition from $\phi_{in}(-1) = 0$ to $\phi_{in}(0) = 2$, with parameter β determining how sharp the transition is. A high value is selected ($\beta = 100$) in order to test MHFV stabilisation strategies in the presence of a shock in the distribution of ϕ. In the case of purely convective flow, ϕ is constant along streamlines, hence the exact solution is given by

$$\phi_{ex}(\psi) = 1 + \tanh\left(\beta\left(1-2\sqrt{1+\psi}\right)\right) = \phi_{ex}(x,y) \ . \tag{4.72}$$

Expression (4.72) is used to enforce Dirichlet values on the left, right and top sides of Ω, where it is close to zero. Finally, an outlet boundary condition is imposed on the bottom side over the interval $0 \leq x \leq 1$.

A convection-dominated regime is achieved by setting a low (constant and isotropic) diffusivity $k = 10^{-6}$, which gives a macroscopic Peclet number $Pe \approx 10^6$. As mentioned above, in the purely convective case ϕ is constant along streamlines and its outlet profile is an exact mirror image of the inlet distribution. Since the test case is run at a high Pe a similar result can be expected, hence the outlet profile given by (4.72) is used as a reference solution.

The discrete problem is solved over a rather coarse ($h \approx 1.96\ \mathrm{E}^{-2}$) polygonal distorted mesh (Figure 4.9(c)). For solution-dependent schemes, fixed-point iterations are performed down to a normalised residual of 10^{-3}.



(a) flux limiters (b) other stabilisers

Figure 4.10: Smith-Hutton outlet profiles of ϕ_F for different stabilisation schemes.

The outlet profile of the hybrid variable ϕ_F obtained via each scheme is plotted in Figure 4.10. As expected, HUPW1 (Figure 4.10(b)) produces a non-oscillatory monotone profile, but causes an excessive increase in diffusivity noticeable in the heavily smoothed near-discontinuity. All limiting techniques on the second-order scheme (Figure 4.10(a)) also exhibit a TVD behaviour while being closer to the reference solution, although a considerable amount of numerical diffusion is still visible. More specifically, the Barth-Jespersen limiters (BJC and BJF) produce less diffusive results compared to Venkatakrishnan (VNKC and VNKF) which is expected since, as Michalak and Ollivier-Gooch observe [166], such strategies introduce a further level of smoothening. It can also be observed that, at least for this test case, both limiter types perform considerably better in their traditional cell-based version compared to the hybrid face-based.

WLSQR and ULSQR produce similar profiles: slight overshoots are present on either side of the sharp transition, but they can reasonably be expected to be bounded regardless of Pe or mesh parameters. In particular, for WLSQR stability is theoretically proven by Fürst [89] for certain types of discontinuous fields; a similarly rigorous mathematical analysis of ULSQR is yet to be conducted. Based on the amplitude of oscillations, in particular on the lower side, WLSQR appears to be slightly more favourable; this result however cannot be considered as general enough, especially considering that the exponents in weight formulation (4.60) may need a case-dependent tuning. Furthermore, as mentioned in Section 4.2.5, ULSQR possesses the attractive feature of not being solution-dependent, meaning that it only requires one linear solve and no iterative processes.

Concerning SUPG, oscillations appear to be completely dumped on the lower side of the step but there is a significant overshoot at the top. Further tests have revealed that, on more regular meshes such as Cartesian ones, the overshoot is not as pronounced.

This might indicate that the MHFV choice of SUPG stabilization parameter (4.43), while limiting oscillations, is not entirely mesh-independent.

The full Smith-Hutton solution fields for all strategies are also reported in Figure 4.11

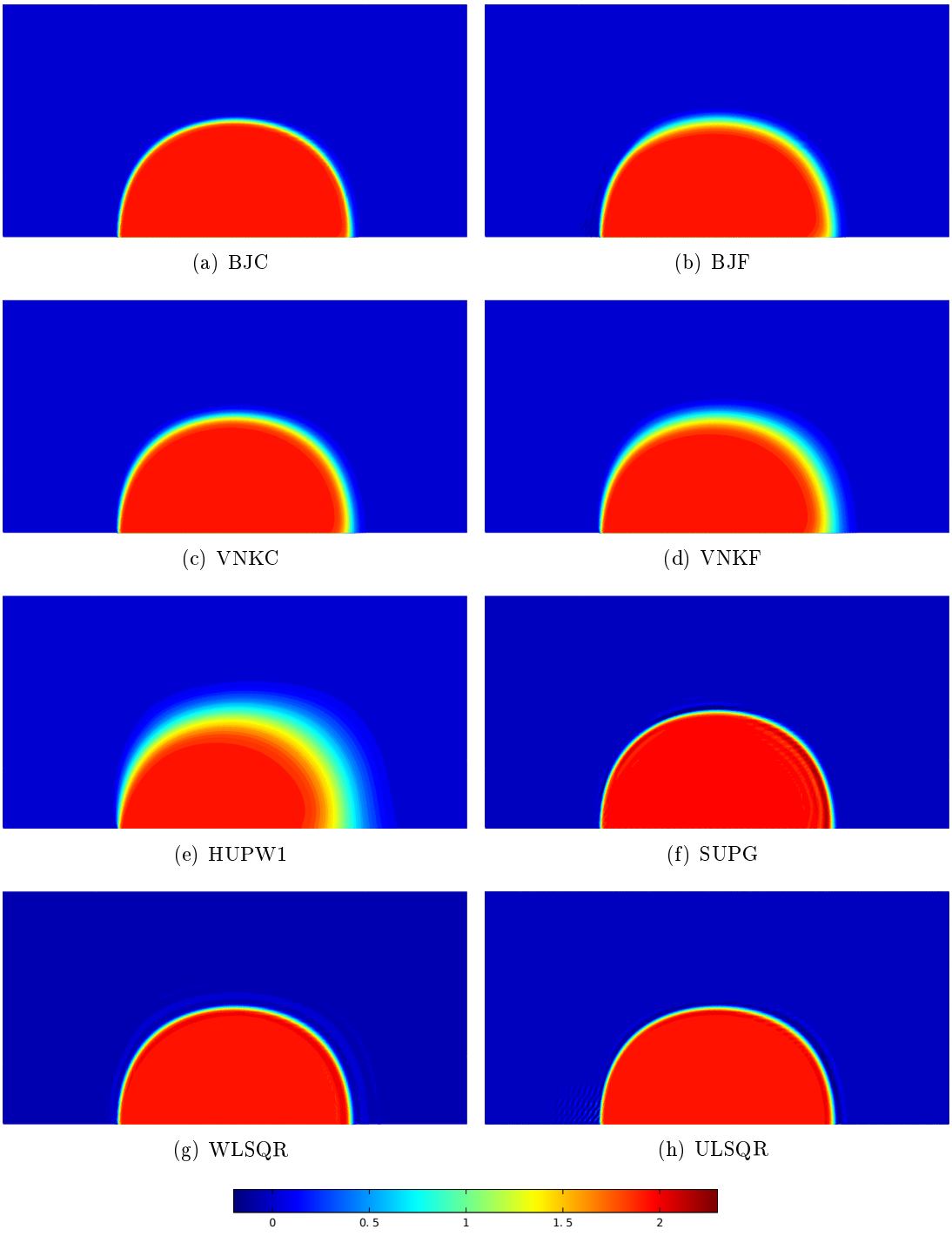
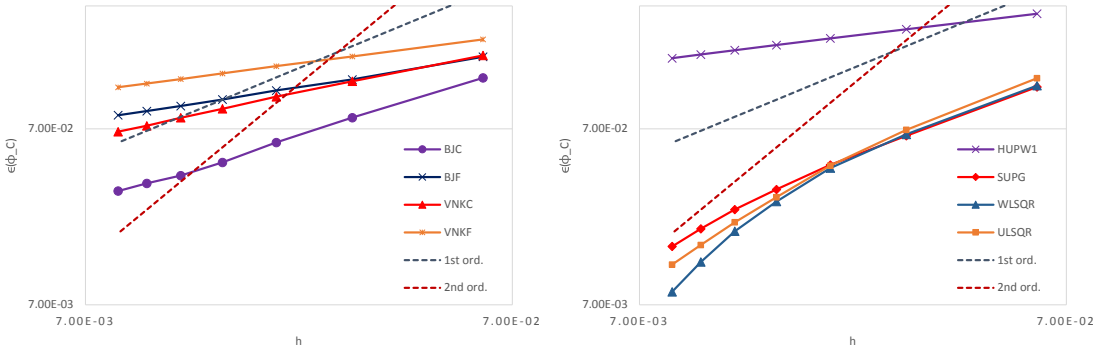


Figure 4.11: Smith-Hutton solution field ϕ for different stabilisation schemes.

for a further analysis. The contours confirm what already observed on outlet profiles. Numerical diffusion is evident for HUPW1 and face-based flux limiters BJF and VNKF, and less so for the cell-based versions BJC and VNKC. For SUPG, WLSQR and ULSQR, oscillations can be seen propagating in both directions from the sharp transition, but in all cases they remain confined within a relatively restricted area surrounding the shock, while the solution remains uniform and smooth elsewhere in zero-gradient areas. There is no definite qualitative superiority of one scheme over the others.



(a) flux limiters (b) other stabilisers

Figure 4.12: Limited/stabilised schemes: h-convergence on the Smith-Hutton test case.

Lastly a h-convergence analysis is performed for all second-order stabilised strategies - along with first-order upwinding for comparison. Accuracy is assessed in terms of L^2 norm of the error on the cell-averaged $\boldsymbol{\phi}$ measured against the pure convective exact solution (4.72). Results are reported in Figure 4.12 and Tables 4-J and 4-K.

It is noticeable how HUPW1 severely underperforms in this case, exhibiting a convergence rate even lower than its nominal first-order value. All flux limiting procedures also degrade the order of accuracy of the method, pushing it back to first-order or below, although the error is consistently smaller than with HUPW1. Barth-Jespersen displays a slightly better h-convergence behaviour compared to Venkatakrishnan and, for both limiters, cell-based versions appear to be superior to face-based - as already observed through the outlet profiles analysis.

On the other hand SUPG, WLSQR and ULSQR all perform comparably: one can see a pre-asymptotic behaviour in the sequence, which steepens as the mesh is refined until it matches (or almost) a second-order slope in the last entries. Again, on this test case a slight superiority is displayed by WLSQR in terms of convergence slope.

Table 4-J: Smith-Hutton test case: errors and convergence rates for flux limiters.

	BJC		BJF		VNKC		VNKF	
h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate
$5.98\ \mathrm{E}^{-2}$	$1.37\ \mathrm{E}^{-1}$	–	$1.80\ \mathrm{E}^{-1}$	–	$1.83\ \mathrm{E}^{-1}$	–	$2.26\ \mathrm{E}^{-1}$	–
$2.95\ \mathrm{E}^{-2}$	$8.11\ \mathrm{E}^{-2}$	0.74	$1.34\ \mathrm{E}^{-1}$	0.42	$1.31\ \mathrm{E}^{-1}$	0.48	$1.81\ \mathrm{E}^{-1}$	0.32
$1.96\ \mathrm{E}^{-2}$	$5.86\ \mathrm{E}^{-2}$	0.79	$1.16\ \mathrm{E}^{-1}$	0.35	$1.07\ \mathrm{E}^{-1}$	0.49	$1.59\ \mathrm{E}^{-1}$	0.31
$1.47\ \mathrm{E}^{-2}$	$4.51\ \mathrm{E}^{-2}$	0.90	$1.03\ \mathrm{E}^{-1}$	0.40	$9.11\ \mathrm{E}^{-2}$	0.55	$1.45\ \mathrm{E}^{-1}$	0.33
$1.17\ \mathrm{E}^{-2}$	$3.80\ \mathrm{E}^{-2}$	0.76	$9.45\ \mathrm{E}^{-2}$	0.38	$8.10\ \mathrm{E}^{-2}$	0.52	$1.34\ \mathrm{E}^{-1}$	0.33
$9.75\ \mathrm{E}^{-3}$	$3.43\ \mathrm{E}^{-2}$	0.56	$8.84\ \mathrm{E}^{-2}$	0.37	$7.30\ \mathrm{E}^{-2}$	0.57	$1.27\ \mathrm{E}^{-1}$	0.32
$8.35\ \mathrm{E}^{-3}$	$3.11\ \mathrm{E}^{-2}$	0.65	$8.37\ \mathrm{E}^{-2}$	0.35	$6.75\ \mathrm{E}^{-2}$	0.50	$1.21\ \mathrm{E}^{-1}$	0.31

Table 4-K: Smith-Hutton test case: errors and convergence rates for stabilisation strategies.

	HUPW1		SUPG		WLSQR		ULSQR	
h	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate	$\epsilon(\phi_C)$	Rate
$5.98\ \mathrm{E}^{-2}$	$3.16\ \mathrm{E}^{-1}$	–	$1.21\ \mathrm{E}^{-1}$	–	$1.23\ \mathrm{E}^{-1}$	–	$1.36\ \mathrm{E}^{-1}$	–
$2.95\ \mathrm{E}^{-2}$	$2.58\ \mathrm{E}^{-1}$	0.29	$6.39\ \mathrm{E}^{-2}$	0.91	$6.52\ \mathrm{E}^{-2}$	0.90	$6.91\ \mathrm{E}^{-2}$	0.96
$1.96\ \mathrm{E}^{-2}$	$2.29\ \mathrm{E}^{-1}$	0.29	$4.37\ \mathrm{E}^{-2}$	0.93	$4.19\ \mathrm{E}^{-2}$	1.08	$4.33\ \mathrm{E}^{-2}$	1.14
$1.47\ \mathrm{E}^{-2}$	$2.10\ \mathrm{E}^{-1}$	0.30	$3.17\ \mathrm{E}^{-2}$	1.10	$2.71\ \mathrm{E}^{-2}$	1.51	$2.86\ \mathrm{E}^{-2}$	1.42
$1.17\ \mathrm{E}^{-2}$	$1.96\ \mathrm{E}^{-1}$	0.30	$2.44\ \mathrm{E}^{-2}$	1.17	$1.84\ \mathrm{E}^{-2}$	1.73	$2.06\ \mathrm{E}^{-2}$	1.46
$9.75\ \mathrm{E}^{-3}$	$1.85\ \mathrm{E}^{-1}$	0.31	$1.90\ \mathrm{E}^{-2}$	1.37	$1.23\ \mathrm{E}^{-2}$	2.20	$1.53\ \mathrm{E}^{-2}$	1.62
$8.35\ \mathrm{E}^{-3}$	$1.77\ \mathrm{E}^{-1}$	0.31	$1.50\ \mathrm{E}^{-2}$	1.49	$8.30\ \mathrm{E}^{-3}$	2.52	$1.19\ \mathrm{E}^{-2}$	1.66

Chapter 5

MHFV Incompressible Navier-Stokes

5.1 The Navier-Stokes scheme

The MHFV hybrid convection-diffusion-reaction (4.38) and divergence (3.13) operators provide all tools necessary for deriving a MHFV scheme for the *incompressible steady-state Navier-Stokes problem*:

$$\begin{cases} \left(\vec{U}\cdot\nabla\right)\vec{U} - \nu\nabla^2\vec{U} + \nabla p = \vec{g} \\ \nabla\cdot\vec{U} = 0 \end{cases} \quad \text{in } \Omega \tag{5.1}$$

where ∇^2 is the vector Laplacian, ν is the kinematic viscosity (considered isotropic in this chapter), $\vec{g}$ is the momentum source term (e.g. gravity) and $\vec{U}$ and p are the unknown velocity and pressure fields respectively (or, more correctly, p is the pressure divided by a constant density ρ).

The literature on the numerical approximation of (5.1) is vast: as Droniou and Eymard [72] observe, mathematicians tend to focus on FE approaches [102, 111] while CFD engineers and physicists prefer FV schemes - traditionally presented first in a simplified form over staggered, structured grids [188, 229] and then adapted to general meshes [83, 206]. Such a preference is attributed to the more straightforward physical interpretation of FV discrete operators, as well as to their ease of implementation. Another considerable advantage of FV for incompressible applications is that the divergence-free condition is imposed to the discrete velocity field such that mass conservation within each control volume is ensured. Conversely, in FE methods mass conservation is in general

satisfied only in a weak sense [161]. A common solution is to penalise the momentum equation by a quantity proportional to the mass imbalance (*grad-div* stabilisation [178]), which is however parameter-dependent. Alternatively, MFE schemes have been proposed based on specific mixed element pairs with the approximation of the velocity being one order higher than the approximation of the pressure, such that the divergence operator maps into the discrete pressure space and the resulting velocity field is divergence-free point-wise (e.g. [46, 149, 247]). It has also been observed [149] how some of these element pairs satisfy the *inf-sup* or *Ladyshenskaya-Brezzi-Babuška* (LBB) stability condition [40]. Discontinuous Galerkin methods (see e.g. [57, 103, 170]) are also mentioned as a separate class of schemes which enforce local mass conservation and are applicable to general meshes.

Addressing the MVE methodology to the Navier-Stokes problem (5.1) could lead to the derivation of operators similar to FV, with a familiar physical interpretation (i.e. the requirement that conservation laws be satisfied at a discrete level over each control volume) but at the same time capable of supporting general polyhedral meshes with fewer restrictions on admissibility compared to classical FV. From the MVE community, the first divergence-free scheme for incompressible Navier-Stokes has been presented very recently by Beirão da Veiga et al. [22], who demonstrate the superiority of MVE over some standard MFE schemes in terms of h-convergence when solving over distorted grids and for a wide range of Reynolds numbers. The method proposed in this chapter attempts to achieve similar results while remaining closer to classical FV, namely by building upon the first-order accurate MFV scheme presented by Droniou and Eymard [72].

5.1.1 Discrete variables and preliminary notation

The degrees of freedom required by MHFV Navier-Stokes are defined in this section. These are the d cell-averaged velocity components $\mathbf{u}^i$ in Q^h:

$$\left(\mathbf{u}^i\right)_C = u^i_C = \frac{1}{|C|}\int_C U^i \, dV \quad \forall C \in \Omega_h \ , \quad i = 1 \cdots d \ ; \tag{5.2}$$

the d face-based velocity components $\widetilde{\mathbf{u}}^i$ (hybrid variables):

$$\left(\widetilde{\mathbf{u}}^i\right)_F = u^i_F = \frac{1}{|F|}\int_F U^i \, dS \quad \forall F \in \Omega_h \ , \quad i = 1 \cdots d \ ; \tag{5.3}$$

the cell-averaged pressure $\mathbf{p}$ in Q^h:

$$(\mathbf{p})_C = p_C = \frac{1}{|C|}\int_C p \, dV \quad \forall C \in \Omega_h \ ; \tag{5.4}$$

the convecting flux $\mathbf{U}$ in X^h:

$$(\mathbf{U})_F = U_F = U_{FC+} = \int_F \vec{U} \cdot \vec{n}_{FC+} \, dS \quad \forall F \in \Omega_h \ . \tag{5.5}$$

In terms of represented physical quantity, there is no difference between the convecting flux and the velocity components since both are a discrete representation of the unknown velocity field $\vec{U}$, although in different spaces. In MHFV, however, $\mathbf{U}$ is intentionally presented as a separate variable. The reason behind this is mainly philosophical: in the MVE/MFV spirit, it is desirable to discretise $\vec{U}$ in X^h when it is considered as a vector field, and in Q^h when each velocity component is treated separately as a scalar value. There is also a practical advantage in separating velocity-related degrees of freedom this way, which will become particularly relevant when assembling the discrete adjoint Navier-Stokes system (Chapter 6). The concept is not completely foreign to classical FV either. An example is the well-known *Rhie-Chow interpolation* [203]: introduced in order to prevent velocity-pressure decoupling - and subsequent checker-board modes - for collocated FV schemes, Rhie-Chow is de facto a face-based definition of discrete convecting fluxes. It is however seldom presented in this light, and it tends to be associated with the solution algorithm rather than the discretisation scheme itself, hence the concept is often overlooked by traditional CFD literature.

The MHFV framework, and in particular the interpretation of hybrid variables as face-based quantities, naturally leads to making the following choice when it comes to establishing the relationship between $\mathbf{U}$ and the $\widetilde{\mathbf{u}}^i$ components: the convecting flux across each face is set to be the scalar product between the hybrid velocity vector and the face vector, i.e.

$$U_{FC} = \sum_{i=1}^{d} u_F^i F_C^i \ . \tag{5.6}$$

Since $\mathbf{U}$ belongs to X^h, all properties of MVE fluxes listed in Section 3.2.1 hold, in particular flux conservation

$$U_{FC+} + U_{FC-} = 0 \tag{5.7}$$

and subsequently the existence of an "unsigned" convecting flux U_F such that

$$(\mathbf{U})_F = U_F = U_{FC+} \ , \tag{5.8}$$

which is compatible with definition (5.6). Finally, (5.6) is also defined globally as a *convecting flux operator* $\mathcal{C}$:

$$\mathbf{U} = \mathcal{C}\widetilde{\mathbf{u}} \tag{5.9}$$

where the notation $\widetilde{\mathbf{u}}$ - without the i superscript - is intended as a vector holding, in

sequence, all d components of the corresponding hybrid velocity field, i.e.

$$\widetilde{\mathbf{u}} = \begin{pmatrix} \widetilde{\mathbf{u}}^1 \\ \widetilde{\mathbf{u}}^2 \\ \cdots \\ \widetilde{\mathbf{u}}^d \end{pmatrix} = \begin{pmatrix} u^1_{F_1} \\ \cdots \\ u^1_{F_{n_F}} \\ u^2_{F_1} \\ \cdots \\ u^2_{F_{n_F}} \\ u^d_{F_1} \\ \cdots \\ u^d_{F_{n_F}} \end{pmatrix} . \tag{5.10}$$

Likewise, the superscript-free notation $\mathbf{u}$ shall denote a vector holding in sequence all d cell-averaged velocity components.

5.1.2 Hybrid momentum operator

There is a well-known similarity between the momentum equation (first equation in (5.1)) and the convection-diffusion-reaction equation (4.1). By isolating i-th component of the momentum equation and re-arranging its terms:

$$\nabla \cdot \left(-\nu \nabla u^i + \vec{U} u^i \right) + (\nabla p)^i = g^i \quad \text{in } \Omega \ , \tag{5.11}$$

the similarity becomes evident. The diffusivity is ν (isotropic), the reaction coefficient is null, the convecting flux is $\vec{U}$ itself, the scalar source term is g^i (the i-th component of $\vec{g}$) and the unknown scalar is the velocity component u^i. The difference is the additional pressure gradient term. Expression (5.11) can be rewritten in mixed form as

$$\begin{cases} \vec{V}^i = -\nu \nabla u^i + \vec{U} u^i + \vec{\delta}^i p \\ \nabla \cdot \vec{V}^i = g^i \end{cases} \quad \text{in } \Omega \tag{5.12}$$

where $\vec{\delta}^i$ is a vector of size d such that $\left(\vec{\delta}^i\right)_j = 1$ if $j = i$, zero otherwise. Formulation (5.12) follows the classical FV argument: integrating (5.11) over a control volume C yields

$$\int_C \left(\nabla \cdot \left(-\nu \nabla u^i + \vec{U} u^i \right) + (\nabla p)^i \right) \, dV = \int_C g^i \, dV \tag{5.13}$$

which, by applying the Gauss-Green formula to the left-hand side, is rewritten as

$$\int_{\partial C} \left(-\nu \nabla u^i + \vec{U} u^i \right) \cdot \vec{n}_{FC} \, dS + \int_{\partial C} p \ n^i_{FC} \, dS = \int_C g^i \, dV \ . \tag{5.14}$$

The first term in (5.14) corresponds to the sum of convective-diffusive fluxes across each face of C. Hence, at the discrete level, this suggests defining locally a total flux V^i_{FC} by adding to the MHFV convective-diffusive flux some discrete representation of the quantity $\int_F p\, n^i_{FC}\, dS$ - which is indeed the de Rham map of $\vec{\delta}^i p$ onto X^h. Droniou and Eymard [72] propose a scheme that, translated to MHFV notation, leads to the MHFV first-order pressure scheme (PRS1):

$$\left(\mathbf{V}^i\right)_{\partial C} = \underbrace{\mathbb{N}_C \left(u^i_C - u^i_F\right)_{\partial C} + \left(\mathbf{U}\right)_{\partial C} u^i_C}_{\text{convection-diffusion}} + \underbrace{\left(\mathbf{F}^i\right)_{\partial C} p_C}_{\text{pressure term}} \tag{5.15}$$

where the viscous term in $\mathbb{N}_C$ is built by using the scalar diffusivity tensor $\mathbb{K}_C = \nu_C \mathbb{I}$, and the notation $\mathbf{F}^i$ was introduced to denote a vector of cardinality n_F holding the i-th component of each face vector $\vec{F}$. The term $\int_F p\, n^i_{FC}\, dS$ is thus discretised as $p_C F^i_C$. Finally, local continuity of the total flux $\vec{V}^i$ is discretised as

$$\sum_{F \in \partial C} V^i_{FC} = |C|\, g^i_C \tag{5.16}$$

where g^i_C is the de Rham map of g^i onto Q^h, i.e.

$$g^i_C = \frac{1}{|C|} \int_C g^i\, dV \ . \tag{5.17}$$

Combining (5.16) and (5.15) leads to a MFV discrete formulation of (5.12) which is first-order accurate for the pressure variable [72]. The present work explores the possibility of extending (5.15) to second-order accuracy via the alternative scheme (PRS2):

$$\left(\mathbf{V}^i\right)_{\partial C} = \underbrace{\mathbb{N}_C \left(u^i_C - u^i_F\right)_{\partial C} + \left(\mathbf{U}\right)_{\partial C} u^i_C}_{\text{convection-diffusion}} + \underbrace{\left(\mathbf{F}^i \left(p_C + \nabla^{\mathcal{L},\mu}_C p \cdot (\vec{x}_F - \vec{x}_C)\right)\right)_{\partial C}}_{\text{second-order pressure term}} \ . \tag{5.18}$$

Formulation (5.18) replaces p_C in (5.15) with a one-sided reconstruction of the value of p at each face obtained via a piecewise-constant LSQ approximation of the pressure gradient on C:

$$\begin{aligned} \nabla^{\mathcal{L},\mu}_C p &= \underset{\vec{A} \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{C'} \frac{1}{\mu_{C'C}} \left(p_{C'} - p_C - \vec{A} \cdot (\vec{x}_{C'} - \vec{x}_C)\right)^2 \\ &= \left(\mathbb{X}^\mu_C\right)^{-1} \vec{b}_{p,\mu} \end{aligned} \tag{5.19}$$

with C' denoting cells neighbouring with C and $\mu_{CC'}$ a weighting factor between C and C'. The $d \times d$ $\mu_{CC'}$-weighted LSQ matrix is defined as

$$\left(\mathbb{X}_C^{\mu}\right)_{ij} = \sum_{C'} \frac{\left(x_{C'}^i - x_C^i\right)\left(x_{C'}^j - x_C^j\right)}{\mu_{CC'}} \tag{5.20}$$

and the LSQ right-hand side is

$$\vec{b}_{p,\mu} = \sum_{C'} \frac{\left(\vec{x}_{C'} - \vec{x}_C\right)\left(p_{C'} - p_C\right)}{\mu_{CC'}} \ . \tag{5.21}$$

It should be highlighted that the LSQ gradient (5.19) is a distinct operator from $\nabla_C^{\mathcal{L},\lambda}$, i.e. the λ_{FC}-weighted LSQ gradient appearing in the diffusion term for u^i (as well as in the convective term when second-order upwinding is employed) as defined in (3.68). The former operates on the pressure space (Q^h), hence involving exclusively cell-averaged degrees of freedom, while the latter hinges on both cell-averaged and hybrid values as shown in (3.50). Therefore the $\mu_{C'C}$ weights in (5.19) may be defined independently of λ_{FC}. Traditional geometric weighting schemes such as those suggested by Mavriplis [163] may be selected. In the present work, a distance-based weighting is chosen:

$$\mu_{C'C} = \left\|\vec{x}_{C'} - \vec{x}_C\right\|^2 \ . \tag{5.22}$$

Notice that the reconstructed face value of p must not be interpreted as a MHFV hybrid variable, as it is discontinuous across faces: for a face F common to $C+$ and $C-$, the value reconstructed from $C+$ will in general differ from that from $C-$. Hence the face value of p must not be considered as a true degree of freedom, but rather as part of the approximation made to derive the second-order accurate gradient operator for $\mathbf{p}$. In order to define a unified framework the generic notation $\overline{p}_{FC}$ is introduced to denote the scheme-dependent face value of p, namely

$$\overline{p}_{FC} = \begin{cases} p_C & \text{PRS1} \quad \text{(First-Order Pressure)} \\ p_C + \nabla_C^{\mathcal{L},\mu} p \cdot \left(\vec{x}_F - \vec{x}_C\right) & \text{PRS2} \quad \text{(Second-Order Pressure)} \end{cases} \ . \tag{5.23}$$

The total flux can thus be written in the generic form

$$\left(\mathbf{V}^i\right)_{\partial C} = \mathbb{N}_C \left(u_C^i - u_F^i\right)_{\partial C} + \left(\mathbf{U}\right)_{\partial C} u_C^i + \left(\overline{p}_{FC}\mathbf{F}^i\right)_{\partial C} \ . \tag{5.24}$$

At this point, regardless of the specific pressure scheme, full hybridisation is feasible on velocity components via the usual static condensation mechanism [40]. An expression of

u_C^i as a function of $\widetilde{\mathbf{u}}^i$ and $\mathbf{p}$ is obtained by replacing (5.24) in (5.16):

$$u_C^i = \frac{\left(\mathbb{N}_C{}^T \mathbb{1}, \left(\widetilde{\mathbf{u}}^i\right)_{\partial C}\right) + |C|\, g_C^i - \left(\left(\overline{p}_{FC}\mathbf{F}^i\right)_{\partial C}, \mathbb{1}\right)}{\left(\mathbb{N}_C \mathbb{1}, \mathbb{1}\right) + \left(\left(\mathbf{U}\right)_{\partial C}, \mathbb{1}\right)} \ . \tag{5.25}$$

Then, conservation (3.6) is imposed on the total flux, yielding the (i-th) MHFV *hybrid momentum operator*, written as

$$\mathcal{F}^i_{\nu,\vec{U}} \widetilde{\mathbf{u}}^i + \mathcal{G}^i \mathbf{p} = \widetilde{\mathbf{g}}^i \tag{5.26}$$

where $\mathcal{F}^i_{\nu,\vec{U}}$ is a hybrid convection-diffusion operator as shown in Section 4.1.5, $\mathcal{G}^i$ the i-th gradient component operator for the pressure space, and $\widetilde{\mathbf{g}}^i$ the i-th hybrid momentum right-hand side:

$$\begin{aligned} \left(\widetilde{\mathbf{g}}^i\right)_F \ = \ & -g^i_{C-} \left|C_-\right| \frac{\left(\mathbb{N}_{C-}\mathbb{1} + \left(\mathbf{U}\right)_{\partial C-}\right)_F}{\left(\mathbb{N}_{C-}\mathbb{1}, \mathbb{1}\right) + \left(\left(\mathbf{U}\right)_{\partial C-}, \mathbb{1}\right)} \\ & -g^i_{C+} \left|C_+\right| \frac{\left(\mathbb{N}_{C+}\mathbb{1} + \left(\mathbf{U}\right)_{\partial C+}\right)_F}{\left(\mathbb{N}_{C+}\mathbb{1}, \mathbb{1}\right) + \left(\left(\mathbf{U}\right)_{\partial C+}, \mathbb{1}\right)} \ . \end{aligned} \tag{5.27}$$

It is also useful at this point to introduce the following shorthand notation for the full momentum operator acting on all velocity components:

$$\boldsymbol{\mathcal{F}}_{\nu,\vec{U}} \widetilde{\mathbf{u}} + \boldsymbol{\mathcal{G}} \mathbf{p} = \widetilde{\mathbf{g}} \ . \tag{5.28}$$

A Picard linearisation of $\mathcal{F}^i_{\nu,\vec{U}}$ (i.e. the freezing of the convecting flow $\mathbf{U}$) allows to represent $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ and $\boldsymbol{\mathcal{G}}$ algebraically by block-diagonal matrices with the i-th block corresponding to $\mathcal{F}^i_{\nu,\vec{U}}$ and $\mathcal{G}^i$ respectively. In 3D they are defined as

$$\boldsymbol{\mathcal{F}}_{\nu,\vec{U}} = \begin{bmatrix} \mathcal{F}^1_{\nu,\vec{U}} & 0 & 0 \\ 0 & \mathcal{F}^2_{\nu,\vec{U}} & 0 \\ 0 & 0 & \mathcal{F}^3_{\nu,\vec{U}} \end{bmatrix} \ ; \quad \boldsymbol{\mathcal{G}} = \begin{bmatrix} \mathcal{G}^1 & 0 & 0 \\ 0 & \mathcal{G}^2 & 0 \\ 0 & 0 & \mathcal{G}^3 \end{bmatrix} \ . \tag{5.29}$$

5.1.3 Full MHFV Navier-Stokes operator

The last step consists in discretising the Navier-Stokes continuity equation (second equation in (5.1)), which is done via the usual Gauss-based divergence operator $\mathcal{D}$ defined in (3.15). The operator maps from X^h to Q^h, and thus is naturally applied to the discrete convecting flux $\mathbf{U}$. As anticipated, $\mathbf{U}$ is considered as a separate variable; however, combining $\mathcal{D}$ with the convecting flux operator $\mathcal{C}$ defined in (5.9) allows to build a (block-diagonal) divergence operator $\boldsymbol{\mathcal{D}} = \mathcal{D}\mathcal{C}$ acting directly on the hybrid velocity

components:

$$\mathcal{D}\mathbf{U} = \mathcal{D}\left(\mathcal{C}\widetilde{\mathbf{u}}\right) = \left(\mathcal{D}\mathcal{C}\right)\widetilde{\mathbf{u}} = \boldsymbol{\mathcal{D}}\widetilde{\mathbf{u}} \ . \tag{5.30}$$

The way $\boldsymbol{\mathcal{D}}$ operates locally on each cell is explicitly shown to be

$$\left(\boldsymbol{\mathcal{D}}\widetilde{\mathbf{u}}\right)_C = \sum_{F\in\partial C}\sum_{i=1}^{d} u_F^i F_C^i \ . \tag{5.31}$$

Notice that, by using (5.31) to discretise the continuity equation, local mass conservation is enforced in a strong (FV-like) sense. At this point the (Picard-linearised) MHFV *steady-state, incompressible Navier-Stokes operator* and system can be assembled:

$$\begin{bmatrix} \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} & \boldsymbol{\mathcal{G}} \\ \boldsymbol{\mathcal{D}} & 0 \end{bmatrix} \begin{pmatrix} \widetilde{\mathbf{u}} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{g}} \\ \mathbf{0} \end{pmatrix} \ . \tag{5.32}$$

As expected, (5.32) is an Oseen-type saddle-point system. The main non-linearity present in $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ is due to the convective term which depends on $\mathbf{U}$, which in turn depends on $\widetilde{\mathbf{u}}$. However further non-linearities $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ may exist, e.g. in case any of the solution-dependent stabilisation techniques described in Section 4.2 are employed.

It can be verified that, if the first-order pressure scheme PRS1 (5.15) is used, then the gradient operator $\boldsymbol{\mathcal{G}}$ is the adjoint of the divergence $\boldsymbol{\mathcal{D}}$, i.e. $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{D}}^T$: PRS1 causes the term $\left(\left(\overline{p}_{FC}\mathbf{F}^i\right)_{\partial C}, \mathbb{1}\right)$ in (5.25) to vanish, since

$$\left(\left(\overline{p}_{FC}\mathbf{F}^i\right)_{\partial C}, \mathbb{1}\right) = \left(\left(p_C\mathbf{F}^i\right)_{\partial C}, \mathbb{1}\right) = p_C \underbrace{\left(\left(\mathbf{F}^i\right)_{\partial C}, \mathbb{1}\right)}_{=0} \ . \tag{5.33}$$

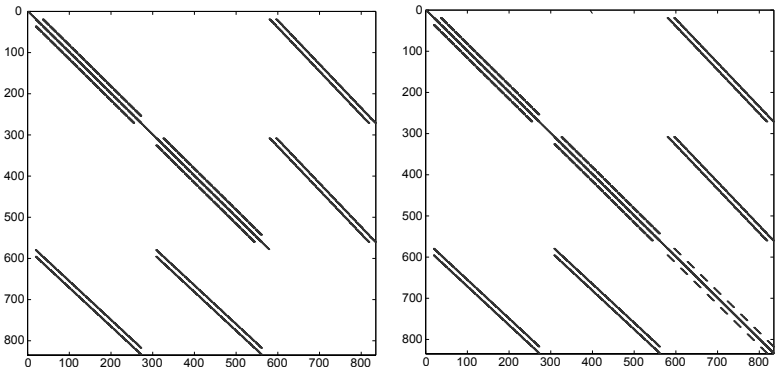
Therefore, when flux conservation is imposed, the only pressure-related term is the one that explicitly appears in the total flux (5.15), yielding a pressure gradient of the form

$$\left(\boldsymbol{\mathcal{G}}\mathbf{p}\right)_F = p_{C+}\vec{F}_{C+} - p_{C-}\vec{F}_{C-} \tag{5.34}$$

which is indeed the transpose of the divergence operator (5.31). As discussed in Chapter 3, such a property fits within a MVE framework where the mimetic nature of the method requires the two discrete operators to be adjoint of each other as their continuous counterparts are in the Gauss-Green formula. Conversely, the PRS2 strategy (5.18) breaks this symmetry thus violating one of the basic MVE principles. The advantage is that, if stable, PRS2 is expected to achieve a superior h-convergence on the pressure variable.

Another interesting remark is related to the presence of the so-called *zero block* for pressure in the discrete continuity equation. This may seem obvious since pressure does not appear in the continuity equation; however, this is not the case for many schemes.

For instance, the block is not zero for stabilised FE schemes that do not satisfy the LBB condition [40]. Classical collocated FV schemes also require an additional pressure-based stabilisation term: for example, the aforementioned Rhie-Chow interpolation scheme [203] is in fact a way of defining a convecting flux $\mathbf{U}$ at the faces which includes a pressure-related quantity and thus, when divergence is applied to such a flux, a pressure term appears in the continuity equation. In MHFV, it can be argued that a Rhie-Chow-



(a) without stabilisation (b) with stabilisation

Figure 5.1: Examples of Oseen sparsity patterns for a 2D Navier-Stokes problem [25].

like interpolation is not necessary: $\mathbf{U}$ exists as a degree of freedom, and it is directly related to the face-based velocity $\widetilde{\mathbf{u}}$ via (5.9), which in turn hinges directly on the cell-based pressure $\mathbf{p}$ via the momentum operator (5.28). Hence all degrees of freedom are naturally staggered, and it may be assumed that no further interpolation is required to stabilise the scheme. This will be further confirmed in Section 5.2.1, where it will be shown how the hybrid momentum equation can be rearranged in a way that highlights its resemblance to a FV-like Rhie-Chow interpolation. When looking at the sparsity pattern of the full Oseen problem (Figure 5.1), the absence of a pressure stabilisation term implies the presence of zeroes on the main diagonal, which will play a role in the devising of solution algorithms (Section 5.2).

5.1.4 Boundary conditions

A suitable definition of boundary conditions for the incompressible Navier-Stokes problem is an ongoing field of research for both the FV and FE communities, with several formulations being proposed over the years due to the vast diversity of problem definitions and the fact that, depending on the specific problem, it is not always immediate and obvious to provide a physically interpretable boundary condition [200]. Within the scope

of this thesis it is sufficient to focus on the three types most commonly found: inlet, wall and outlet, which were already discussed in Section 4.1.6 for convection-diffusion-reaction problems.

An inlet is implemented according to the FV *velocity inlet* definition, consisting in specifying inlet velocity components, which in MHFV translates to imposing a strong Dirichlet value to the hybrid velocity $\widetilde{\mathbf{u}}$ at the corresponding inlet faces. The convecting flux $\mathbf{U}$ is thus also imposed automatically via (5.9), meaning that inlet degrees of freedom for $\widetilde{\mathbf{u}}$ and $\mathbf{U}$ are effectively eliminated from the problem. A (no-slip) wall boundary condition is analogous: zero velocity - or the wall's velocity in case of moving wall - is imposed to faces corresponding to the wall. In this sense, wall and inlet conditions are mathematically identical.

Concerning the outlet, the MHFV framework lends itself to a straightforward implementation of the so-called *do-nothing* boundary condition, arguably the most established condition for FE methods [104, 105]. The strategy owes its name to the fact that the condition appears automatically in the FE weak formulation due to partial integration of the viscous term and the pressure gradient over outlet faces. More specifically, the basic formulation proposed by Heywood, Rannacher and Turek [121], which imposes

$$-\nu\frac{\partial\vec{U}}{\partial\vec{n}}+p\vec{n}=\vec{0} \tag{5.35}$$

at the outlet, is hereby adopted. In MHFV this is achieved in the momentum equation by imposing at the outlet boundary $\partial\Omega_h^O$, for each velocity component, that the sum of the viscous term (the diffusive flux) and the pressure flux be equal to zero:

$$\left(\mathbb{M}_C^{-1}\left(u_C^i-u_F^i\right)_{\partial C}\right)_F+\overline{p}_{FC}F_C^i=0\quad\forall F\in\partial\Omega_h^O\ ,\quad i=1\cdots d\ . \tag{5.36}$$

The cell-averaged u_C^i in (5.36) is then eliminated as previously shown for the convection-diffusion-reaction outlet condition (Section 4.1.6).

If the outlet plane is placed away from significant flow phenomena and oriented suitably, then it can reasonably be expected $\nabla\vec{U}\cdot\vec{n}=\vec{0}$, meaning that (5.36) implies zero pressure at the outlet: $\overline{p}_{FC}=0$. Conversely, for a problem where no outlet is present - such as the lid-driven test case, see Section 5.3.2 - the pressure field is only defined up to an additive constant. In that case one may choose to either fix the value p_C at an arbitrary cell, or renormalise the whole solution field $\mathbf{p}$ in order to have e.g. zero-average pressure across the domain. Throughout this work, the second solution is preferred.

5.2 Solution algorithms for incompressible Navier-Stokes

The Picard-linearised form of (5.28) facilitates solving the non-linear problem via *Picard iterations*, i.e. by using at iteration n a convecting flux computed via velocity values known from iteration $n-1$. The Picard linearisation is preferred mainly because of its ease of implementation. A suitable alternative would be to perform Newton iterations: Newton linearisation is known to be more robust and exhibits a higher rate of convergence, but each Newton iteration is computationally more expensive and its convergence depends on the initial solution estimate [183].

At each Picard iteration one must solve a saddle-point linear system. If an iterative algorithm is employed to solve the latter as well, since there is no interest in obtaining the exact Oseen solution at each Picard iteration, the two are typically performed at the same time in a "one-shot" fashion: this gives rise to many well-known CFD solution algorithms, which are in fact the combination of preconditioners for linearised Oseen-type systems with outer non-linear iterations.

Arguably the most popular solution strategy - and one of the earliest devised - to solve the FV-discretised Navier-Stokes is the SIMPLE algorithm (*Semi-Implicit Method for Pressure Linked Equations*) and all its variants, first developed by Spalding and Patankar [188]. In the past few decades, however, constant increases in computing power have driven the advances in algorithm development, specifically towards block preconditioners for Oseen-type systems more efficient than traditional SIMPLE-like strategies. Research has successfully produced a number of alternative algorithms, although mostly restricted so far to the FE community. Besides a few variants of classical ILU preconditioners [199, 240], many strategies have been put forward based on the so-called *approximate commutators* - in particular the *Least-Squares Commutator* (LSC) [80] and the *Pressure Convection-Diffusion* (PCD) commutator [69, 139, 214], based on the earlier *BFBt algorithm* [78]. Examples of other recently developed preconditioners include the *Augmented Lagrangian* (AL) approach [28–30], the *Artificial Compressibility* (AC) preconditioner [67], the *Grad-Div* (GD) preconditioner [67, 119], and those based on dimensional splitting along velocity components [26, 27]. Several interesting publications have also surfaced reviewing and comparing various Navier-Stokes preconditioners [25, 79, 179, 198, 211].

Many of these generic algorithms - both traditional and novel - can be adapted to MHFV. In the following sections a few existing strategies are outlined first for a generic

Oseen system written in the following notation:

$$\begin{bmatrix} \mathbb{F} & \mathbb{G} \\ \mathbb{D} & \mathbb{C} \end{bmatrix} \begin{pmatrix} \mathbf{u}_h \\ \mathbf{p}_h \end{pmatrix} = \begin{pmatrix} \mathbf{g}_h \\ \mathbf{s}_h \end{pmatrix} \tag{5.37}$$

and the following short-hand notation for (5.37):

$$\mathbb{A}\mathbf{w}_h = \mathbf{r}_h \tag{5.38}$$

where the subscript h indicates that vectors belong to some discretisation space (not necessarily the same for velocity and pressure). The adaptation of each algorithm to the MHFV framework is then described.

5.2.1 SIMPLEC

As anticipated, classical FV often make use of SIMPLE and SIMPLE-like solution algorithms [188, 229]. Their initial popularity can be primarily attributed to their segregated nature, implying that they require solving linear systems relatively small and better conditioned in comparison to the full linearised Oseen problem. The efficiency of SIMPLE-like strategies is however debatable [199]: they are shown to be stable in many cases, but they exhibit a rather poor convergence rate, they are prone to stagnation and their performance is known to be affected by mesh refinement.

Despite traditionally being presented from the "segregated algorithm" viewpoint, highlighting the fact that they solve separately for velocity and pressure, SIMPLE-like schemes can be seen as a way of preconditioning the discrete Oseen problem [211]. The generic Oseen matrix in (5.37) can be factorised as

$$\mathbb{A} = \begin{bmatrix} \mathbb{F} & \mathbb{G} \\ \mathbb{D} & \mathbb{C} \end{bmatrix} = \begin{bmatrix} \mathbb{F} & 0 \\ \mathbb{D} & -\mathbb{S} \end{bmatrix} \begin{bmatrix} \mathbb{I} & \mathbb{F}^{-1}\mathbb{G} \\ 0 & \mathbb{I} \end{bmatrix}, \tag{5.39}$$

where $\mathbb{I}$ is the identity matrix and $\mathbb{S}$ is known as *Schur complement* [155, 179]:

$$\mathbb{S} = \mathbb{D}\mathbb{F}^{-1}\mathbb{G} - \mathbb{C}. \tag{5.40}$$

This suggests a way of preconditioning the Oseen system. In fact, an exact Schur complement would provide an exact preconditioner, i.e. it would allow to solve the linearised (5.37), separately for $\mathbf{u}_h$ and $\mathbf{p}_h$, in one iteration only. However this would require inverting operator $\mathbb{F}$, which is unfeasible in practice. Therefore an approximate Schur

complement is computed instead:

$$\widehat{\mathbb{S}} = \mathbb{D}\widehat{\mathbb{F}^{-1}}\mathbb{G} - \mathbb{C} \tag{5.41}$$

where $\widehat{\mathbb{F}^{-1}}$ is some approximation of $\mathbb{F}^{-1}$, and a preconditioner is defined as

$$\mathbb{P} = \begin{bmatrix} \mathbb{F} & 0 \\ \mathbb{D} & -\widehat{\mathbb{S}} \end{bmatrix} \begin{bmatrix} \mathbb{I} & \widehat{\mathbb{F}^{-1}}\mathbb{G} \\ 0 & \mathbb{I} \end{bmatrix} . \tag{5.42}$$

A Richardson iteration for the preconditioned system $\mathbb{P}^{-1}\mathbb{A}\mathbf{w}_h = \mathbb{P}^{-1}\mathbf{b}_h$ is then devised based on the splitting $\mathbb{A} = \mathbb{P} - (\mathbb{P} - \mathbb{A})$, such that at the n-th iteration one solves:

$$\mathbb{P}\mathbf{w}_h^{n+1} = (\mathbb{P} - \mathbb{A})\,\mathbf{w}_h^n + \mathbf{r}_h \ , \tag{5.43}$$

i.e. :

$$\begin{aligned}
\begin{pmatrix} \mathbf{u}_h^{n+1} \\ \mathbf{p}_h^{n+1} \end{pmatrix} &= \begin{pmatrix} \mathbf{u}_h^n \\ \mathbf{p}_h^n \end{pmatrix} + \left(\begin{bmatrix} \mathbb{F} & 0 \\ \mathbb{D} & -\widehat{\mathbb{S}} \end{bmatrix} \begin{bmatrix} \mathbb{I} & \widehat{\mathbb{F}^{-1}}\mathbb{G} \\ 0 & \mathbb{I} \end{bmatrix} \right)^{-1} \begin{pmatrix} \mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n - \mathbb{F}\mathbf{u}_h^n \\ \mathbf{s}_h - \mathbb{D}\mathbf{u}_h^n \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{u}_h^n \\ \mathbf{p}_h^n \end{pmatrix} + \begin{bmatrix} \mathbb{I} & \widehat{\mathbb{F}^{-1}}\mathbb{G} \\ 0 & \mathbb{I} \end{bmatrix}^{-1} \begin{pmatrix} \mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \mathbf{u}_h^n \\ \widehat{\mathbb{S}}^{-1}\left(\mathbb{D}\mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \mathbf{s}_h\right) \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{u}_h^n \\ \mathbf{p}_h^n \end{pmatrix} \\
&+ \begin{pmatrix} \mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \mathbf{u}_h^n - \widehat{\mathbb{F}^{-1}}\mathbb{G}\left(\widehat{\mathbb{S}}^{-1}\left(\mathbb{D}\mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \mathbf{s}_h\right)\right) \\ \widehat{\mathbb{S}}^{-1}\left(\mathbb{D}\mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \mathbf{s}_h\right) \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \widehat{\mathbb{F}^{-1}}\mathbb{G}\left(\widehat{\mathbb{S}}^{-1}\left(\mathbb{D}\mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \mathbf{s}_h\right)\right) \\ \mathbf{p}_h^n + \widehat{\mathbb{S}}^{-1}\left(\mathbb{D}\mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right) - \mathbf{s}_h\right) \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{u}_h^{n+1/2} - \widehat{\mathbb{F}^{-1}}\mathbb{G}\boldsymbol{\delta}\mathbf{p}_h \\ \mathbf{p}_h^n + \boldsymbol{\delta}\mathbf{p}_h \end{pmatrix}
\end{aligned} \tag{5.44}$$

where $\mathbf{u}_h^{n+1/2} = \mathbb{F}^{-1}\left(\mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n\right)$ is the *velocity prediction*, and $\boldsymbol{\delta}\mathbf{p}_h = \widehat{\mathbb{S}}^{-1}\left(\mathbb{D}\mathbf{u}_h^{n+1/2} - \mathbf{s}_h\right)$ the *pressure correction*. The algorithm is stated as follows:

1. solve $\mathbb{F}\mathbf{u}_h^{n+1/2} = \mathbf{g}_h - \mathbb{G}\mathbf{p}_h^n$ (*predictor step* for velocity);
2. solve $\widehat{\mathbb{S}}\boldsymbol{\delta}\mathbf{p}_h = \mathbb{D}\mathbf{u}_h^{n+1/2} - \mathbf{s}_h$ (*pseudo-Laplacian* for pressure correction);
3. update pressure: $\mathbf{p}_h^{n+1} = \mathbf{p}_h^n + \boldsymbol{\delta}\mathbf{p}_h$;
4. update velocity: $\mathbf{u}_h^{n+1} = \mathbf{u}_h^{n+1/2} - \widehat{\mathbb{F}^{-1}}\mathbb{G}\boldsymbol{\delta}\mathbf{p}_h$ (*corrector step*);
5. compute new convecting flux; update operator $\mathbb{F}$ (Picard step).

The block-diagonal form of $\mathbb{F}$ in the linearised Navier-Stokes system allows to split step 1 into d separate linear solves, each corresponding to the momentum equation in its respective spatial direction; this is a further advantage of SIMPLE-like methods.

In its most basic implementation, SIMPLE approximates the inverse of $\mathbb{F}$ with the inverse of its main diagonal:

$$\mathbb{F}^{-1} \approx \widehat{\mathbb{F}^{-1}} = \left(\mathrm{DIAG}\left(\mathbb{F}\right)\right)^{-1}. \tag{5.45}$$

It has been observed [229] that, for steady-state Navier-Stokes, SIMPLE often requires heavily relaxing both momentum equation and pressure correction, which makes it a rather inefficient algorithm. For this reason, in this work the variant of SIMPLE known as SIMPLEC [226] (where the "C" stands for *Consistent*) is considered instead. In the steady-state case it operates by adding to the momentum equations in (5.37) some form of implicit relaxation by factor α:

$$\mathbb{F}^{\alpha} = \mathbb{F} + \alpha \mathrm{DIAG}\left(\mathbb{F}\right) \tag{5.46}$$

and subsequently approximating the inverse of $\mathbb{F}$ as:

$$\mathbb{F}^{-1} \approx \widehat{\mathbb{F}^{-1}} = \frac{1}{\alpha}\left(\mathrm{DIAG}\left(\mathbb{F}\right)\right)^{-1}. \tag{5.47}$$

Classical FV literature [229] provides the following interpretation: SIMPLEC, in the steady-state case, yields a pseudo-Laplacian pressure equation aimed at correcting the velocity increment $\left(\mathbf{u}_h^{n+1/2} - \mathbf{u}_h^n\right)$ rather than the velocity itself, and since it acts on relaxed velocity increments, it does not require relaxing the pressure correction step.

Despite the overall inefficiency of SIMPLE-type preconditioners, the practical advantages due to variable segregation make it worth implementing a version of SIMPLEC adapted to the MHFV framework. As mentioned in Section 5.1.1, in the case of a collocated FV scheme the Rhie-Chow momentum interpolation [203] is required in order to avoid decoupling of pressure and velocity and subsequent checker-board modes. In this work, the version of Rhie-Chow with relaxation proposed by Majumdar [157] is considered. Contrary to the original Rhie-Chow interpolation for SIMPLEC, the Majumdar formulation converges to a steady-state solution field which does not depend on the relaxation factor. It operates by deriving an expression for the velocity components at each face: $u_{h,F}$ involving both velocity and pressure variables, of the generic form

$$\beta_F u_{h,F} + \nabla_F p_h = \langle \beta u_h \rangle_F + \langle \nabla_C p_h \rangle_F \tag{5.48}$$

where $\langle \cdot \rangle_F$ represents a face-averaging procedure, and β_F is the resulting face-based

central coefficient subsequently used for implicit relaxation. The goal is thus to derive a MHFV expression analogous to (5.48) and identify the corresponding β_F. To do so, the full flux conservation expression at a face F is considered, with convective scheme HUPW2 (4.53) and pressure scheme PRS2 (5.18):

$$\begin{aligned}
&\underbrace{U^{dw}_{FC+}u^i_F + U^{uw}_{FC+}\left(u^i_{C+} + \nabla^{\mathcal{L},\lambda}_{C+}u^i\cdot(\vec{x}_F-\vec{x}_{C+})\right)}_{C_+\text{ convection}} + \underbrace{\overline{p}_{FC+}F^i_{C+}}_{C_+\text{ pressure}} \\
-\ &\underbrace{\nu_{C+}\nabla^{\mathcal{G}}_{C+}u^i\cdot\vec{F}_{C+} - \frac{1}{\lambda_{FC+}}\left(u^i_F - u^i_{C+} - \nabla^{\mathcal{L},\lambda}_{C+}u^i\cdot(\vec{x}_F-\vec{x}_{C+})\right)}_{C_+\text{ diffusion}} \\
+\ &\underbrace{U^{dw}_{FC-}u^i_F + U^{uw}_{FC-}\left(u^i_{C-} + \nabla^{\mathcal{L},\lambda}_{C-}u^i\cdot(\vec{x}_F-\vec{x}_{C-})\right)}_{C_-\text{ convection}} + \underbrace{\overline{p}_{FC-}F^i_{C-}}_{C_-\text{ pressure}} \\
-\ &\underbrace{\nu_{C-}\nabla^{\mathcal{G}}_{C-}u^i\cdot\vec{F}_{C-} - \frac{1}{\lambda_{FC-}}\left(u^i_F - u^i_{C-} - \nabla^{\mathcal{L},\lambda}_{C-}u^i\cdot(\vec{x}_F-\vec{x}_{C-})\right)}_{C_-\text{ diffusion}} = 0\ .
\end{aligned} \tag{5.49}$$

Denoting by λ_F the harmonic average of the λ_{FC} weights:

$$\lambda_F = \frac{2\lambda_{FC+}\lambda_{FC-}}{\lambda_{FC+}+\lambda_{FC-}}\ , \tag{5.50}$$

and introducing the definition of *local Reynolds number*

$$Re_F = \frac{\lambda_F\,|U_F|}{2}\ , \tag{5.51}$$

equation (5.49) may be rearranged and rescaled to give

$$\begin{aligned}
\frac{2\,(1+Re_F)}{\lambda_F}u^i_F - (p_{C+}-p_{C-})\,F^i_{C+} &= \\
\left(U^{uw}_{FC+} + \frac{1}{\lambda_{FC+}}\right)\left(u^i_{C+} + \nabla^{\mathcal{L},\lambda}_{C+}u^i\cdot(\vec{x}_F-\vec{x}_{C+})\right) &+ \\
\left(U^{uw}_{FC-} + \frac{1}{\lambda_{FC-}}\right)\left(u^i_{C-} + \nabla^{\mathcal{L},\lambda}_{C-}u^i\cdot(\vec{x}_F-\vec{x}_{C-})\right) &- \\
\left(\nu_{C+}\nabla^{\mathcal{G}}_{C+}u^i - \nu_{C-}\nabla^{\mathcal{G}}_{C-}u^i\right)\cdot\vec{F}_{C+} &+ \\
\left(\nabla^{\mathcal{L},\mu}_{C+}p\cdot(\vec{x}_F-\vec{x}_{C+}) - \nabla^{\mathcal{L},\mu}_{C-}p\cdot(\vec{x}_F-\vec{x}_{C-})\right)F^i_{C+}&
\end{aligned} \tag{5.52}$$

obtainable by taking into account the PRS2 scheme definition (5.18) and the trivial identities: $-\left(U^{dw}_{FC+}+U^{dw}_{FC-}\right) = |U_{FC+}| = |U_{FC-}| = |U_F|$ and $\vec{F}_{C-} = -\vec{F}_{C+}$. A parallelism can be drawn between (5.52) and the generic FV momentum interpolation (5.48): (5.52) is analogous to a FV (Majumdar) Rhie-Chow interpolation as it provides an explicit expression for the velocity component at the face u^i_F which combines an averaged value based on cell-centred quantities u^i_C with an additional pressure-dependent term. Hence

a suitable candidate to act as coefficient for a Majumdar-like relaxation is identified as

$$\beta_F = \frac{2\left(1 + Re_F\right)}{\lambda_F} \ . \tag{5.53}$$

The momentum equation is thus relaxed by using a modified convection-diffusion operator of the form:

$$\boldsymbol{\mathcal{F}}^{\alpha}_{\nu,\vec{U}} = \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} + \alpha\left(\mathrm{diag}\left(\beta_F\right)\right) \ . \tag{5.54}$$

Notice that (5.54) is a form of *inertial relaxation*: β_F is proportional to the local Reynolds number Re_F, meaning that stronger relaxation is applied in areas where convective phenomena dominate. Finally, the MHFV approximated Schur complement is defined as

$$\widehat{\mathcal{S}}^{\alpha} = \boldsymbol{\mathcal{D}}\left(\mathrm{diag}\left(\frac{1}{\alpha\beta_F}\right)\right)\boldsymbol{\mathcal{D}}^T \ . \tag{5.55}$$

Comparison with the generic Schur complement expression (5.41) raises two observations. First, matrix $\mathbb{C}$ does not have a MHFV equivalent since it corresponds to the pressure block in the continuity equation which, as shown before, is zero in MHFV. Second, the gradient operator $\boldsymbol{\mathcal{G}}$ is replaced with the transpose of the divergence operator $\boldsymbol{\mathcal{D}}^T$; as mentioned in Section 5.1.3, the two are identical when the pressure gradient is discretised via the PRS1 scheme (5.15). For PRS2 (5.18) that is no longer the case, meaning that formulation (5.55) introduces a further level of approximation in the Schur complement; however, since this approximation only affects the pressure correction step, the overall algorithm still converges. The choice is made in order not to degrade the sparsity pattern of the pseudo-Laplacian and ease the linear solve at the pressure correction step.

Algorithm 1 MHFV SIMPLEC

$n = 0$
Initialise $\widetilde{\mathbf{u}}^0$, $\mathbf{p}^0$
while not converged **do**
  Solve relaxed hybrid momentum equation (*velocity prediction*):
  $\boldsymbol{\mathcal{F}}^{\alpha}_{\nu,\vec{U}}\widetilde{\mathbf{u}}^{n+1/2} = \widetilde{\mathbf{g}} - \boldsymbol{\mathcal{G}}\mathbf{p}^n + \alpha\left(\mathrm{diag}\left(\beta_F\right)\right)\widetilde{\mathbf{u}}^n$;
  Solve Schur complement pseudo-Laplacian (*pressure correction*):
  $\widehat{\mathcal{S}}^{\alpha}\boldsymbol{\delta}\mathbf{p} = \boldsymbol{\mathcal{D}}\widetilde{\mathbf{u}}^{n+1/2}$;
  Update pressure:
  $\mathbf{p}^{n+1} = \mathbf{p}^n + \boldsymbol{\delta}\mathbf{p}$;
  Update hybrid velocity:
  $\widetilde{\mathbf{u}}^{n+1} = \widetilde{\mathbf{u}}^{n+1/2} - \left(\mathrm{diag}\left(\frac{1}{\alpha\beta_F}\right)\right)\boldsymbol{\mathcal{D}}^T\boldsymbol{\delta}\mathbf{p}$;
  Update convecting flux $\mathbf{U}$, cell-averaged velocity $\mathbf{u}$ and operator $\boldsymbol{\mathcal{F}}^{\alpha}_{\nu,\vec{U}}$;
  n=n+1
end while
return $\mathbf{u}$, $\widetilde{\mathbf{u}}$, $\mathbf{p}$

The overall iterative algorithm (Algorithm 1) is analogous to the generic one, with one last difference being that, since incompressible flow is being considered, the source term of the continuity equation is zero except for cells adjacent to Dirichlet boundaries.

5.2.2 Block-Coupled

The *Block-Coupled* (BCPL) solution strategy requires solving the linearised discrete Oseen problem (5.37) as it is - i.e. for velocity and pressure simultaneously - then update the value of the convecting flux via the newly computed velocity, re-assemble the convection-diffusion operator $\mathbb{F}$, and iterate. Hence a BCPL solver proceeds from one Picard iteration to the next, without any inner Oseen iterations.

BCPL typically exhibits superior convergence properties compared to any segregated algorithm, and it was shown [66] to be less dependent on grid quality and size, but solution of the fully coupled system is not trivial. Firstly, while segregated algorithms solve at each iteration multiple relatively small linear systems, the BCPL approach requires solving the larger linearised Oseen system (5.37). Storage of the full matrix - and potentially of its factorisation, depending on the chosen linear solver - can be prohibitive for large industrial cases, although recent advancements in memory capabilities and parallel computing help alleviate the problem. Secondly, it has been observed [234] that the saddle-point nature of system (5.37) poses a challenge for standard linear solvers especially in the presence of zero elements on the main diagonal in the discrete continuity equation ($\mathbb{C} = 0$), which is the case in MHFV. This causes the coupled system to be ill-conditioned, stiff and difficult to solve.

The implementation of a BCPL solver suitable for MHFV may require the development of a linear solver adapted to saddle-point systems of this nature. In order to solve the pressure zero-block problem, node renumbering techniques have been proposed [234, 239, 240] which avoid zero pivots when factorising the matrix; these can be applied to direct solvers as well as to ILU preconditioners [198, 199] for Krylov subspace linear solvers. In order to ease and accelerate linear solver convergence, algebraic multigrid methods are found to be effective [65, 66].

Linear solver theory is deemed beyond the scope if the present work, which is therefore limited to outlining the BCPL algorithm (Algorithm 2) under the assumption that the linearised Oseen problem can indeed be solved. All BCPL results shown in this work (Section 5.3.3) were obtained by direct solvers, which is however not a viable option for large industrial cases.

Algorithm 2 MHFV Block-Coupled

$n = 0$
Initialise $\widetilde{\mathbf{u}}^0$, $\mathbf{p}^0$
while not converged **do**
  Solve Picard-linearised Oseen problem (momentum may be relaxed if necessary):
  $\begin{bmatrix} \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} & \boldsymbol{\mathcal{G}} \\ \boldsymbol{\mathcal{D}} & 0 \end{bmatrix} \begin{pmatrix} \widetilde{\mathbf{u}}^{n+1} \\ \mathbf{p}^{n+1} \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{g}} \\ \mathbf{0} \end{pmatrix}$;
  Update convecting flux $\mathbf{U}$, cell-averaged velocity $\mathbf{u}$ and operator $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$;
  n=n+1
end while
return $\mathbf{u}$, $\widetilde{\mathbf{u}}$, $\mathbf{p}$

5.2.3 Augmented Lagrangian

The *Augmented Lagrangian* (AL) preconditioning scheme for Oseen-type problems was first presented by Benzi and Olshanskii [28], and further developed with several variants [29, 30, 119, 177]. AL-based preconditioners have been so far investigated mostly within FE frameworks, and have been proven to be theoretically almost optimal [30] in terms of grid and *Re*-dependency. It is thus worth adapting the AL methodology to MHFV.

AL literature always presents the method applied to LBB-stable FE schemes, implying that $\mathbb{C} = 0$ in the generic Oseen system (5.37); since this property is also verified in MHFV, the same assumption is made in the following exposition. The AL method starts by rewriting the discrete Oseen system (5.37) as

$$\begin{bmatrix} \mathbb{F}^\gamma & \mathbb{G} \\ \mathbb{D} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}_h \\ \mathbf{p}_h \end{pmatrix} = \begin{pmatrix} \mathbf{g}_h^\gamma \\ \mathbf{s}_h \end{pmatrix} \tag{5.56}$$

where

$$\mathbb{F}^\gamma = \mathbb{F} + \gamma \mathbb{G}\mathbb{W}^{-1}\mathbb{D} \tag{5.57}$$

and

$$\mathbf{g}_h^\gamma = \mathbf{g}_h + \gamma \mathbb{G}\mathbb{W}^{-1}\mathbf{s}_h, \tag{5.58}$$

with γ a positive augmentation factor and $\mathbb{W}$ an arbitrary SPD matrix. Systems (5.56) and (5.37) are equivalent, since (5.57) and (5.58) add to the velocity blocks a term proportional to the residual of the continuity equation. Hence the AL method can be interpreted as the addition to the momentum equations of a penalisation term proportional to the mass imbalance, which is driven to zero for a converged solution.

As observed by Benzi et al. [29], an advantage of using an AL formulation (besides it being largely insensitive to grid size and regularity, as well as Reynolds number) is that the

issue of finding a good approximation for the Schur complement (5.40) is circumvented. If the augmentation factor is large enough, then the penalisation term will prevail on the operator $\mathbb{F}$ itself, thus justifying the approximation

$$(\mathbb{F}^\gamma)^{-1} \approx \widehat{(\mathbb{F}^\gamma)^{-1}} = \left(\gamma \mathbb{G}\mathbb{W}^{-1}\mathbb{D}\right)^{-1} \tag{5.59}$$

which yields an approximate Schur complement of the form:

$$\widehat{\mathbb{S}} = \mathbb{D}\left(\gamma \mathbb{G}\mathbb{W}^{-1}\mathbb{D}\right)^{-1}\mathbb{G} = \frac{1}{\gamma}\mathbb{W}. \tag{5.60}$$

The generic AL iterative procedure is outlined as follows:

1. solve $\quad \mathbb{F}^\gamma \mathbf{u}_h^{n+1} = \mathbf{g}_h^\gamma - \mathbb{G}\mathbf{p}_h^n \qquad$ (penalised momentum equation);
2. solve $\quad \frac{1}{\gamma}\mathbb{W}\boldsymbol{\delta}\mathbf{p}_h = \mathbb{D}\mathbf{u}_h^{n+1} - \mathbf{s}_h \qquad$ (pressure correction);
3. update pressure: $\quad \mathbf{p}_h^{n+1} = \mathbf{p}_h^n + \boldsymbol{\delta}\mathbf{p}_h$;
4. compute new convecting flux; update augmented operator $\mathbb{F}^\gamma$ (Picard step).

In FE the matrix $\mathbb{W}$ is often chosen to be the pressure mass matrix or, for practical reasons, a diagonal approximation of it (usually the main diagonal, or a lumped mass matrix [30]). In that case, the pressure correction (step 2) involving the approximate Schur complement (5.60) only requires inverting a diagonal matrix, i.e. it doesn't call for a linear solve.

Despite their great potential, AL-based preconditioners suffer from a few considerable drawbacks. Chief amongst them is the fact that, as shown by Benzi and Olshanskii [28], too large values of γ cause the penalised block $\mathbb{F}^\gamma$ to become increasingly ill-conditioned since $\mathbb{G}\mathbb{W}^{-1}\mathbb{D}$ is a singular matrix, and therefore increasingly challenging for linear solvers. On the other hand, the approximated Schur complement (5.60) is only close to the exact one if γ is large enough, hence if γ is chosen too small, the overall algorithm may underperform or not converge at all. A trade-off between these two extremes is required, possibly combined with inexact solves of the augmented momentum equations. A further disadvantage of AL is that the augmented momentum operator $\mathbb{F}^\gamma$ is no longer block-diagonal, because each of the velocity components contributes to the penalisation term of the momentum equations in all spatial dimensions. Therefore, step 1 in the procedure above entails a single coupled linear solve for all d velocity components simultaneously and, unlike SIMPLEC, it cannot be decoupled into segregated smaller systems, at least for the basic AL formulation. Attempts have been made to circumvent the problem, see e.g. Benzi et al. [29] who suggest a modified AL formulation where the off-diagonal

penalisation blocks are treated explicitly in order to allow for dimensional splitting.

The AL preconditioner for the MHFV Navier-Stokes scheme is implemented as follows, and the corresponding iterative scheme outlined in Algorithm 3. The AL penalisation procedure (5.57) applied to the hybrid convection-diffusion operator is expressed as

$$\boldsymbol{\mathcal{F}}^{\gamma}_{\nu,\vec{U}} = \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} - \gamma\mu\boldsymbol{\mathcal{D}}^T \left(\text{diag}\left(\frac{1}{|C|}\right)\right)\boldsymbol{\mathcal{D}} \ , \tag{5.61}$$

(with μ a scaling factor discussed below) whereas the hybrid right-hand side $\widetilde{\mathbf{g}}$ remains unmodified since there is no source term for the continuity equation. The matrix diag $(|C|)$ - a diagonal matrix holding values of cell volumes - is intended to play in (5.61) the role of $\mathbb{W}$ in the generic formulation (5.57). In a FV-like framework such as MHFV this is interpretable as the equivalent of the FE pressure mass matrix, and since it is diagonal it can be inverted with no further approximation, making the pressure correction step trivial and computationally cheap.

Notice that, as already done for the SIMPLEC Schur complement (Section 5.2.1), the MHFV AL augmentation term makes use of $\boldsymbol{\mathcal{D}}^T$ rather than $\boldsymbol{\mathcal{G}}$. It was already highlighted how the two are identical for the PRS1 scheme (5.15). It was chosen to do so regardless of the specific pressure scheme in place, since a second-order accurate $\boldsymbol{\mathcal{G}}$ as defined in PRS2 (5.18) would further increase the complexity - in terms of connectivity and subsequent sparsity pattern - of the already challenging augmented operator (5.61). This choice does not in any way affect the order of accuracy of the solution itself: the AL algorithm ultimately drives $\boldsymbol{\mathcal{D}}\widetilde{\mathbf{u}}$, and therefore the penalisation term, to zero, so that the converged flow field satisfies the original non-augmented Navier-Stokes problem (5.32). A further feature specific to the MHFV AL formulation is the additional scaling factor

Algorithm 3 MHFV Augmented Lagrangian

$n = 0$
Initialise $\widetilde{\mathbf{u}}^0$, $\mathbf{p}^0$
while not converged **do**
  Solve augmented hybrid momentum equation:
  $\boldsymbol{\mathcal{F}}^{\gamma}_{\nu,\vec{U}}\widetilde{\mathbf{u}}^{n+1} = \widetilde{\mathbf{g}} - \boldsymbol{\mathcal{G}}\mathbf{p}^n$;
  Compute pressure correction:
  $\boldsymbol{\delta}\mathbf{p} = -\gamma\mu\left(\text{diag}\left(\frac{1}{|C|}\right)\right)\boldsymbol{\mathcal{D}}\widetilde{\mathbf{u}}^{n+1}$;
  Update pressure:
  $\mathbf{p}^{n+1} = \mathbf{p}^n + \boldsymbol{\delta}\mathbf{p}$;
  Update convecting flux $\mathbf{U}$, cell-averaged velocity $\mathbf{u}$ and operator $\boldsymbol{\mathcal{F}}^{\gamma}_{\nu,\vec{U}}$;
  n=n+1
end while
return $\mathbf{u}$, $\widetilde{\mathbf{u}}$, $\mathbf{p}$

μ by which the penalisation term is multiplied in (5.61). This is done in order to obtain a range of values for γ that work reasonably well regardless of the specific mesh size and problem physics. Following the guidelines of Benzi and Olshanskii [28] for the magnitude of the penalisation coefficient, μ is set to scale with the velocity:

$$\mu = \max\left(\left\|\vec{u}_C\right\|, C \in \Omega_h\right) \tag{5.62}$$

where $\left\|\vec{u}_C\right\|$ is the magnitude of the cell-averaged velocity: $\left\|\vec{u}_C\right\| = \sqrt{\sum_{i=1}^{d}\left(u_C^i\right)^2}$.

5.3 Validation of MHFV for incompressible Navier-Stokes

5.3.1 h-convergence for Navier-Stokes

As shown in Section 5.1.2, the Picard-linearised MHFV momentum equation combines a convection-diffusion operator and the pressure gradient scheme; since the former was already analysed in all its variants in Chapter 4, validation of the MHFV incompressible, steady-state Navier-Stokes scheme in terms of h-convergence will be focused on the pressure scheme. The test case proposed here is defined over the 2D square domain $\Omega =]0,1[\times]0,1[$; an artificial source term is added to the momentum equation in order to enforce the following manufactured solution:

$$\vec{U}_{ex}(x,y) = \begin{pmatrix} u_{ex}(x,y) \\ v_{ex}(x,y) \end{pmatrix} = \begin{pmatrix} 3x^2 - 2y \\ -6xy - x^2 \end{pmatrix}, \tag{5.63}$$

$$p_{ex}(x,y) = x + y^2, \tag{5.64}$$

where the exact velocity field (5.63) is evidently divergence-free, hence satisfying the continuity equation. Kinematic viscosity is set to $\nu = 10^{-3}$ in order to obtain a fairly convection-dominated problem. Centred convective schemes do fail (at least on the coarser meshes in the sequence), therefore the chosen convective scheme is second-order upwind with ULSQR stabilisation (Section 4.2.5). The λ_{FC} weights for the viscous term are of type OVRN (3.84). Dirichlet boundary conditions are applied to the velocity throughout, meaning that the pressure field $\mathbf{p}$ is only defined up to a constant and, as discussed in Section 5.1.4, the solver will renormalise it to a zero-average field; in order to compute the error correctly, the same procedure is applied to the exact solution p_{ex}.

The Navier-Stokes equations are solved over the two usual mesh sequences: polygonal distorted (Figure 3.6) and Cartesian. Errors in L^2 norm for cell-averaged components of $\mathbf{u}$ (u and v) and cell-averaged pressure $\mathbf{p}$ are plotted in Figure 5.2 and 5.3, for the PRS1

and PRS2 scheme respectively. Corresponding values and convergence rates are reported in Tables 5-A through 5-D.

Table 5-A: First-order pressure scheme (PRS1) - polygonal distorted mesh: errors and convergence rates.

h	$\epsilon(u_C)$	Rate	$\epsilon(v_C)$	Rate	$\epsilon(p_C)$	Rate
4.449 E^{-2}	3.713 E^{-2}	–	8.321 E^{-3}	–	8.564 E^{-2}	–
2.206 E^{-2}	1.549 E^{-2}	1.246	4.155 E^{-3}	0.990	4.009 E^{-2}	1.082
1.466 E^{-2}	8.787 E^{-3}	1.387	2.646 E^{-3}	1.104	2.323 E^{-2}	1.335
1.097 E^{-2}	5.796 E^{-3}	1.435	1.885 E^{-3}	1.170	1.538 E^{-2}	1.422
8.769 E^{-3}	4.220 E^{-3}	1.417	1.472 E^{-3}	1.104	1.134 E^{-2}	1.361

Table 5-B: Second-order pressure scheme (PRS2) - polygonal distorted mesh: errors and convergence rates.

h	$\epsilon(u_C)$	Rate	$\epsilon(v_C)$	Rate	$\epsilon(p_C)$	Rate
4.449 E^{-2}	3.817 E^{-4}	–	1.831 E^{-4}	–	2.639 E^{-3}	–
2.206 E^{-2}	9.400 E^{-5}	1.998	3.505 E^{-5}	2.357	6.278 E^{-4}	2.047
1.466 E^{-2}	4.213 E^{-5}	1.964	1.553 E^{-5}	1.992	2.748 E^{-4}	2.022
1.097 E^{-2}	2.398 E^{-5}	1.944	9.123 E^{-6}	1.835	1.577 E^{-4}	1.915
8.769 E^{-3}	1.554 E^{-5}	1.937	6.111 E^{-6}	1.789	1.031 E^{-4}	1.898

Table 5-C: First-order pressure scheme (PRS1) - Cartesian mesh: errors and convergence rates.

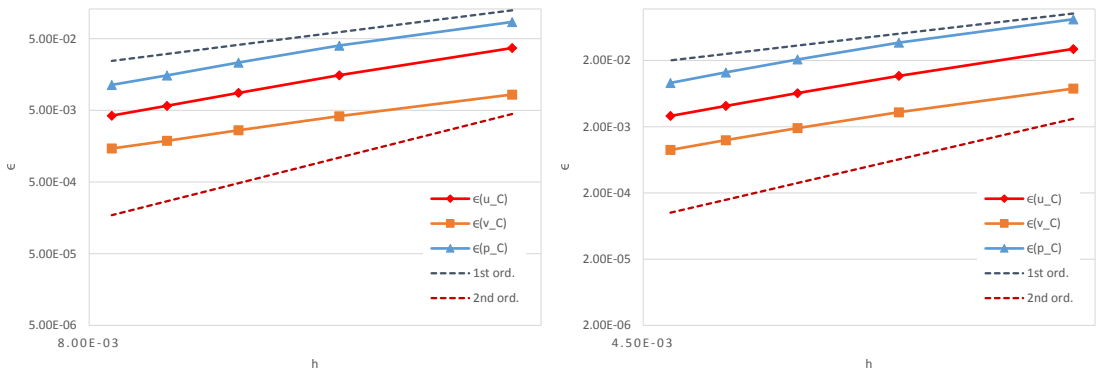
h	$\epsilon(u_C)$	Rate	$\epsilon(v_C)$	Rate	$\epsilon(p_C)$	Rate
2.564 E^{-2}	2.99 E^{-2}	–	7.553 E^{-3}	–	8.383 E^{-2}	–
1.266 E^{-2}	1.17 E^{-2}	1.327	3.315 E^{-3}	1.167	3.721 E^{-2}	1.151
8.403 E^{-3}	6.43 E^{-3}	1.464	1.910 E^{-3}	1.345	2.078 E^{-2}	1.421
6.289 E^{-3}	4.13 E^{-3}	1.525	1.255 E^{-3}	1.449	1.323 E^{-2}	1.558
5.025 E^{-3}	2.91 E^{-3}	1.560	8.955 E^{-4}	1.504	9.180 E^{-3}	1.629

Table 5-D: Second-order pressure scheme (PRS2) - Cartesian mesh: errors and convergence rates.

h	$\epsilon(u_C)$	Rate	$\epsilon(v_C)$	Rate	$\epsilon(p_C)$	Rate
$2.564\ \mathrm{E}^{-2}$	$3.662\ \mathrm{E}^{-4}$	–	$1.547\ \mathrm{E}^{-4}$	–	$2.690\ \mathrm{E}^{-3}$	–
$1.266\ \mathrm{E}^{-2}$	$7.158\ \mathrm{E}^{-5}$	2.313	$2.444\ \mathrm{E}^{-5}$	2.615	$6.499\ \mathrm{E}^{-4}$	2.013
$8.403\ \mathrm{E}^{-3}$	$2.610\ \mathrm{E}^{-5}$	2.462	$9.289\ \mathrm{E}^{-6}$	2.360	$2.838\ \mathrm{E}^{-4}$	2.022
$6.289\ \mathrm{E}^{-3}$	$1.240\ \mathrm{E}^{-5}$	2.568	$4.783\ \mathrm{E}^{-6}$	2.291	$1.578\ \mathrm{E}^{-4}$	2.025
$5.025\ \mathrm{E}^{-3}$	$6.907\ \mathrm{E}^{-6}$	2.608	$2.877\ \mathrm{E}^{-6}$	2.265	$1.001\ \mathrm{E}^{-4}$	2.029

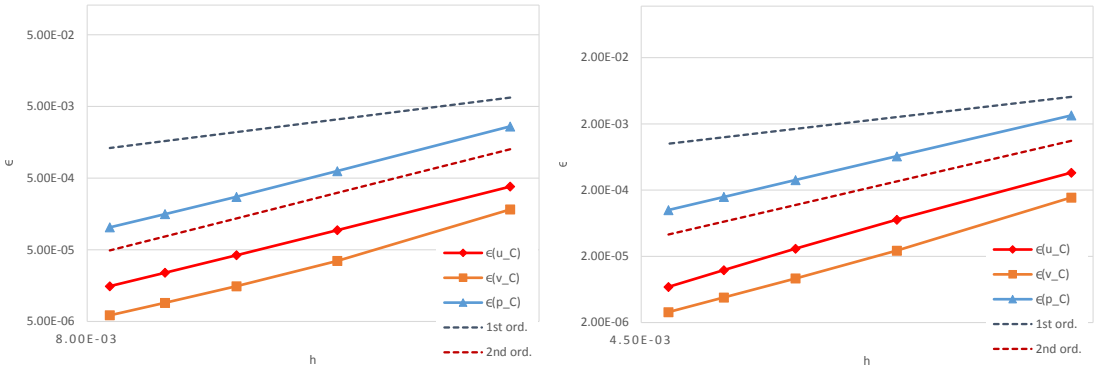
Numerical results show that the PRS2 scheme introduced in this thesis behaves according to its nominal second-order accuracy and, at least for this basic test case, it appears to be rid of any instabilities. There are no significant differences in the convergence behaviour of $\mathbf{p}$ between different mesh types, indicating that the pressure scheme does not hamper the overall grid-independent nature of MHFV operators. As expected, components of $\mathbf{u}$ also exhibit second-order convergence thanks to the ULSQR scheme - in fact, for this simple test case they exhibit accuracy above second-order on Cartesian meshes.

Concerning the PRS1 scheme $\mathbf{p}$, it appears to perform above its nominal first-order accuracy - and improve with grid refinement - on both mesh types. However, results also highlight a coupling between velocity and pressure errors: PRS1 has an adverse effect on the accuracy of $\mathbf{u}$, namely by degrading its order of convergence especially on coarser meshes. The PRS1 scheme noticeably leads to consistently higher errors on all variables on both grid sequences.



(a) polygonal distorted mesh (b) Cartesian mesh

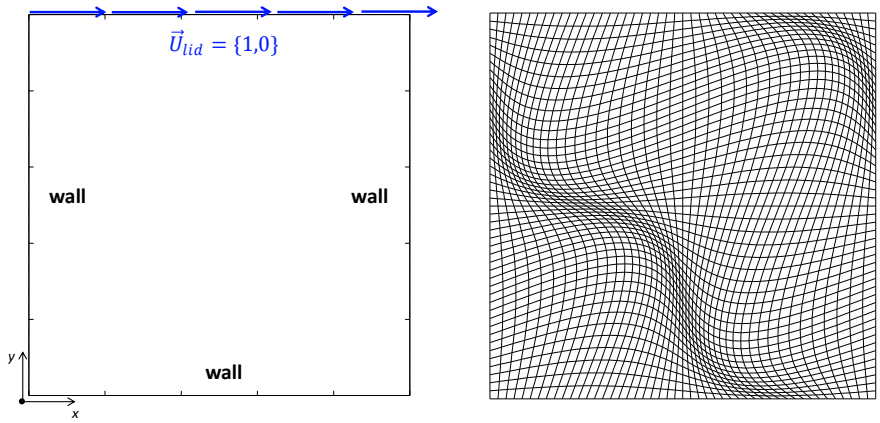
Figure 5.2: First-order pressure scheme (PRS1): h-convergence.



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 5.3: Second-order pressure scheme (PRS2): h-convergence.

5.3.2 Lid-driven cavity test case



(a) boundary conditions (b) mesh type

Figure 5.4: Lid-driven test case setup.

A further validation of the MHFV Navier-Stokes solver is performed against the well-known 2D *lid-driven cavity* benchmark case, which is set up as in Figure 5.4(a) over the square domain $\Omega =]0,1[\times]0,1[$. The forced x-velocity on the "lid" (the top side of the domain) is set to $u_{lid} = 1$, while viscosity ν is varied in order to match $Re = 10^2$, 10^3 and 10^4 which will allow comparison with reference results from previous literature [94]. MHFV schemes are set to: OVRN weights for the viscous term; ULSQR for the convective term; PRS2 for pressure. Each equation is solved down to a scaled residual of

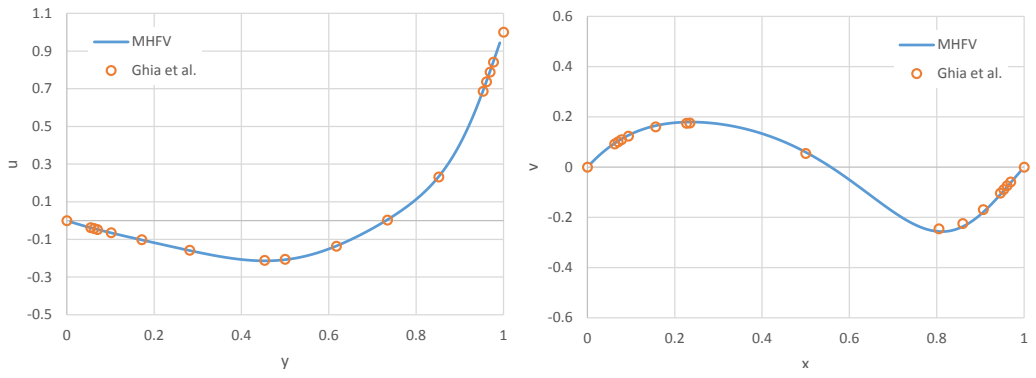
10^{-4}. The scaled residual R_u^i for the i-th momentum equation is defined as follows:

$$R_u^i = \frac{\sum_{F\in\Omega_h} \left|\left(\widetilde{\mathbf{r}^i}\right)_F\right|}{\sum_{F\in\Omega_h} \left(\left(\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}\right)_{FF} \left\|\vec{\widetilde{\mathbf{u}}}_F\right\|\right)} \tag{5.65}$$

where $\widetilde{\mathbf{r}^i}$ is the i-th hybrid momentum residual and $\left(\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}\right)_{FF}$ the central hybrid velocity coefficient for face F; this definition is based on what is commonly done in most commercial FV-based CFD solvers. Concerning the continuity residual R_c, it is less trivial to define a suitable scaling procedure. CFD solvers often scale by the largest divergence value computed at an early iteration, but this scaling procedure depends on the initial solution estimate. For the results presented here, the continuity residual is left unscaled, i.e.

$$R_c = \sum_{C\in\Omega_h} \left|(\boldsymbol{\mathcal{D}}\widetilde{\mathbf{u}})_C\right| \ . \tag{5.66}$$

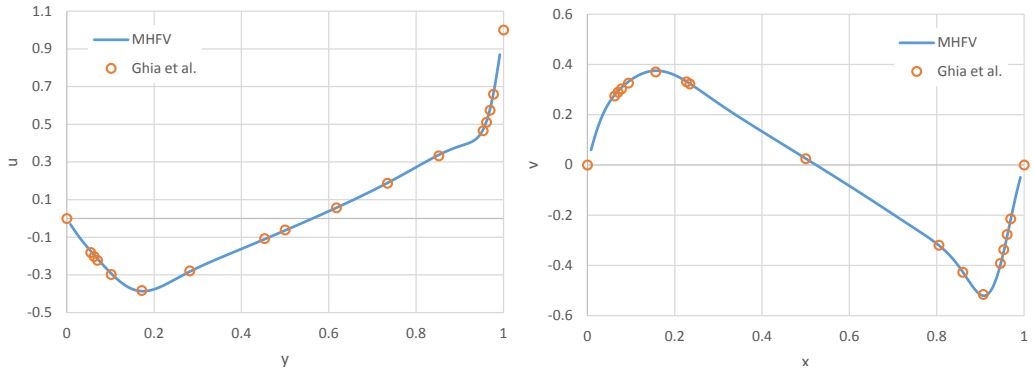
Results are compared to those reported by Ghia et al. [94], which are computed over a uniform Cartesian grid of size 129×129. MHFV simulations are run on a quadrilateral distorted mesh (Figure 5.4(b)) in order to introduce non-orthogonal cells and therefore verify grid independence. This type of grid features two sets of faces whose centres of gravity are aligned with the x and y axes of symmetry of the domain, locations where Ghia et al. [94] report velocity components v and u, respectively; this allows direct comparison with MHFV hybrid velocity components computed along those axes. Since the distortion pattern causes local mesh coarsening and refinement in certain (arbitrarily located) areas, MHFV simulations are run on a slightly coarser grid (119×119), which gives an averaged cell-to-cell centre distance $h_{avg} \approx 8.87\ \mathrm{E}^{-3}$ roughly equivalent to that used for the reference results.



(a) u along y-axis

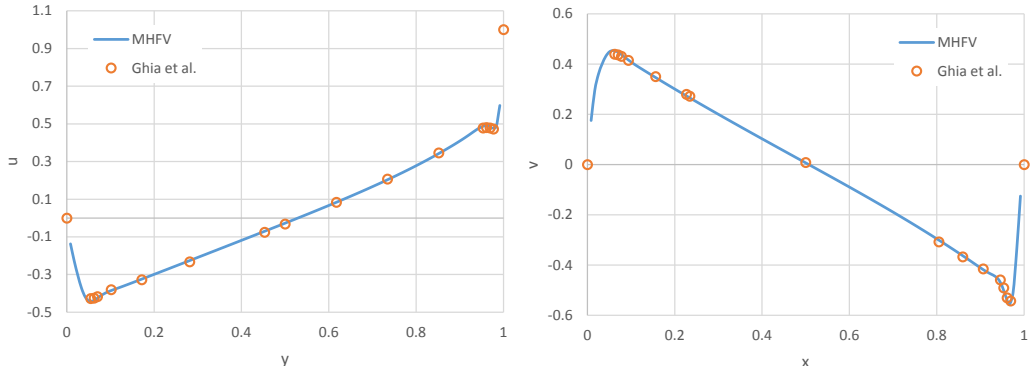
(b) v along x-axis

Figure 5.5: Lid-driven cavity, $Re = 10^2$: results comparison.



(a) u along y-axis (b) v along x-axis

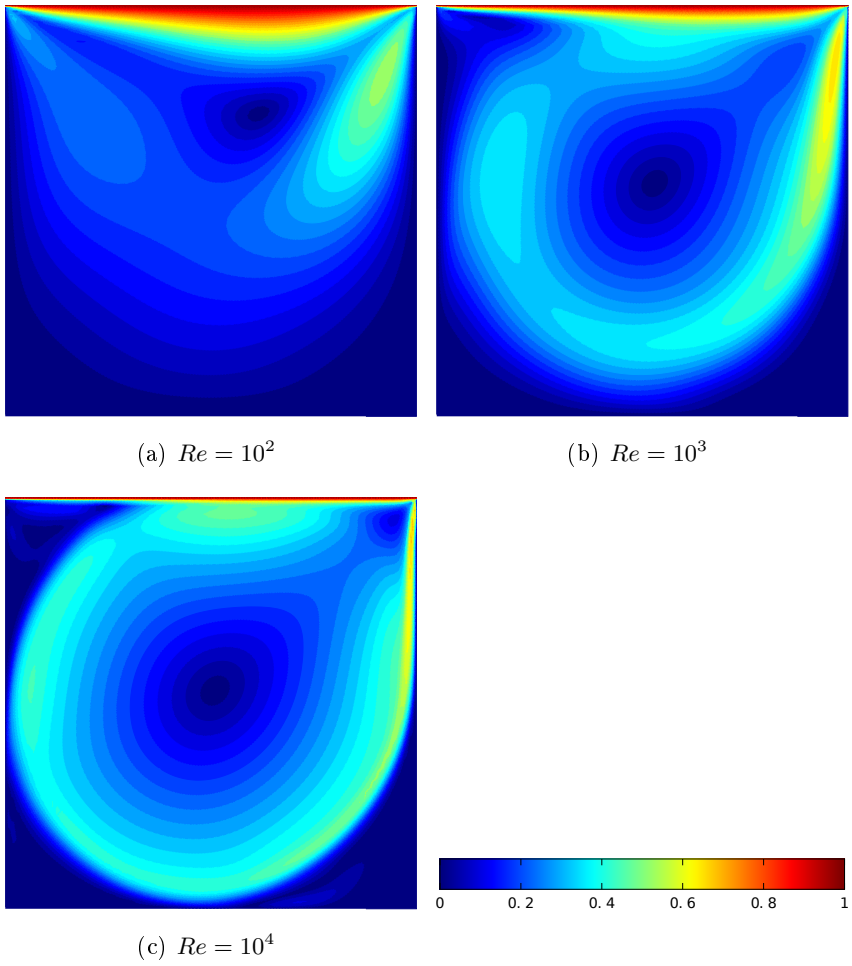
Figure 5.6: Lid-driven cavity, $Re = 10^3$: results comparison.



(a) u along y-axis (b) v along x-axis

Figure 5.7: Lid-driven cavity, $Re = 10^4$: results comparison.

The two sets of values mentioned above (u_F along a vertical line and v_F along a horizontal line passing through the geometric centre of the cavity) are extracted from the MHFV solution field and plotted together with those found in literature in Figures 5.5 through 5.7. Excellent agreement is observed for all three values of Re; MHFV results appear to be completely unaffected by the underlying distorted grid pattern, as also qualitatively confirmed by solution contours (Figure 5.8). Further results on this test case over an under-resolved grid are reported in Appendix B.



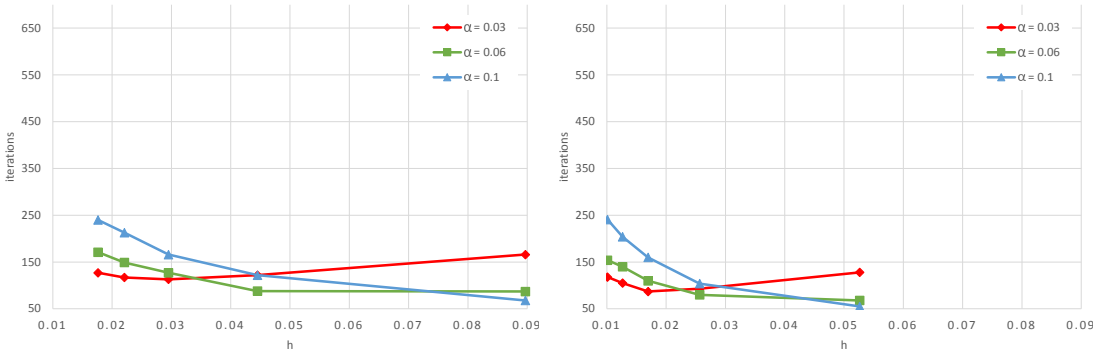
(a) $Re = 10^2$ (b) $Re = 10^3$

(c) $Re = 10^4$

Figure 5.8: Lid-driven cavity solution field (velocity magnitude $\left\|\vec{U}\right\|$) for different values of Re.

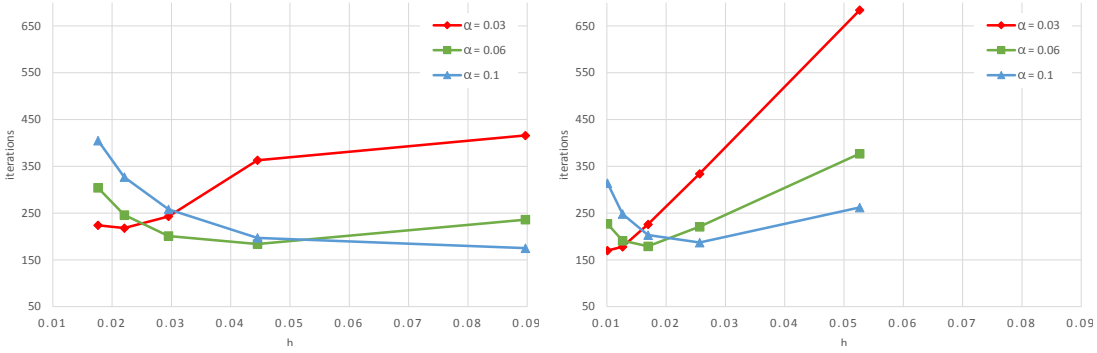
5.3.3 Algorithm performance

The 2D lid-driven cavity case is also used to test and compare performance of the various solution algorithms derived in Section 5.2. It is interesting in particular to analyse how each algorithm responds to changes in the grid (in terms both of coarseness and orthogonality) as well as in the problem's physics (in terms of global Reynolds number). The MHFV discretisation scheme is identical to that described in Section 5.3.2. Each algorithm is tested as usual on two sets of progressively refined grids (one polygonal distorted, one Cartesian), each for two different Reynolds values $Re = 10^2$ and $Re = 10^3$. All linear systems are solved via linear solvers. The stopping criterion is satisfied when all scaled residuals fall below 10^{-4}. For SIMPLEC, each test case is run for three different values of the relaxation factor α: 3 E^{-2}, 6 E^{-2} and 1 E^{-1}, in order to verify whether the



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 5.9: SIMPLEC, $Re = 10^2$: performance.



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 5.10: SIMPLEC, $Re = 10^3$: performance.

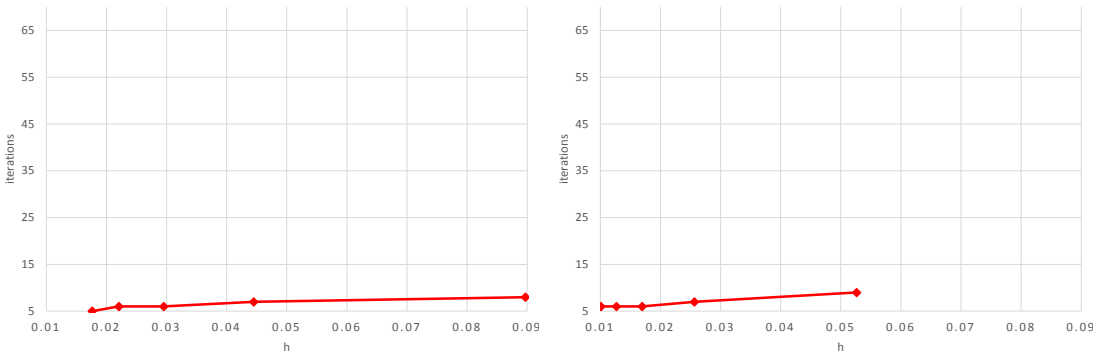
optimal value of α depends on the other parameters. Results are shown in Figure 5.9 and 5.10. There is a noticeable trend observable on both grid types and for both Re values: on coarser meshes, increasing the relaxation factor α improves performance. This trend is inverted as meshes are refined, with smaller values of α resulting in lower iteration counts on finer grids. In all runs each value of α presents a tipping point with respect to grid refinement, suggesting that for a fixed value of α there exists an optimal mesh size h minimising the iteration count and, vice-versa, for a fixed h there exists an optimum α. This optimum seems to be only marginally affected by mesh type; it is however heavily dependent on the problem's physics, and while a parametric study might lead to a generic procedure to determine the optimal α in function of h and Re, results based on this test case alone would be insufficient to ensure the generality of such a procedure.

As expected, the iteration count itself appears to be affected by the physics of the problem, being higher for a higher Re. The difference is however relatively marginal, at least for the values tested here, and selecting an appropriate α generally allows to

converge in a number of iterations in the same order of magnitude regardless of Re, especially on finer grids. Dependency of SIMPLEC on mesh type is also relatively small but noticeable, with the algorithm generally performing better on Cartesian meshes for an equal h and optimal α. Again, this is especially true for finer grids.

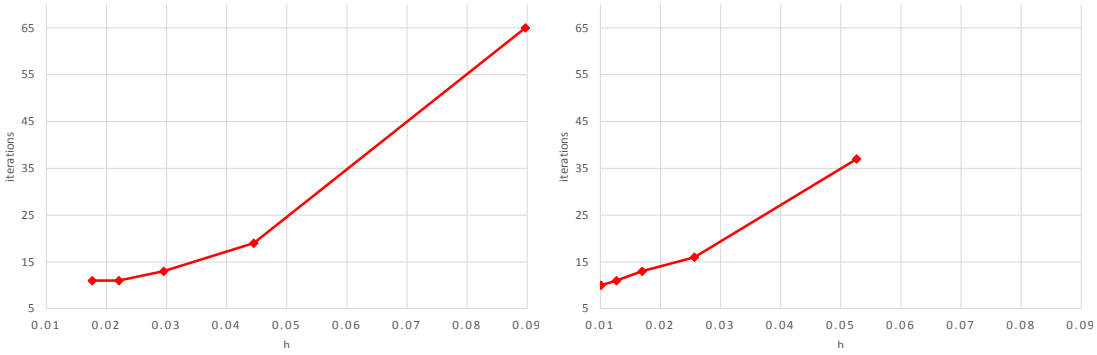
In general, the trend suggests that using small values of α on finer meshes will yield faster convergence. There is however a limit on how small the relaxation factor can be, lest the overall algorithm diverge: the approximated Schur complement (5.55) scales with α^{-1}; therefore, in a realistic engineering application - where h must typically be small enough to yield reliable results - SIMPLEC is ultimately limited in performance, in the sense that once the smallest possible α is determined that doesn't lead to divergence, SIMPLEC performance will then deteriorate if further mesh refinement is desired.

Compared to SIMPLEC, performance of the BCPL algorithm (Figure 5.11 and 5.12) is very favourable. Regardless of mesh type and size, and independently of Re, the



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 5.11: Block-Coupled (BCPL), $Re = 10^2$: performance.

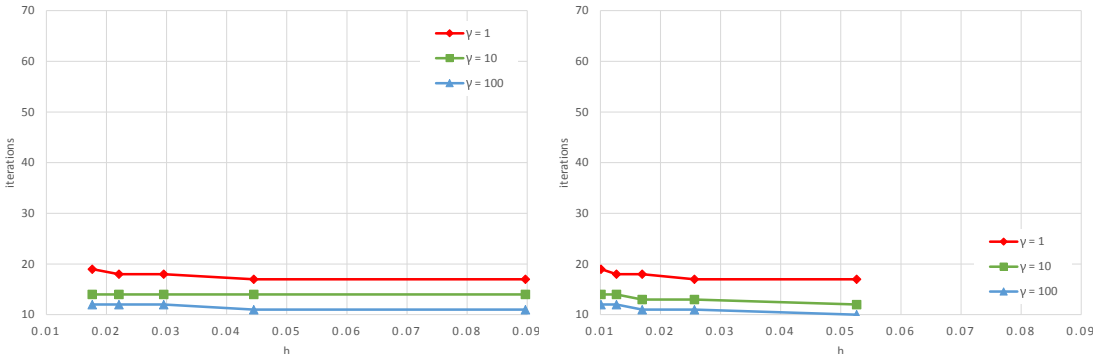


(a) polygonal distorted mesh (b) Cartesian mesh

Figure 5.12: Block-Coupled (BCPL), $Re = 10^3$: performance.

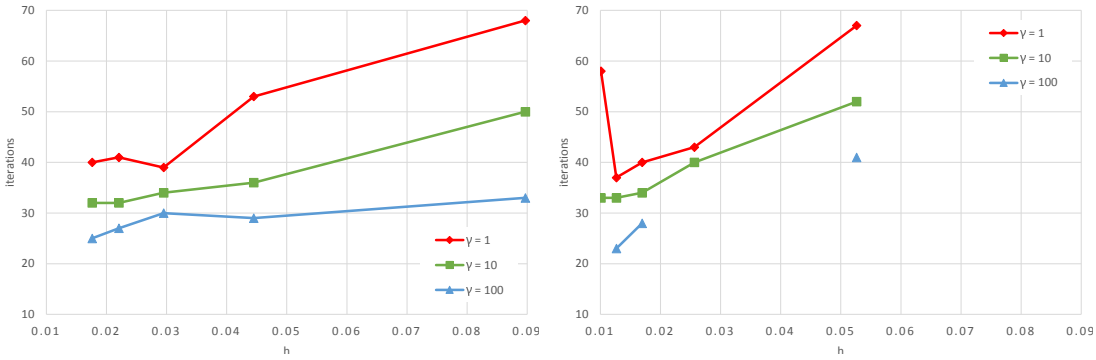
iteration count is always considerably lower. More specifically, at $Re = 10^2$ the iteration count is consistently small (below 10) and does not appear to be affected by mesh size or quality. At $Re = 10^3$ an asymptotic behaviour with respect to mesh size is observed, namely the iteration count is higher on coarser meshes and tends to settle around a value in the order of 10 as the mesh is refined; this behaviour is identical over both mesh types. Evidence thus suggests that BCPL is independent of mesh quality and mesh size (barring the coarsest mesh cases, where it underperforms slightly compared to SIMPLEC with optimal α), and only marginally affected by an increase of Re.

Concerning the AL algorithm, it is tested for three different values of penalisation factor γ: 1, 10 and 100; results are reported in Figure 5.13 and 5.14. The curves show that there is no definite correlation between iteration count and grid size nor grid type, suggesting that the MHFV AL implementation is completely mesh independent - barring an underperformance on coarse meshes for $Re = 10^3$, similarly to what observed for BCPL. As already observed on the other preconditioners, there is however a slight



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 5.13: Augmented Lagrangian (AL), $Re = 10^2$: performance.



(a) polygonal distorted mesh (b) Cartesian mesh

Figure 5.14: Augmented Lagrangian (AL), $Re = 10^3$: performance.

performance degradation for higher Re values. The iteration count itself remains in the order of 10, hence much lower than that of SIMPLEC and fairly close to BCPL results, thus confirming the near-optimality of the AL algorithm.

Results also highlight how higher values of γ consistently reduce the total number of iterations. This is expected because, as discussed in Section 5.2.3, when γ is high the convection-diffusion operator $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ in (5.61) becomes negligible compared to the penalisation term, and therefore the diagonal approximate Schur complement used in the AL pressure correction step is close to the exact one. However, too high values of γ will cause the augmented operator $\boldsymbol{\mathcal{F}}^{\gamma}_{\nu,\vec{U}}$ to be nearly singular, thus hindering the linear solve for velocity prediction. This is also highlighted by the present results: setting $\gamma = 100$ leads to divergence over certain Cartesian meshes at high Re, which explains the missing data in Figure 5.14(b); this is due to the direct linear solver failing to factorise the augmented operator. Keeping γ in the order of 1 appears to be a reasonable choice.

5.3.4 Benchmark against classical Finite Volumes

It was discussed in Section 2.4 how certain classical FV numerical artefacts (e.g. limiters) may prevent the full convergence of solution algorithms to steady-state. Alleviating this problem by improving the spatial discretisation was a key motivating factor for the present work. In order to verify whether the MHFV Navier-Stokes scheme succeeds in that sense, a comparison is carried out against the commercial FV solver ACE+[1].

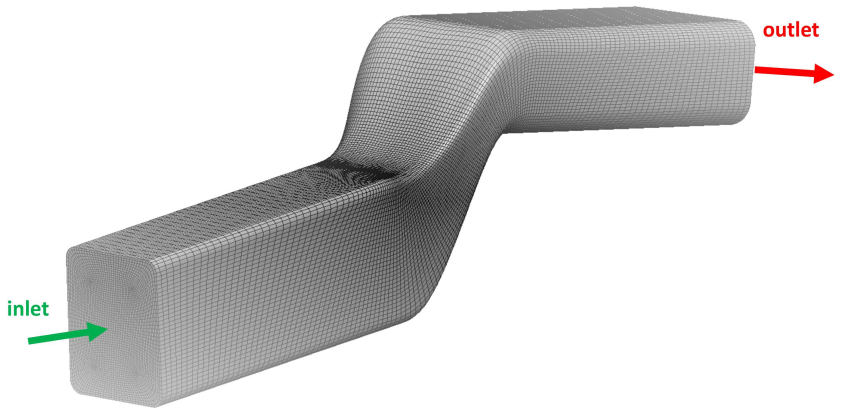


Figure 5.15: S-bend test case: geometry, mesh and boundary conditions.

The 3D *S-bend* test case is considered (Figure 5.15): an internal flow case simulating the incompressible, steady-state flow of air ($\nu = 1.589\ \mathrm{E}^{-5}\ m^2/s$) through a 3D S-shaped

[1] https://www.esi-group.com/software-solutions/virtual-environment/cfd-multiphysics/ace-suite/cfd-ace

duct used in the HVAC system of a passenger car. The duct has a rectangular cross-section with variable height in the order of 0.1 m. Imposing a velocity of 0.12 m/s at the inlet gives a Reynolds number $Re \approx 480$ (based on the height of the duct at the inlet): low enough to ensure that the flow regime remains laminar and that no recirculation occurs at the outlet, but slightly higher than that usually considered in the literature ($Re \approx 300$, see e.g. [241]) in order to test both solvers under more challenging conditions. The mesh contains 465976 hexahedral cells. MHFV discretisation strategies are set to: OVRN weight type for viscous terms; ULSQR for convective terms; PRS2 for pressure. Linear systems are solved via a ILU-preconditioned GMRES solver. In ACE+, a second-order upwinding strategy with a Barth-Jespersen limiter [11] is selected, which is known to cause stalling of the solution algorithm (see Section 4.2.3). A SIMPLE-type solution algorithm is used in both solvers.

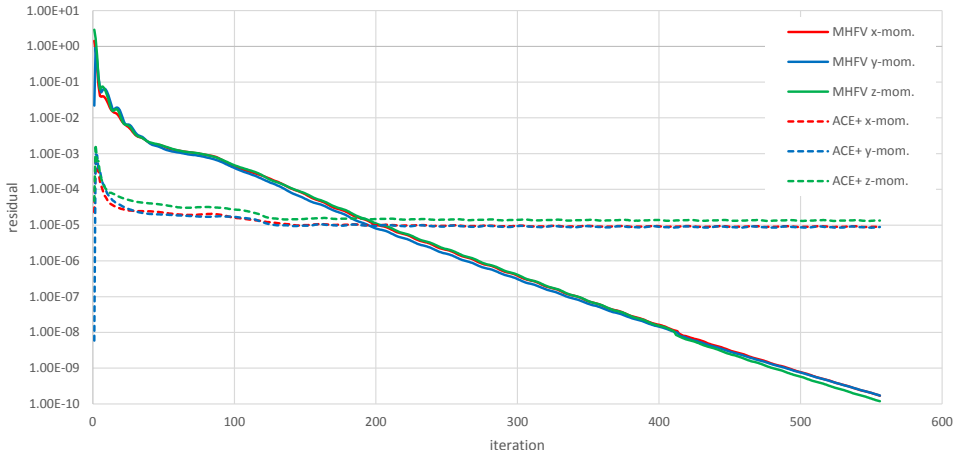


Figure 5.16: Convergence history of momentum residuals: MHFV (second-order, ULSQR-stabilised) vs. classical FV (second-order, Barth-Jespersen limiter).

The convergence history for the scaled x, y and z momentum residuals is plotted in Figure 5.16. A direct comparison of values is not significant considering that the scaling factor for the two schemes is not equivalent. However, there is a visible difference in the trend: while ACE+ stalls and enters limit cycle oscillations shortly after the 100-th iteration, MHFV converges steadily. At the 560-th iteration, the maximum absolute values of the unscaled hybrid momentum residual vectors ($\widetilde{\mathbf{r}_u}^x$, $\widetilde{\mathbf{r}_u}^y$, $\widetilde{\mathbf{r}_u}^z$) and corresponding L^2 norms are

$$\max|\widetilde{\mathbf{r}_u}^x| = 2.773\ \mathrm{E}^{-16} \quad ; \quad \max|\widetilde{\mathbf{r}_u}^y| = 3.184\ \mathrm{E}^{-16} \quad ; \quad \max|\widetilde{\mathbf{r}_u}^z| = 1.838\ \mathrm{E}^{-16}$$
$$\sqrt{\|\widetilde{\mathbf{r}_u}^x\|^2} = 1.103\ \mathrm{E}^{-14} \quad ; \quad \sqrt{\|\widetilde{\mathbf{r}_u}^y\|^2} = 9.455\ \mathrm{E}^{-15} \quad ; \quad \sqrt{\|\widetilde{\mathbf{r}_u}^z\|^2} = 9.324\ \mathrm{E}^{-15}$$

indicating that the MHFV solution is approaching machine precision accuracy. The test case itself is not strictly representative of a real industrial necessity (for example one may

switch to a first-order scheme in ACE+ and obtain better convergence with acceptable results). Nevertheless the comparison demonstrates how, at equal order of accuracy, an improved spatial discretisation scheme such as MHFV can have a positive effect in terms of convergence to steady-state. Contour plots of the MHFV solution fields are reported in Figure 5.17 and 5.18. To the author's knowledge, the present work is the first to present results for a MFV-type Navier-Stokes scheme on a 3D model other than a manufactured solution.

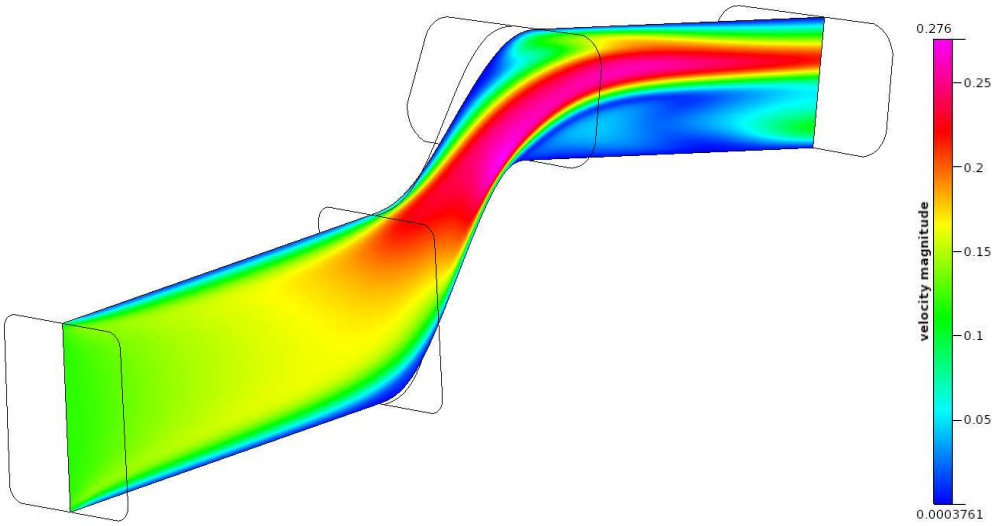


Figure 5.17: S-bend: velocity magnitude (m/s), cross-section.

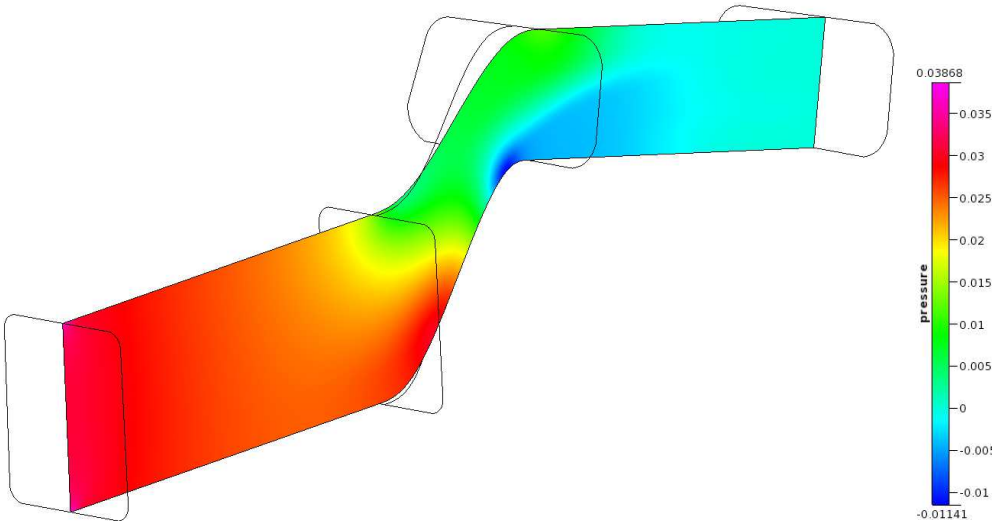


Figure 5.18: S-bend: pressure (N/m^2), cross-section.

Chapter 6

MHFV Discrete Adjoint Navier-Stokes

6.1 Assembly of the adjoint system

Having derived and validated in Chapter 5 the MHFV incompressible, steady-state Navier-Stokes scheme, it is now possible to consider its discrete adjoint counterpart. This chapter discusses the tools and procedures employed for adjoint assembly and solution. As anticipated in Section 2.3.3, the present work follows the guidelines of the *Equational Differentiation* (ED) philosophy which demands a clear distinction between the equations being solved (Section 6.1) and the solution algorithms (Section 6.2). This often calls for an explicit assembly of the full adjoint system, which in turn requires evaluating, for a converged solution, the tangent matrix (*Jacobian*) of the primal and the vector of partial derivatives of the objective function J with respect to each flow variable, which will serve as right-hand side of the adjoint system.

6.1.1 Full MHFV discrete adjoint Navier-Stokes

Before delving into the details of how an adjoint system can be assembled in practice, it is beneficial to perform a preliminary analysis of the form taken by the MHFV discrete adjoint Navier-Stokes. The residual vector of the Navier-Stokes problem (5.32) is denoted as

$$\mathbf{r}_{NS} = \begin{pmatrix} \widetilde{\mathbf{r}_u} \\ \mathbf{r}_p \end{pmatrix} = \begin{bmatrix} \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} & \boldsymbol{\mathcal{G}} \\ \boldsymbol{\mathcal{D}} & 0 \end{bmatrix} \begin{pmatrix} \widetilde{\mathbf{u}} \\ \mathbf{p} \end{pmatrix} - \begin{pmatrix} \widetilde{\mathbf{g}} \\ \mathbf{0} \end{pmatrix} \tag{6.1}$$

where $\widetilde{\mathbf{r}_u}$ and $\mathbf{r}_p$ identify residuals of the hybrid momentum and the continuity equation, respectively. The Jacobian matrix $\mathcal{A}$ is thus defined as

$$\mathcal{A} = \begin{bmatrix} \dfrac{\partial \widetilde{\mathbf{r}_u}}{\partial \widetilde{\mathbf{u}}} & \dfrac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{p}} \\ \dfrac{\partial \mathbf{r}_p}{\partial \widetilde{\mathbf{u}}} & \dfrac{\partial \mathbf{r}_p}{\partial \mathbf{p}} \end{bmatrix} . \tag{6.2}$$

A more specific form of $\mathcal{A}$ can be deduced via a few preliminary considerations. The residual of the hybrid momentum equation in the i-th spatial dimension is isolated and written as

$$\widetilde{\mathbf{r}_u}^i = \mathcal{F}^i_{\nu,\vec{U}} \widetilde{\mathbf{u}}^i + \mathcal{G}^i \mathbf{p} - \widetilde{\mathbf{g}}^i \ . \tag{6.3}$$

Expression (6.3) shows that the contribution of the pressure variable $\mathbf{p}$ to the i-th momentum residual $\widetilde{\mathbf{r}_u}^i$ is solely due to the gradient operator $\mathcal{G}^i$ which, as shown in Section 5.1.2, is linear regardless of the specific pressure scheme. It follows that

$$\frac{\partial \widetilde{\mathbf{r}_u}^i}{\partial \mathbf{p}} = \mathcal{G}^i \ , \tag{6.4}$$

implying that all matrix entries of the Jacobian block $\frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{p}}$ are identical to those of the primal block-diagonal operator $\boldsymbol{\mathcal{G}}$ itself.

The same cannot be said for the velocity block, due to the non-linearity of the convection-diffusion operator $\mathcal{F}^i_{\nu,\vec{U}}$. As discussed in Section 5.1.3 the main source of non-linearity is the convective term, which depends on the convecting flux $\mathbf{U}$ which in turn depends on $\widetilde{\mathbf{u}}$ by virtue of the convecting flux definition (5.9). Further non-linearities may be present if the chosen convective scheme is also solution-dependent, such as flux limiting or WLSQR (Section 4.2.3 and 4.2.4 respectively). These dependencies are formally expressed as

$$\mathcal{F}^i_{\nu,\vec{U}} = \mathcal{F}^i_{\nu,\vec{U}} \left(\mathbf{U}\left(\widetilde{\mathbf{u}}\right), \widetilde{\mathbf{u}}^i \right) \ . \tag{6.5}$$

Moreover, since each face-based degree of freedom of $\mathbf{U}$ depends on all d velocity components at the corresponding face, it follows that each Jacobian block $\frac{\partial \widetilde{\mathbf{r}_u}^i}{\partial \widetilde{\mathbf{u}}^j}$ is in general non-zero. Hence the complete Jacobian velocity block, hereby denoted by $\overline{\boldsymbol{\mathcal{F}}}$, is not block-diagonal, unlike the Picard-linearised operator $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ from which it stems. For a generic 3D case, $\overline{\boldsymbol{\mathcal{F}}}$ takes the form

$$\overline{\boldsymbol{\mathcal{F}}} = \begin{bmatrix} \dfrac{\partial \widetilde{\mathbf{r}_u}^1}{\partial \widetilde{\mathbf{u}}^1} & \dfrac{\partial \widetilde{\mathbf{r}_u}^1}{\partial \widetilde{\mathbf{u}}^2} & \dfrac{\partial \widetilde{\mathbf{r}_u}^1}{\partial \widetilde{\mathbf{u}}^3} \\ \dfrac{\partial \widetilde{\mathbf{r}_u}^2}{\partial \widetilde{\mathbf{u}}^1} & \dfrac{\partial \widetilde{\mathbf{r}_u}^2}{\partial \widetilde{\mathbf{u}}^2} & \dfrac{\partial \widetilde{\mathbf{r}_u}^2}{\partial \widetilde{\mathbf{u}}^3} \\ \dfrac{\partial \widetilde{\mathbf{r}_u}^3}{\partial \widetilde{\mathbf{u}}^1} & \dfrac{\partial \widetilde{\mathbf{r}_u}^3}{\partial \widetilde{\mathbf{u}}^2} & \dfrac{\partial \widetilde{\mathbf{r}_u}^3}{\partial \widetilde{\mathbf{u}}^3} \end{bmatrix} = \begin{bmatrix} \overline{\mathcal{F}}^{11} & \overline{\mathcal{F}}^{12} & \overline{\mathcal{F}}^{13} \\ \overline{\mathcal{F}}^{21} & \overline{\mathcal{F}}^{22} & \overline{\mathcal{F}}^{23} \\ \overline{\mathcal{F}}^{31} & \overline{\mathcal{F}}^{32} & \overline{\mathcal{F}}^{33} \end{bmatrix} \tag{6.6}$$

where all derivative values are evaluated at convergence of the primal flow field. The extra-diagonal blocks are the most evident difference between the adjoint velocity block and the corresponding linearised primal. They can be interpreted by analogy with continuous adjoint theory as a discrete equivalent of the *Adjoint Transpose Convection* (ATC) [95, 138, 181]: *convection* because, in continuous adjoint theory, its form closely resembles that of a convection operator; *transpose* because the adjoint system matrix is in fact the transpose of the Jacobian.

Concerning the adjoint continuity residual, it was mentioned in Section 5.1.3 that the MHFV scheme is presumed LBB-stable and thus features a zero-block for the pressure variable; as a consequence, $\frac{\partial \mathbf{r}_p}{\partial \mathbf{p}} = 0$, i.e. the zero-block is maintained in the Jacobian. The remaining term $\frac{\partial \mathbf{r}_p}{\partial \widetilde{\mathbf{u}}}$ stems from the divergence operator $\boldsymbol{\mathcal{D}}$ which is linear with respect to the hybrid velocity components, hence it corresponds to $\boldsymbol{\mathcal{D}}$ itself, similarly to what observed above for $\boldsymbol{\mathcal{G}}$ in the momentum equation. Finally, the right-hand side of the discrete adjoint system $\mathbf{g}^*$ holds partial derivatives of the cost function J with respect to all degrees of freedom of the primal:

$$\mathbf{g}^* = \begin{pmatrix} \widetilde{\mathbf{g}_u}^* \\ \mathbf{g}_p^* \end{pmatrix} = \begin{pmatrix} \left(\dfrac{\partial J}{\partial \widetilde{\mathbf{u}}}\right)^T \\ \left(\dfrac{\partial J}{\partial \mathbf{p}}\right)^T \end{pmatrix} . \tag{6.7}$$

The full MHFV adjoint Navier-Stokes system is then written as:

$$\begin{bmatrix} \overline{\boldsymbol{\mathcal{F}}} & \boldsymbol{\mathcal{G}} \\ \boldsymbol{\mathcal{D}} & 0 \end{bmatrix}^T \begin{pmatrix} \widetilde{\mathbf{u}}^* \\ \mathbf{p}^* \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{g}_u}^* \\ \mathbf{g}_p^* \end{pmatrix} \tag{6.8}$$

where $\widetilde{\mathbf{u}}^*$ and $\mathbf{p}^*$ are the adjoint hybrid velocity and the adjoint pressure, respectively, to which the notation conventions described in Section 5.1.1 shall apply.

6.1.2 Reverse assembly of the adjoint system

The Jacobian and right-hand side appearing in the adjoint system (6.8) are in theory rather straightforward to assemble: the expressions of both the residual $\mathbf{r}$ for each equation and the objective function J are a combination of elementary mathematical operations whose partial derivatives can be derived analytically and then coded; this is the hand-derived approach (ED1 in Section 2.3.3). As already mentioned, hand-derivation is affected by several drawbacks. Besides being tedious and error-prone (especially for the Jacobian, where the complexity of the derivation quickly grows as the underlying operators extend to higher-order stencils [173]), the process cannot be automated, meaning

that each time a new feature (a discretisation scheme, a model, a stabilisation strategy, etc.) is added or modified in the primal, its discrete adjoint counterpart must also be coded or modified accordingly, lest lose consistency between adjoint and primal solver. Furthermore, hand-derivation requires by definition full knowledge of the primal operators to be differentiated; while this does not pose a problem in the context of this work, it does in general when one has no access to the primal source code or detailed documentation.

When the source code is accessible, the *reverse accumulation* strategy (ED3 in Section 2.3.3) is a powerful alternative to hand-derivation which allows to obtain a consistent discrete adjoint solution in an automated way, namely by applying reverse-mode Automatic Differentiation (AD) to the procedures responsible for residual computation and exploit the results as shown in Section 2.3.3 - specifically by using the AD routine to compute the adjoint residual, and use this in a FPI solution scheme. ED3 never explicitly assembles/stores the full Jacobian: this entails a considerable advantage in terms of memory consumption - provided that the AD process is optimised in that sense - but it precludes the implementation of adjoint solution algorithms which require knowledge of the full Jacobian (such as those that will be discussed in Sections 6.2.2 and 6.2.3) as well as the "black-box" approach of feeding the entire adjoint system - Jacobian and right-hand side - to a third-party linear solver.

Since the investigation of such solution strategies is among the objectives of the present work, the *reverse assembly* technique (ED2) is hereby chosen as the preferred approach. The goal of ED2 is to assemble the full Jacobian and adjoint right-hand side in an automated way; the automation may be done either via Finite Differences (FD) or AD. Considering the generic primal $\mathbf{r}\left(\mathbf{w}\left(\boldsymbol{\alpha}\right),\boldsymbol{\alpha}\right)=\mathbf{0}$, the FD strategy for assembling the Jacobian consists in introducing in the converged primal field a perturbation, separately for each degree of freedom, and computing the corresponding residual, i.e.

$$\mathbf{r}_k=\mathbf{r}\left(\left(\mathbf{w}_0+\boldsymbol{\delta}\mathbf{w}_k\right),\boldsymbol{\alpha}\right) \tag{6.9}$$

where $\mathbf{w}_0$ is the converged solution and $\boldsymbol{\delta}\mathbf{w}_k$ is the perturbation for the k-th degree if freedom, i.e. a zero vector everywhere except for $\left(\boldsymbol{\delta}\mathbf{w}_k\right)_k=\delta w_k$ (with δw_k a suitable FD step-length, further discussed in Section 6.1.5). Then, if e.g. a first-order forward FD formula is used, the Jacobian is assembled by computing

$$\frac{\partial\mathbf{r}}{\partial w_k}=\frac{\mathbf{r}_k-\mathbf{r}_0}{\delta w_k},\quad k=1\cdots n_w \tag{6.10}$$

where $\mathbf{r}_0$ is the residual evaluated at $\mathbf{w}_0$ (therefore close to zero up to the specified tolerance for the primal solution), and $\frac{\partial\mathbf{r}}{\partial w_k}$ gives the k-th Jacobian column. An AD-

based assembly is similar: given the routine `RESIDUAL`($\rightarrow\mathbf{w}$,$\rightarrow\boldsymbol{\alpha}$,$\leftarrow\mathbf{r}$), its forward-mode AD counterpart `DRESIDUAL`($\rightarrow\mathbf{w}$,$\rightarrow\mathbf{dw}$,$\rightarrow\boldsymbol{\alpha}$,$\rightarrow\mathbf{d}\boldsymbol{\alpha}$,$\leftarrow\mathbf{r}$,$\leftarrow\mathbf{dr}$) can be used to compute $\frac{\partial \mathbf{r}}{\partial w_k}$ (returned in $\mathbf{dr}$) by setting all seeds to zero except for $(\mathbf{dw})_k = 1$. An analogous process (FD or AD-based) applied to the cost function computation allows to assemble the adjoint right-hand side. It will be shown in Section 6.1.3 how the sparsity pattern of the Jacobian can be exploited to make the cost of reverse assembly independent of the number of degrees of freedom.

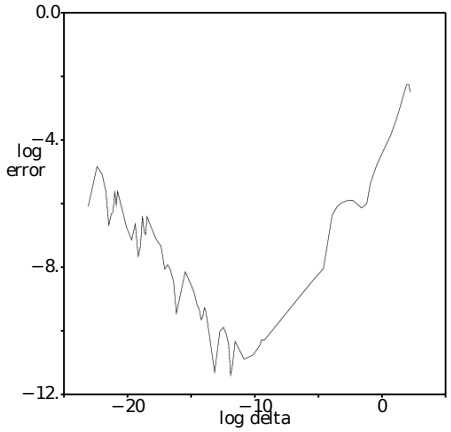


Figure 6.1: Finite Differences error dependence on delta (step-length) [63].

The AD approach has the considerable advantage of being able to produce the exact derivative values regardless of the nature of the primal problem. Conversely, a FD-computed derivative will always contain some form of error unless the primal is linear - which is not the case for the Navier-Stokes problem. The magnitude of the error depends on the choice of the FD step-length: too large a step-length will result in excessive truncation error, while too small a step-length will incur round-off error (Figure 6.1).

On the other hand, a FD-based assembly makes for a far less intrusive procedure. It will be shown in Section 6.1.4 how the approach does not necessarily require access to the primal source code: it is sufficient for the user to be able to provide the solver with a perturbed solution field and read back the corresponding residual vector, which is typically feasible via so-called "user subroutines" in many commercial CFD solvers. This is the basic principle - and the main selling point - of ESI's *i-Adjoint library* [193], which is the tool used in this thesis to reverse-assemble the MHFV adjoint. The "i" in i-Adjoint stands for *independent*, highlighting the fact that the tool may be considered as an external plug-in compatible with virtually any CFD solver capable of performing a user-requested residual computation.

6.1.3 Graph colouring

The reverse assembly strategy, both in its FD and AD version, suffers from one obvious drawback: if n_w is the number of primal degrees of freedom, then a naïve implementation would incur a CPU cost roughly equivalent to n_w matrix-vector products (one per residual computation), where the matrices are $n_w \times n_w$. This cost quickly becomes unsustainable for a realistic industrial application. However, in the context of CFD, the primal is always a system produced by some form of discretisation scheme (FV, FE, MHFV, etc.), implying that its matrix is always *sparse*: this sparsity pattern makes reverse assembly feasible in practice, thanks to *graph colouring* [146].

Graph colouring (more specifically, *vertex colouring*) is the assignment of labels to each vertex of an incidence graph such that no edge connects two identically labelled vertices. These labels are traditionally referred to as *colours*, as the underlying theory stems from the following observation known as the *Four-Colour Theorem* [9]: given any separation of a plane into contiguous regions (interpretable as the graphic representation of a *planar graph*: a graph that can be embedded in a plane, i.e. that can be drawn in a plane without any edges crossing [114]), no more than four colours are required to colour each region so that no two adjacent regions (i.e. sharing an edge) are of the same colour. Graph colouring has practical applications where several tasks are to be performed, and one may benefit from knowing which ones are in conflict or depend on each other. In case of no conflict or dependency, then two tasks are of the same colour and can be executed at the same time, which suggests e.g. a useful application in *scheduling problems* related to parallel computation [160].

The exploitation of the sparsity pattern for an efficient FD-based Jacobian evaluation was first proposed by Curtis, Powell and Reid [62], and modelled as a colouring problem by Coleman and Moré [58]. Since then, several authors have extended the concept to the AD framework [10, 33, 92, 93, 108]. When the objective is to assemble the Jacobian, the incidence graph to be coloured is determined by the location of non-zeroes in the primal system: vertices are the problem's degrees of freedom, and an edge between two degrees of freedom exists if the residual related to the first depends on the value of the second. The concept is clarified via a simple example. Considering as primal the following system of four variables:

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & 0 & 0 \\ 0 & \alpha_{22} & \alpha_{23} & 0 \\ 0 & 0 & \alpha_{33} & \alpha_{34} \\ 0 & 0 & \alpha_{43} & \alpha_{44} \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} \tag{6.11}$$

where for simplicity the α_{ij} coefficients are assumed constant, the sparsity pattern of the system allows to group all degrees of freedom in two colours: colour A, containing w_1 and w_3, and colour B, containing w_2 and w_4. These are graphically represented in Figure 6.2 in red and blue, respectively. One can verify how variables of the same colour indeed target (influence) disjoint subsets of the residual components: for colour A, w_1 targets r_1 while w_3 targets r_2, r_3 and r_4; for colour B, w_2 targets r_1 and r_2 while w_4 targets r_3 and r_4.

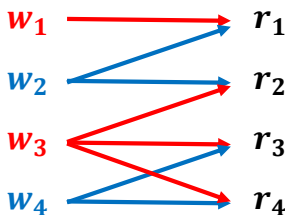


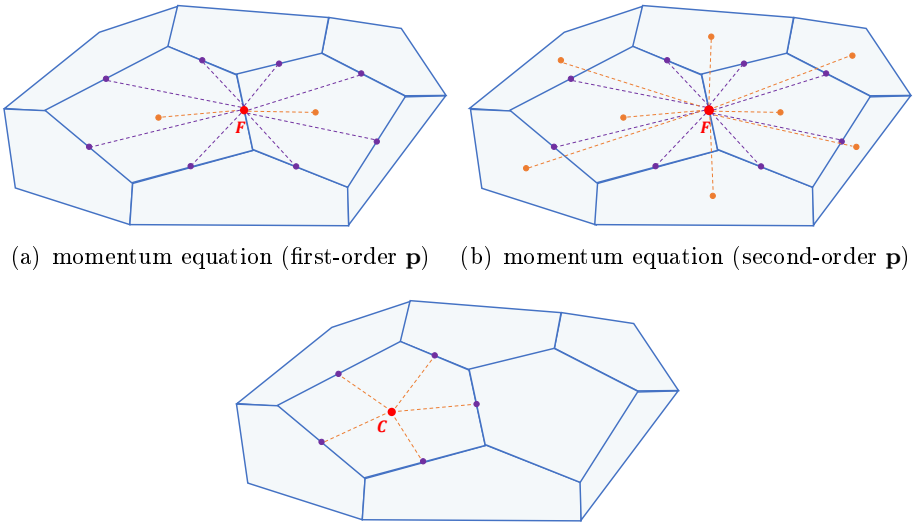
Figure 6.2: Coloured graph for Jacobian assembly of (6.11).

Once the colours are identified, reverse assembly of the Jacobian can be done by evaluating one residual vector per colour. For instance, assuming a FD-based strategy is employed, a perturbed residual $\mathbf{r}_A$ for colour A is obtained by adding a perturbation δw to all A-coloured variables simultaneously. The FD derivative evaluation then yields

$$\mathbf{dr}_A = \frac{\mathbf{r}_A - \mathbf{r}_0}{\delta w} \ , \tag{6.12}$$

and restricting $\mathbf{dr}_A$ to the subset of residual components targeted by w_1 and w_3 gives the first and third Jacobian column, respectively. In practice, this restriction is done by filtering $\mathbf{dr}_A$ through a mask $\boldsymbol{\delta}_i$ for each A-coloured variable i, where $(\boldsymbol{\delta}_i)_j = 1$ if variable i targets residual j, and zero otherwise. The same procedure applied to B-coloured degrees of freedom allows to compute the remaining columns, hence the full Jacobian can be assembled with two residual evaluations only. For an AD-based assembly the process is analogous: the (forward-mode) differentiated code is seeded for all same-coloured variables at the same time (e.g., for colour A: $\mathbf{dw} = \texttt{\{1,0,1,0\}}$), then a residual computation per colour is launched and the resulting differentiated residual $\mathbf{dr}$ is masked as described above.

The sparsity pattern of the graph of a discrete problem depends on the stencil of each equation in the system, which defines for each residual component the set of degrees of freedom it depends on. The stencils for the MHFV Navier-Stokes are shown in Figure 6.3, where each face holds d degrees of freedom - the d hybrid velocity components (5.3) - and each cell holds one - the cell-averaged pressure (5.4). If two primal degrees of freedom do not share a target, i.e. if they never appear together in the same stencil, then they can be of the same colour, meaning that they influence two completely disjoint subsets of the



(a) momentum equation (first-order **p**) (b) momentum equation (second-order **p**)

(c) continuity equation

Figure 6.3: MHFV stencils for the full Oseen system, determining the Navier-Stokes incidence graph.

residual vector $\mathbf{r}$. As a consequence, all Jacobian entries related to same-coloured degrees of freedom can be computed at the same time, namely by computing the residual after introducing a FD perturbation - or AD seeding - for all degrees of freedom of a certain colour at the same time. This reduces the number of required residual computations to the number of colours, which only depends on the sparsity pattern of the primal - and thus on the discretisation scheme - and is independent of the problem size.

Graph colouring can also be used to reverse-assemble the adjoint right-hand side $\left(\frac{\partial J}{\partial \mathbf{w}}\right)^T$, i.e. the vector of partial derivatives of the objective function J with respect to each primal variable. This might seem unfeasible at first: J is a scalar-valued function which depends simultaneously on all degrees of freedom involved in its computation, and therefore a FD or AD reconstruction would seem to require as many evaluations of J as the number of variables involved. However, in real CFD applications, J is typically a *discrete integral quantity*, meaning that it can be expressed as the summation of n_{cst} "local cost functions" evaluated at n_{cst} locations, or *probes*:

$$J = \sum_{i=1}^{n_{cst}} j_i \ . \tag{6.13}$$

For example, if the cost function is the total pressure drop, then it is computed as the sum of the total pressure evaluated locally on each boundary face. Then the partial

derivative of J with respect to a given variable w_k is also a summation over probes:

$$\frac{\partial J}{\partial w_k} = \sum_{i=1}^{n_{cst}} \frac{\partial j_i}{\partial w_k} \ . \tag{6.14}$$

If a CFD-like sparsity pattern is assumed, then $\frac{\partial j_i}{\partial w_k}$ will be zero in most cases except for those w_k which directly influence (target) the value of j_i at probe i (for example, total pressure loss evaluated at a certain boundary face will only be targeted by velocity and pressure values on the local stencil of that face). Therefore a sparse graph relating primal degrees of freedom to a cost function vector $\mathbf{j}$ (with $(\mathbf{j})_i = j_i$) can be created, coloured and used to compute several $\frac{\partial j_i}{\partial w_k}$ entries simultaneously; ultimately, the cost associated with the full assembly of the adjoint right-hand side will depend on the number of colours in this graph.

A simple example is provided in order to clarify. A generic primal is considered featuring four degrees of freedom: w_1, w_2, w_3, w_4. The cost function J is defined as a discrete integral form

$$J = \gamma_1 w_1 w_3 + \gamma_2 w_1 w_4 + \gamma_3 w_2 w_4 \tag{6.15}$$

with γ_i coefficients assumed constant. J may be written as a summation over three probes $\sum_{i=1}^{3} (\mathbf{j})_i$, where

$$\mathbf{j} = \begin{pmatrix} j_1 \\ j_2 \\ j_3 \end{pmatrix} = \begin{pmatrix} \gamma_1 w_1 w_3 \\ \gamma_2 w_1 w_4 \\ \gamma_3 w_2 w_4 \end{pmatrix} \ ; \tag{6.16}$$

therefore a **w**-to-**j** graph can be created and coloured according to which state variables $\mathbf{w}$ target which components of $\mathbf{j}$. Two colours suffice: colour A for w_1 and w_2, colour B for w_3 and w_4 (red and blue, respectively, in Figure 6.4). For a FD-based assembly,

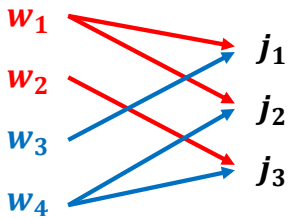


Figure 6.4: Coloured graph for adjoint right-hand side assembly for (6.16).

a perturbation δw is introduced for all variables of the same colour simultaneously. For colour A, this produces the new probe-wise cost function $\mathbf{j}_A$ from which the FD derivative value is computed:

$$\mathbf{dj}_A = \frac{\mathbf{j}_A - \mathbf{j}_0}{\delta w} \tag{6.17}$$

where $\mathbf{j}_0$ is the original probe-wise cost function. Then, filtering through masks corresponding to each A-coloured variable gives

$$\frac{\partial \mathbf{j}}{\partial w_1} = (\mathbf{dj}_A)(\boldsymbol{\delta}_1) \quad \text{and} \quad \frac{\partial \mathbf{j}}{\partial w_2} = (\mathbf{dj}_A)(\boldsymbol{\delta}_2) \ . \tag{6.18}$$

Finally, summation over probes yields the adjoint right-hand side entries corresponding to each A-coloured degree of freedom:

$$\frac{\partial J}{\partial w_1} = \sum_{i=1}^{3} \left(\frac{\partial \mathbf{j}}{\partial w_1} \right)_i \quad \text{and} \quad \frac{\partial J}{\partial w_2} = \sum_{i=1}^{3} \left(\frac{\partial \mathbf{j}}{\partial w_2} \right)_i \ . \tag{6.19}$$

Performing the same operations for colour B allows to assemble the remaining entries of the adjoint right-hand side, hence evaluating the full vector requires as many cost function evaluations as the number of colours (notice that here the expression "cost function evaluation" does not imply a primal solve, but rather the sequence of operations that, starting from a given primal solution, leads to a probe-wise objective function value). An analogous procedure can be implemented by using AD rather than FD; in this case, however, reverse-mode AD is advantageous: a single call to the reverse-differentiated procedure computing the cost function produces the full adjoint right-hand side, hence the subdivision of J into probe-wise values and subsequent colouring is superfluous.

For any given graph there exists an optimal colouring scheme, i.e. one that uses the lowest possible number of colours (known as *chromatic number* of the graph [146]), but finding the optimal colouring scheme is known to be a *NP-hard* problem [92]: there are currently no known polynomial-time solutions, hence optimal colouring is practically unfeasible for all but smaller graphs. Therefore a number of heuristic colouring algorithms have been proposed with the purpose of performing a CPU-efficient graph colouring leading to an acceptable - although sub-optimal - colouring scheme. The colouring algorithms available in i-Adjoint are:

- *Welsh-Powell* [237]: graph vertices are ordered according to their *valence*, i.e. the number of edges associated to the vertex, from highest to lowest. The first vertex is assigned a colour, then the list is descended and all vertices not connected to the already coloured ones are assigned the same colour. The process is then repeated with a new colour, always starting with the uncoloured vertex with the highest valence, until all vertices have been coloured. The idea is that, by treating the highest valence vertices first, vertices with the largest number of conflicts are taken care of as early as possible.

- *DSATUR* [44]: the algorithm takes into account the *degree of saturation* of a vertex,

i.e. the number of different colours already assigned to its neighbours. At each iteration, the uncoloured vertex with the highest degree of saturation is identified and coloured with the smallest possible colour.

- *MDSATUR* [6]: a non-trivial modified version of DSATUR, it considers not only the degree of saturation but also the vicinity of each candidate vertex to the vertex which was coloured last.
- *Planar-6* (also known as *6-COLOR*) [162]: similarly to Welsh-Powell it is based on vertex valence, but its vertex ranking algorithm is designed so that the valence of neighbouring vertices is also taken into account. As the name suggests, it is capable of colouring a planar graph with at most six colours.
- *Modified Planar-6*: analogous to Planar-6, with the same additional heuristics implemented in MDSATUR.

6.1.4 Reverse assembly with i-Adjoint

The i-Adjoint tool is coded within the *FORTRAN Template Library* (FTL) framework [193], a non-standard extension of the FORTRAN language which allows for object-oriented coding with the additional possibility of defining classes with template arguments. Among the classes provided by FTL, the main ones used by the top-level i-Adjoint routines are: `MeasureSpace`, a descriptor of a discrete space (i.e. the number of probes and the number of degrees of freedom attached to each probe); `MeasureField`, an array holding values of a field belonging to a certain space; `Topology`, a descriptor of mesh connectivity; `Graph`, a sparse incidence graph in Compressed Column Storage (CCS) form; `Matrix`, a block-sparse matrix object also stored in CCS.

As an example, the scheme below outlines how an i-Adjoint driver code (right) interacts with a generic primal solver (left) in order to reverse-assemble the Jacobian matrix.

Primal		**i-Adjoint**
solve $\mathbf{r}(\mathbf{w}) = \mathbf{0}$		
	$\xrightarrow{\mathbf{w}_0,\, \mathbf{r}_0}$	read $\mathbf{w}_0$, $\mathbf{r}_0$ (field and residual) as `MeasureFields`
	$\xrightarrow{\text{connectivity}}$	read mesh connectivity
		assemble $\mathbf{T}$ (`Topology`) from connectivity
		assemble $\mathbf{G}$ (`Graph`) from $\mathbf{T}$ & user-defined stencil size
		colour $\mathbf{G}$
		reverse-assemble Jacobian $\mathbf{A}$ (`Matrix`):
		do `a = 1, NumColours`($\mathbf{G}$)

		set $\mathbf{w}_a = \mathbf{w}_0 + \boldsymbol{\delta}\mathbf{w}_a$ (perturbation for colour `a`)
	$\xleftarrow{\mathbf{w}_a}$	write $\mathbf{w}_a$ to primal
compute $\mathbf{r}_a = \mathbf{r}(\mathbf{w}_a)$		
	$\xrightarrow{\mathbf{r}_a}$	read $\mathbf{r}_a$
		evaluate FD: $\mathbf{dr}_a = (\mathbf{r}_a - \mathbf{r}_0)/\boldsymbol{\delta}\mathbf{w}_a$
		store columns of $\mathbf{A}$ for colour `a`
		end do

A few observations are in order:

- As anticipated in Section 6.1.2, no direct access is required to the primal source code as long as the primal solver provides interfaces for reading/writing operations and user-requested residual computations.

- Once the mesh connectivity is read from the primal, the assembly and colouring of the `Graph` object are carried out entirely on the i-Adjoint side. The assembly however does require some user input: the extent of local stencils on which primal operators act cannot usually be read directly from the primal without accessing the code, and thus must be specified. In practice, an educated guess is sufficient to build the graph correctly. For example, if the primal is known to use a collocated, second-order accurate FV scheme, then the graph is built based on a second-neighbourhood cell-to-cell relationship deducible from the topology. In case of doubt the user may choose to provide an "over-graph", i.e. a graph that is known with certainty to contain all primal connectivities and possibly some additional ones, at the cost of working with more colours than necessary.

- An analogous procedure can be implemented to compute the adjoint right-hand side $\frac{\partial J}{\partial \mathbf{w}}$. However, as explained in Section 6.1.3, the cost function needs to be read in a probe-wise form $\mathbf{j}$ in order to be able to apply a colouring scheme. This is typically not an option in a standard CFD solver, where J is usually returned as a single scalar value. In this case a routine computing the vector $\mathbf{j}(\mathbf{w})$ must be implemented within the i-Adjoint driver code itself.

6.1.5 Considerations on FD-based assembly

The non-intrusive nature of a FD-based reverse assembly has so far been mentioned as its main advantage over AD. This is however irrelevant in the context of this thesis, where full access to the MHFV code is granted. The FD approach was chosen for a different practical reason: the MHFV solver, being written in the non-standard FTL language, contains

several code statements (e.g. declaration and usage of template arguments in FORTRAN modules) which are not recognised by AD tools, unless substantial modifications to the FTL makefile are made. A preliminary attempt with the operator-overloading tool DCO[1] proved this to be a non-trivial, time-consuming task. Conversely, interfacing with i-Adjoint was facilitated by the fact that the tool also uses FTL-native objects.

The downside of incurring a source of error due to the FD approach cannot in general be avoided completely. However, i-Adjoint is equipped with a mechanism designed to alleviate the problem by estimating the location of the "sweet spot" minimising both truncation and round-off error (Figure 6.1): starting with an initial step-length $\delta w_k^{(1)}$, FD evaluations are carried out iteratively with a progressively smaller step-length computed at the n-th iteration as

$$\delta w_k^{(n)} = \frac{\delta w_k^{(n-1)}}{t} \tag{6.20}$$

where $t > 1$ is a user-defined step divider ($t = 2$ is empirically found to be a suitable choice). Convergence is then monitored component-wise based on a relative *Cauchy criterion*:

$$\varepsilon^{(n)} = \left| \frac{FD^{(n)} - FD^{(n-1)}}{FD^{(1)}} \right| \tag{6.21}$$

where $FD^{(n)}$ is the computed derivative value at the n-th iteration. The algorithm is stopped when $\varepsilon^{(n)}$ falls below a specified tolerance, as this indicates that the linear (or polynomial for higher-order FD) approximation induced by the FD expression represents with sufficient accuracy the function being differentiated within the range delimited by the current step-length $\delta w_k^{(n)}$.

This approach would require the user to specify a suitable initial step-length as an absolute value, which is not a trivial task since it is typically problem-specific. A better choice is to have the user specify a relative coefficient λ instead, and use this to generate a scaled initial step-length of the form

$$\delta w_k^{(1)} = \lambda w_{k,s} \tag{6.22}$$

where $w_{k,s}$ is a factor introduced to normalise the step-length on the variable $\mathbf{w}$. It will be shown throughout this chapter how, in the context of MHFV incompressible Navier-Stokes, all required FD operations can be grouped in two categories depending on the nature of the variable with respect to which one wishes to differentiate: discrete flow variables and nodal coordinates. For the former, the FD step-length can be scaled by the magnitude of the flow variable itself. More specifically, the initial step-length for hybrid

[1] https://www.stce.rwth-aachen.de/research/software/dco-fortran

velocity components is defined as

$$\delta u_F^i = \lambda \left\| \vec{u}_F \right\| \tag{6.23}$$

where $\left\| \vec{u}_F \right\| = \sqrt{\sum_{i=1}^{d} \left(u_F^i \right)^2}$ is the magnitude of the hybrid velocity at face F. In case of zero or near-zero magnitude, $\left\| \vec{u}_F \right\|$ is replaced with a fraction of the velocity magnitude averaged over the entire field. Concerning the cell-based pressure field $\mathbf{p}$, since for incompressible Navier-Stokes it is defined up to a reference value p_{ref}, a more suitable choice for the initial step-length is

$$\delta p_C = \lambda \left(\max \left(\mathbf{p} \right) - \min \left(\mathbf{p} \right) \right) \ , \tag{6.24}$$

i.e. a uniform step-length scaled by the difference between the maximum and minimum pressure values in the flow field. Scaling is however less relevant for pressure than for velocity, at least for the MHFV Navier-Stokes scheme: the pressure gradient operator is linear, therefore any FD step-length will yield the exact derivative value as long as it is large enough to avoid round-off error.

In the case of nodal coordinates, on the other hand, it would not make sense to scale by a quantity representative of the coordinate value itself: given that they are involved in the differentiation of local distances and inertial quantities, it is much more appropriate to define the scaling factor such that the resulting FD step-length for node N corresponds to a fraction of a value h_N representing a local discretisation length scale:

$$\delta x_N^i = \lambda h_N \tag{6.25}$$

where x_N^i denotes the i-th coordinate of node N. A suitable choice for h_N is the shortest edge length formed with a neighbouring node:

$$h_N = \min \left(\left\| \vec{x}_{N'} - \vec{x}_N \right\| \quad \forall N' \text{ sharing an edge with } N \right) \ , \tag{6.26}$$

which avoids the generation of too large an initial step-length over shorter edges in the case of a highly anisotropic mesh.

6.1.6 Reduced reverse assembly

The cost of a colour-based reverse assembly of the Jacobian depends not only on the matrix sparsity graph but also on the number of degrees of freedom attached to each vertex of the graph. In particular, the MHFV Navier-Stokes scheme (5.32) presents a heterogeneous form: vertices can either be faces, carrying d degrees of freedom each

(hybrid velocity components (5.3)), or cells, carrying one degree of freedom each (cell-averaged pressure (5.4)). Therefore, if colours are assigned based on stencils shown in Figure 6.3, reverse assembly will ultimately require as many residual evaluations per colour as the maximum number of degrees of freedom attached to any vertex of that colour - which will typically be d, unless the colouring algorithm happens to produce some colours containing cell-averaged pressure values only (which is rather unlikely, but in this case the reverse assembly for those colours will require one residual evaluation only).

The present section presents a possible way of reducing this cost - a *reduced reverse assembly* option - which takes full advantage of primal code access and knowledge of the MHFV Navier-Stokes operator. As mentioned in Section 5.1.1, the convecting flux $\mathbf{U}$ is de facto considered as a separate variable, meaning that it can be explicitly added to the Picard-linearised Navier-Stokes:

$$\begin{bmatrix} \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} & \boldsymbol{\mathcal{G}} & 0 \\ \boldsymbol{\mathcal{D}} & 0 & 0 \\ -\boldsymbol{\mathcal{C}} & 0 & \mathcal{I} \end{bmatrix} \begin{pmatrix} \widetilde{\mathbf{u}} \\ \mathbf{p} \\ \mathbf{U} \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{g}} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \tag{6.27}$$

where $\boldsymbol{\mathcal{C}}$ is the convecting flux operator defined in (5.9) and $\mathcal{I}$ is the identity matrix. Under the assumption that, aside from the dependency of the convection-diffusion operator $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ on $\mathbf{U}$, no other non-linearities are present (which holds as long as no solution-dependent stabilisation schemes are used), all variables in (6.27) are linear with respect to each other, and the Navier-Stokes Jacobian can be expressed as

$$\mathcal{A} = \begin{bmatrix} \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} & \boldsymbol{\mathcal{G}} & \boldsymbol{\mathcal{T}} \\ \boldsymbol{\mathcal{D}} & 0 & 0 \\ -\boldsymbol{\mathcal{C}} & 0 & \mathcal{I} \end{bmatrix} \tag{6.28}$$

where $\boldsymbol{\mathcal{T}} = \frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{U}}$ defines the partial derivative of the momentum residual with respect to the convecting flux. The corresponding adjoint system then becomes

$$\begin{bmatrix} \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} & \boldsymbol{\mathcal{G}} & \boldsymbol{\mathcal{T}} \\ \boldsymbol{\mathcal{D}} & 0 & 0 \\ -\boldsymbol{\mathcal{C}} & 0 & \mathcal{I} \end{bmatrix}^T \begin{pmatrix} \widetilde{\mathbf{u}}^* \\ \mathbf{p}^* \\ \mathbf{U}^* \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{g}'_u}^* \\ \mathbf{g}_p^* \\ \mathbf{g}_U^* \end{pmatrix} \tag{6.29}$$

where $\mathbf{U}^*$ is an *adjoint convecting flux* and $\mathbf{g}_U^* = \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T$. Notice that $\widetilde{\mathbf{g}'_u}^*$ in (6.29) does not correspond to $\widetilde{\mathbf{g}_u}^*$ in (6.7), as the former takes into account only direct dependencies of the cost function J on $\widetilde{\mathbf{u}}$ and not those coming from $\mathbf{U}$, which are now expressed in $\mathbf{g}_U^*$ instead. The third equation in (6.29) provides an explicit definition of the adjoint

convecting flux as a function of the adjoint hybrid velocity:

$$\mathbf{U}^* = -\boldsymbol{\mathcal{T}}^T \widetilde{\mathbf{u}}^* + \mathbf{g}_U^* \ . \tag{6.30}$$

It can therefore be eliminated in the first equation and, noticing that

$$\boldsymbol{\mathcal{C}}^T \mathbf{g}_U^* + \widetilde{\mathbf{g}'_u}^* = \left(\frac{\partial \mathbf{U}}{\partial \widetilde{\mathbf{u}}}\right)^T \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T + \left(\frac{\partial J}{\partial \widetilde{\mathbf{u}}}\right)^T = \widetilde{\mathbf{g}_u}^* \ , \tag{6.31}$$

the adjoint system is retrieved in the form

$$\begin{bmatrix} \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} + \boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{C}} & \boldsymbol{\mathcal{G}} \\ \boldsymbol{\mathcal{D}} & 0 \end{bmatrix}^T \begin{pmatrix} \widetilde{\mathbf{u}}^* \\ \mathbf{p}^* \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{g}_u}^* \\ \mathbf{g}_p^* \end{pmatrix} \tag{6.32}$$

where $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}} + \boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{C}}$ corresponds to $\overline{\overline{\boldsymbol{\mathcal{F}}}}$ in (6.8) and thus, more specifically, $(\boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{C}})^T$ is the discrete ATC operator mentioned in Section 6.1.1.

Formulation (6.32) suggests a fast way of reverse-assembling the Jacobian matrix: operators $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$, $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{D}}$ are readily available in the primal code; to obtain the full Jacobian one needs only evaluate $\boldsymbol{\mathcal{T}}$. Reverse assembly of $\boldsymbol{\mathcal{T}}$ requires only one residual evaluation per colour - because $\mathbf{U}$ carries only one degree of freedom per face, as opposed to $\widetilde{\mathbf{u}}$ which carries d (one for each velocity component). As a consequence, the cost of the Jacobian assembly is reduced roughly by factor d - as confirmed by test results in Section 6.4.2.

The reduced assembly option presented here can be seen as a suitable compromise between a hand-derived and an automated Jacobian computation: anything linear is "hand-derived" in the sense that matrix blocks are recycled verbatim from the primal operator, while non-linearities, which are more error-prone in hand-derivation, are dealt with separately by an automated procedure only involving those degrees of freedom which directly cause such non-linearities. As mentioned above, for MHFV Navier-Stokes such a separation is only feasible provided that the convection-diffusion operator $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ does not depend on the hybrid velocity components $\widetilde{\mathbf{u}}$ other than through the convecting flux $\mathbf{U}$. Therefore, a reduced assembly cannot be done in combination with solution-dependent convective schemes such as flux limiters (Section 4.2.3) or WLSQR (Section 4.2.4).

A further advantage of this approach is that, in certain cases, the issue of having to choose an appropriate step-length for a FD-based assembly is circumvented: under the assumption that the hybrid momentum residual $\widetilde{\mathbf{r}_u}$ is linear with respect to $\mathbf{U}$, then $\boldsymbol{\mathcal{T}}$ is constant for a converged primal field (i.e. it does not depend on $\mathbf{U}$) and a FD-based evaluation will return the exact $\boldsymbol{\mathcal{T}}$ regardless of the chosen step-length. This condition

is verified by any MHFV convective scheme which is linear with respect to $\mathbf{U}$: centred schemes (both MIXC and HYBC, Section 4.1.1) and HUPW1 (Section 4.1.2).

6.2 Solution algorithms for adjoint Navier-Stokes

The adjoint MHFV Navier-Stokes problem (6.8) is a linear system that closely resembles its non-linear primal counterpart, being an Oseen-type saddle-point discrete problem. The main difference is the additional cross-coupling in the adjoint momentum equations due to the ATC term - the extra-diagonal blocks in $\overline{\boldsymbol{\mathcal{F}}}^T$. To solve the adjoint Navier-Stokes system, a Block-Coupled (BCPL) approach can be expected to be the optimal choice in terms of performance by analogy with the results shown for the primal in Section 5.3.3. The fully coupled system however is known to pose significant computational challenges for standard linear solvers: the presence of the ATC negatively affects the diagonal dominance of the Jacobian, causes numerical stiffness and can lead to an ill-conditioned matrix [138]. The development of robust solution strategies using the full Jacobian in such conditions is in its early stages: for discrete adjoints, Osusky et al. [180] and Xu and Timme [244] successfully investigated a family of ILU-preconditioned Krilov solvers for fully coupled adjoint RANS in complex flow conditions (including differentiated turbulence models); similarly, for continuous adjoints, Vezyris et al. [233] present promising results for steady and unsteady cases with a pseudo-compressibility BCPL approach using a preconditioned GMRES solver.

The topic of linear solvers is beyond the scope of this thesis. However, the fact that primal and adjoint problems are similar in form implies that the segregated algorithms devised for the primal MHFV Navier-Stokes (Section 5.2) can be recycled and adjusted to facilitate the solution of the adjoint system as well, as discussed in the following sections.

6.2.1 Adjoint SIMPLEC

Adjoint SIMPLE-like iterative strategies have been implemented for FV schemes either by "brute-force" differentiating the primal algorithm [137, 175, 222] or by recycling and adapting the original preconditioning scheme [4, 218]. Following the latter approach, the SIMPLEC algorithm from Section 5.2.1 can be easily adapted to the adjoint system (Algorithm 4). As mentioned, the adjoint momentum operator $\overline{\boldsymbol{\mathcal{F}}}^T$ is not block-diagonal; since the ability to solve separately for each velocity component is one of the most attractive features of SIMPLE-like algorithms, it is desirable to maintain this de-coupled nature in the adjoint version. A suitable block-diagonal approximation to $\overline{\boldsymbol{\mathcal{F}}}^T$ is required to be

used as system matrix for the velocity prediction step, whilst treating all cross-coupled terms explicitly. An intuitively good choice is the transpose of the convection-diffusion operator $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$ itself, assembled using the converged convecting flux values $\mathbf{U}$.

Algorithm 4 MHFV Adjoint SIMPLEC

$n = 0$
Initialise $\widetilde{\mathbf{u}}^{*,0}$, $\mathbf{p}^{*,0}$
while not converged **do**
  Solve relaxed adjoint hybrid momentum equation (*adjoint velocity prediction*):
  $\boldsymbol{\mathcal{F}}^{\alpha}_{\nu,\vec{U}}{}^{T}\widetilde{\mathbf{u}}^{*,n+1/2} = \widetilde{\mathbf{g}_u}^{*} - \boldsymbol{\mathcal{D}}^{T}\mathbf{p}^{*,n} - \left(\overline{\boldsymbol{\mathcal{F}}}^{T} - \boldsymbol{\mathcal{F}}^{\alpha}_{\nu,\vec{U}}{}^{T}\right)\widetilde{\mathbf{u}}^{*,n}$;
  Solve Schur complement pseudo-Laplacian (*adjoint pressure correction*):
  $\widehat{\mathcal{S}}^{\alpha}\boldsymbol{\delta}\mathbf{p}^{*} = \boldsymbol{\mathcal{G}}^{T}\widetilde{\mathbf{u}}^{*,n+1/2} - \mathbf{g}_p^{*}$;
  Update adjoint pressure:
  $\mathbf{p}^{*,n+1} = \mathbf{p}^{*,n} + \boldsymbol{\delta}\mathbf{p}^{*}$;
  Update adjoint hybrid velocity:
  $\widetilde{\mathbf{u}}^{*,n+1} = \widetilde{\mathbf{u}}^{*,n+1/2} - \left(\mathrm{diag}\left(\frac{1}{\alpha\beta_F}\right)\right)\boldsymbol{\mathcal{D}}^{T}\boldsymbol{\delta}\mathbf{p}^{*}$;
  n=n+1
end while
return $\widetilde{\mathbf{u}}^{*}$, $\mathbf{p}^{*}$

As done for the primal, inertial relaxation is applied in the form (5.54) to the adjoint momentum equation, which is necessary for steady-state SIMPLEC. The scaling factor β_F is recycled from the primal (5.53), while the relaxation factor α is set independently. Thus the adjoint velocity prediction step at iteration n requires solving:

$$\begin{aligned}\boldsymbol{\mathcal{F}}^{\alpha}_{\nu,\vec{U}}{}^{T}\widetilde{\mathbf{u}}^{*,n+1/2} &= \widetilde{\mathbf{g}_u}^{*} - \boldsymbol{\mathcal{D}}^{T}\mathbf{p}^{*,n} - \left(\overline{\boldsymbol{\mathcal{F}}}^{T} - \boldsymbol{\mathcal{F}}^{\alpha}_{\nu,\vec{U}}{}^{T}\right)\widetilde{\mathbf{u}}^{*,n} \\ &= \widetilde{\mathbf{g}_u}^{*} - \boldsymbol{\mathcal{D}}^{T}\mathbf{p}^{*,n} + \alpha\left(\mathrm{diag}\left(\beta_F\right)\right)\widetilde{\mathbf{u}}^{*,n} - \left(\overline{\boldsymbol{\mathcal{F}}}^{T} - \boldsymbol{\mathcal{F}}^{T}_{\nu,\vec{U}}\right)\widetilde{\mathbf{u}}^{*,n}\end{aligned} \tag{6.33}$$

where the quantity $\left(\overline{\boldsymbol{\mathcal{F}}}^{T} - \boldsymbol{\mathcal{F}}^{T}_{\nu,\vec{U}}\right)\widetilde{\mathbf{u}}^{*,n}$ on the right-hand side is the contribution due to the cross-coupling of adjoint velocity components (ATC) that is treated explicitly in order to allow to solve separately for each component.

For the adjoint pressure correction step, as for the primal, $\boldsymbol{\mathcal{G}}^{T}$ is replaced with $\boldsymbol{\mathcal{D}}$ regardless of the specific pressure scheme in order not to deteriorate the sparsity of the Schur complement. As a consequence, the approximated Schur complement $\widehat{\mathcal{S}}^{\alpha}$ is the transpose of the one used for the primal (5.55). Since the latter is self-adjoint, the two are identical.

6.2.2 Adjoint Velocity-Coupled

The adjoint version of SIMPLEC solves for each component of $\widetilde{\mathbf{u}}^*$ in a segregated fashion, moving the ATC to the right-hand side. The ATC is known to be a troublesome term especially when treated explicitly, causing severe instabilities. High *Re* problems are particularly affected, since large values on the right-hand side of (6.33) might lead to divergence unless heavy relaxation is applied; damping or eliminating the ATC in sensitive areas has been proposed as a solution [138, 181]. For continuous adjoints an implicit treatment of the ATC is found to improve stability [233]. An analogous discrete approach is proposed here for MHFV, namely by introducing a *Velocity-Coupled* (VCPL) preconditioning scheme. VCPL is a version of the MHFV adjoint SIMPLEC in which, in the predictor step, all velocity components are coupled, i.e. kept on the left-hand side and treated implicitly:

$$\left(\overline{\boldsymbol{\mathcal{F}}}^T + \alpha\left(\operatorname{diag}\left(\beta_F\right)\right)\right)\widetilde{\mathbf{u}}^{*,n+1/2} = \widetilde{\mathbf{g}_u}^* - \boldsymbol{\mathcal{D}}^T\mathbf{p}^{*,n} + \alpha\left(\operatorname{diag}\left(\beta_F\right)\right)\widetilde{\mathbf{u}}^{*,n} \;; \tag{6.34}$$

the pressure correction step, on the other hand, remains unchanged.

Besides tackling the stability issues mentioned above, implicit treatment of the ATC can also be expected to reduce the iteration count. The main drawback of the VCPL approach (6.34) is that it requires solution of larger and more stiff linear systems compared to SIMPLEC.

6.2.3 Adjoint Augmented Lagrangian

Lastly, an adaptation of the AL preconditioner (Section 5.2.3) to the adjoint Navier-Stokes system is presented. Applying the AL penalisation mechanism (5.57) yields an augmented adjoint velocity block:

$$\begin{aligned}\overline{\boldsymbol{\mathcal{F}}^{\gamma}}^T &= \overline{\boldsymbol{\mathcal{F}}}^T - \gamma\mu\boldsymbol{\mathcal{D}}^T\left(\operatorname{diag}\left(\frac{1}{|C|}\right)\right)\boldsymbol{\mathcal{G}}^T \\ &\approx \overline{\boldsymbol{\mathcal{F}}}^T - \gamma\mu\boldsymbol{\mathcal{D}}^T\left(\operatorname{diag}\left(\frac{1}{|C|}\right)\right)\boldsymbol{\mathcal{D}}\;,\end{aligned} \tag{6.35}$$

while the adjoint momentum right-hand side becomes:

$$\widetilde{\mathbf{g}_u^{\gamma}}^* = \widetilde{\mathbf{g}_u}^* - \gamma\mu\boldsymbol{\mathcal{D}}^T\left(\operatorname{diag}\left(\frac{1}{|C|}\right)\right)\mathbf{g}_p^*\;. \tag{6.36}$$

The scaling factor μ is taken from the primal, while the penalisation coefficient γ is defined independently.

Similarly to the primal AL, $\boldsymbol{\mathcal{G}}^T$ has been replaced with $\boldsymbol{\mathcal{D}}$ in the adjoint augmentation term (6.35) in order to avoid excessively complex connectivities and stiffness. For the adjoint AL, however, this introduces an inconsistency: the adjoint continuity equation dictates that penalisation be done by a quantity proportional to $\left(\boldsymbol{\mathcal{G}}^T\widetilde{\mathbf{u}}^* - \mathbf{g}_p^*\right)$ exactly. Therefore, if $\boldsymbol{\mathcal{G}}^T \neq \boldsymbol{\mathcal{D}}$, i.e. for the PRS2 pressure scheme (5.18), penalising by $\left(\boldsymbol{\mathcal{D}}\widetilde{\mathbf{u}}^* - \mathbf{g}_p^*\right)$ instead as done in (6.35) leads to an adjoint solution corresponding to a PRS1 (first-order pressure scheme) primal, where the adjoint velocity does not satisfy the adjoint continuity equation

$$\boldsymbol{\mathcal{G}}^T\widetilde{\mathbf{u}}^* = \mathbf{g}_p^* \ . \tag{6.37}$$

A way around the issue is to treat all the second-order contributions in $\boldsymbol{\mathcal{G}}^T$ explicitly, as shown in Algorithm 5.

Algorithm 5 MHFV Adjoint Augmented Lagrangian

$n = 0$
Initialise $\widetilde{\mathbf{u}}^{*,0}$, $\mathbf{p}^{*,0}$
while not converged **do**
  Solve augmented adjoint hybrid momentum equation:
  $\overline{\boldsymbol{\mathcal{F}}^{\gamma}}^T\widetilde{\mathbf{u}}^{*,n+1} = \widetilde{\mathbf{g}_u^{\gamma}}^* - \boldsymbol{\mathcal{D}}^T\mathbf{p}^{*,n} + \gamma\mu\boldsymbol{\mathcal{D}}^T\left(\mathrm{diag}\left(\frac{1}{|C|}\right)\right)\left(\boldsymbol{\mathcal{G}}^T - \boldsymbol{\mathcal{D}}\right)\widetilde{\mathbf{u}}^{*,n}$;
  Compute adjoint pressure correction:
  $\boldsymbol{\delta}\mathbf{p}^* = -\gamma\mu\left(\mathrm{diag}\left(\frac{1}{|C|}\right)\right)\left(\boldsymbol{\mathcal{G}}^T\widetilde{\mathbf{u}}^{n+1} - \mathbf{g}_p^*\right)$;
  Update adjoint pressure:
  $\mathbf{p}^{*,n+1} = \mathbf{p}^{*,n} + \boldsymbol{\delta}\mathbf{p}^*$;
  n=n+1
end while
return $\widetilde{\mathbf{u}}^*$, $\mathbf{p}^*$

6.3 Adjoint shape optimisation and mesh morphing

It was shown in the previous sections how the MHFV adjoint Navier-Stokes problem is assembled and solved, resulting in the discrete adjoint states $\widetilde{\mathbf{u}}^*$ and $\mathbf{p}^*$. The adjoint field on its own is of limited interest to an engineer looking to perform gradient-based optimisation: they will rather be interested in obtaining the gradient of J with respect to a set of design parameters $\boldsymbol{\alpha}$ which they can control directly. For shape optimisation problems based on discrete adjoint formulations, the choice of $\boldsymbol{\alpha}$ typically falls on either CAD parameters which determine the shape of the item being optimised (see e.g. [115, 204, 241, 245]), or directly on the coordinates of a subset of mesh nodes defining the surface of said item (see e.g. [91, 124, 209]).

6.3.1 Preliminary notation

Relative merits and drawbacks of CAD-based and node-based methods have been discussed in the literature [109, 110, 207]. Regardless of the specific approach, a common generic notation can be defined as follows:

- $\boldsymbol{\alpha}$ and $\boldsymbol{\delta\alpha}$ to identify the shape parameters and a displacement applied to them, respectively. In a gradient-based optimisation algorithm, $\boldsymbol{\delta\alpha}$ will typically be oriented in the direction opposite to the gradient $\frac{dJ}{d\boldsymbol{\alpha}}$ computed at the previous iteration, in order to achieve cost function reduction.

- $\mathbf{x}_b$ to denote the coordinates of boundary mesh nodes defining the surfaces whose shape is controlled by $\boldsymbol{\alpha}$, and $\boldsymbol{\delta}\mathbf{x}_b$ to identify their displacements. These nodes shall henceforth be referred to as *free nodes*. The free nodes displacement will be a function of a variation in shape parameters, i.e. $\boldsymbol{\delta}\mathbf{x}_b = \boldsymbol{\delta}\mathbf{x}_b\left(\boldsymbol{\delta\alpha}\right)$. In particular, for a most basic node-based approach it holds $\boldsymbol{\delta}\mathbf{x}_b = \boldsymbol{\delta\alpha}$ and thus $\mathbf{x}_b = \boldsymbol{\alpha}$, i.e. the shape parameters are the free nodes' coordinates themselves, while in a CAD-based case the relationship between $\boldsymbol{\delta\alpha}$ and $\delta\mathbf{x}_b$ will be determined by whichever geometric law is applied by the CAD tool to define the surface shape (hence the distribution of $\mathbf{x}_b$) based on $\boldsymbol{\alpha}$.

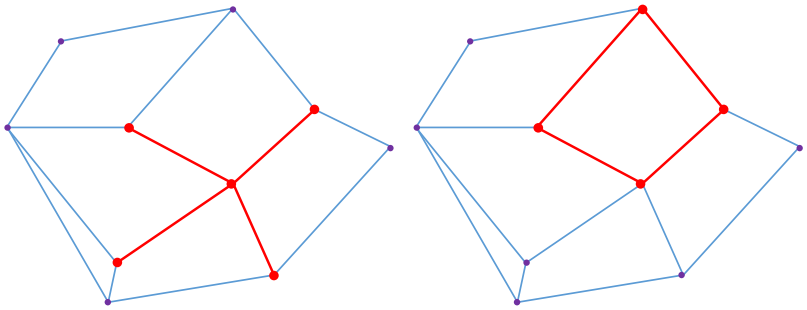
- $\mathbf{x}$ to denote the entire field of coordinates of all mesh nodes, and $\boldsymbol{\delta}\mathbf{x}$ their displacement caused by $\boldsymbol{\delta}\mathbf{x}_b$. The relationship between $\boldsymbol{\delta}\mathbf{x}_b$ and $\boldsymbol{\delta}\mathbf{x}$ will be dictated by a *mesh morpher*: a tool designed to adapt an existing mesh to the movement of a subset of its nodes ($\mathbf{x}_b$ in this case) in order to maintain its overall quality.

An alternative to mesh morphing would be re-meshing the domain at the end of each optimisation iteration, once the deformation $\boldsymbol{\delta}\mathbf{x}_b$ is applied. Re-meshing is however difficult to automate, and it would also cause a loss of consistency in the computed gradient: since the gradient is computed based on the discrete adjoint state (it will be shown how in Section 6.3.3), a re-meshing procedure would effectively "destroy" the very discrete space that the adjoint field belongs to and create a new one. The overall optimisation procedure would most likely still converge - since the shape change induced by $\boldsymbol{\delta\alpha}$ supposedly works towards a reduction of the cost function in a physical sense as well - but the notion of discrete adjoint gradient as *the exact gradient of the cost function*, which is one of the main strengths of the discrete adjoint approach, would be lost. Mesh morphing allows to maintain the same discrete spaces unchanged throughout the whole process: cells, faces, nodes, their topological connectivities - and thus the primal degrees of freedom attached to each - are conserved by the morphing process, which operates in general by moving nodes in an attempt to preserve grid quality -in terms of relative cell sizes, orthogonal-

ity, absence of negative volumes (*element inversion*) etc. - while propagating the free boundary deformation $\boldsymbol{\delta}\mathbf{x}_b$ to the interior mesh.

6.3.2 Rigid Motion Mesh Morpher

Over the years several morphing methodologies have been proposed to adapt an interior mesh to a boundary deformation. Typical examples include *Laplacian Smoothing* [113], *Linear Elasticity Morphing* [156, 219], *Spring Analogy* [34, 84], *Radial Basis Function Morphing* [130].



(a) node-centred (b) cell-centred

Figure 6.5: Examples of stencil definitions (in red) for RMMM.

For the work presented here it was chosen to couple the MHFV Navier-Stokes solver with the *Rigid Motion Mesh Morphing* tool (RMMM), developed by Eleftheriou [77] in the same FTL framework as the primal and adjoint solvers. A coarse outline of RMMM is sketched here. The scheme requires defining, within the grid, n_S stencils of nodes. The definition of stencil is not unique (Figure 6.5): it may be e.g. a central node plus those sharing an edge with it, or all nodes belonging to a FV cell. Then a total *deformation energy* is defined as

$$E = \sum_{S \in \Omega_h} w_S \sum_{i \in S} \mu_{S,i} \left\| \delta\vec{x}_i - \left(\vec{a}_S + \vec{b}_S \times (\vec{x}_i - \vec{x}_{S,C}) \right) \right\|^2 \tag{6.38}$$

where S denotes a stencil, w_S a weight associated with stencil S, $\mu_{S,i}$ a weight associated with node i of stencil S, and $\vec{a}_S$ and $\vec{b}_S$ represent an ideal rigid movement of the stencil (translation and rotation, respectively) about its centre of gravity, $\vec{x}_{S,C}$. RMMM subsequently operates by finding the stencil-wise $\vec{a}_S$, $\vec{b}_S$ and nodal displacements such that the total deformation energy (6.38) is minimised in the least-squares sense. In other words, RMMM reacts to a boundary deformation $\boldsymbol{\delta}\mathbf{x}_b$ by maintaining the displacement of internal nodes as rigid as possible. Variables $\vec{a}_S$ and $\vec{b}_S$ are typically eliminated from

the unknowns by static condensation, ultimately yielding a linear system of the form

$$\mathcal{M}\boldsymbol{\delta}\mathbf{x} = \mathbf{m} \tag{6.39}$$

where $\mathcal{M}$ is a sparse SPD matrix and $\mathbf{m}$ a right-hand side, arising from the static condensation process, which is zero everywhere except for free boundary nodes where the imposed displacement $\boldsymbol{\delta}\mathbf{x}_b$ is non-zero.

In its most basic implementation RMMM enforces the boundary deformation $\boldsymbol{\delta}\mathbf{x}_b$ in the form of strong Dirichlet boundary conditions. A more sophisticated version, developed by Liatsikouras [148], is the *Soft Handle CAD-Free Parametrisation* (SHCFP) which allows instead to impose a *target displacement* to the boundary nodes, or a subset thereof (*handles*), and then attempts to match these target values while enforcing a certain smoothness constraint. This is of particular interest in the context of shape optimisation with a CAD-free parametrisation: using boundary nodes coordinates as shape parameters can lead to computation of noisy gradients and, subsequently, to optimal surface configurations that are too jagged to be of any practical use; the inclusion of a smoothness requirement in the morphing phase helps alleviate the problem. With SHCFP the design parameters $\boldsymbol{\alpha}$ are defined as the target coordinates for those surface nodes selected as handles. The parametrisation modifies the morphing system (6.39) to

$$\widehat{\mathcal{M}}\boldsymbol{\delta}\mathbf{x} = \mathbf{m}_{\delta\alpha} \tag{6.40}$$

where $\widehat{\mathcal{M}}$ is the modified morpher matrix (more specifically: $\widehat{\mathcal{M}} = \mathcal{M} + \mathcal{U}$ with $\mathcal{U}$ a diagonal matrix independent of $\boldsymbol{\alpha}$) and $\mathbf{m}_{\delta\alpha}$ is the modified right-hand side linearly dependent on the target displacement $\boldsymbol{\delta\alpha}$, i.e.

$$\mathbf{m}_{\delta\alpha} = \mathbf{m} + \mathcal{V}\boldsymbol{\delta\alpha} \tag{6.41}$$

with $\mathcal{V}$ a constant matrix.

6.3.3 Final gradient computation

It is now possible to show how one computes the final gradient $\frac{dJ}{d\boldsymbol{\alpha}}$ by taking into account the action of the morphing tool from Section 6.3.2. Firstly, it is observed that the parametrised RMMM system (6.40) can be equivalently written as

$$\widehat{\mathcal{M}}\mathbf{x} = \mathbf{m}_{\delta\alpha} + \widehat{\mathcal{M}}\mathbf{x}_0 \ , \tag{6.42}$$

obtained by replacing $\boldsymbol{\delta}\mathbf{x} = \mathbf{x} - \mathbf{x}_0$ where $\mathbf{x}_0$ are the starting nodal coordinates. In an optimisation algorithm, each iteration starts by computing the new nodal coordinates $\mathbf{x}$ by solving (6.42); these are used to compute inertial quantities reuqired to assemble the primal Navier-Stokes system, which is then solved iteratively with any of the algorithms discussed in Section 5.2. In this perspective, mesh morphing can be seen as part of the primal itself, which can thus be rewritten to include the morpher:

$$\begin{bmatrix} \widehat{\mathcal{M}} & 0 & 0 \\ 0 & \boldsymbol{\mathcal{F}}_{\nu,\vec{U}} & \boldsymbol{\mathcal{G}} \\ 0 & \boldsymbol{\mathcal{D}} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \widetilde{\mathbf{u}} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{m}_{\delta\alpha} + \widehat{\mathcal{M}}\mathbf{x}_0 \\ \widetilde{\mathbf{g}} \\ \mathbf{0} \end{pmatrix} . \tag{6.43}$$

The morpher is independent from the Navier-Stokes problem and must be solved first, because operators $\boldsymbol{\mathcal{F}}_{\nu,\vec{U}}$, $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{D}}$ all depend on $\mathbf{x}$ - as they all require computing inertial quantities (volumes, areas, centres of gravity, etc.) directly dependent on grid coordinates. Furthermore, since the morpher is linear with respect to $\mathbf{x}$, it follows that

$$\frac{\partial \mathbf{r}_x}{\partial \mathbf{x}} = \widehat{\mathcal{M}} \tag{6.44}$$

where $\mathbf{r}_x$ is the residual of the SHCFP-parametrised RMMM system. Hence the discrete adjoint counterpart of (6.43) results in

$$\begin{bmatrix} \widehat{\mathcal{M}} & 0 & 0 \\ \mathcal{P}_u & \overline{\boldsymbol{\mathcal{F}}} & \boldsymbol{\mathcal{G}} \\ \mathcal{P}_p & \boldsymbol{\mathcal{D}} & 0 \end{bmatrix}^T \begin{pmatrix} \mathbf{x}^* \\ \widetilde{\mathbf{u}}^* \\ \mathbf{p}^* \end{pmatrix} = \begin{pmatrix} \mathbf{g}_x^* \\ \widetilde{\mathbf{g}_u}^* \\ \mathbf{g}_p^* \end{pmatrix} \tag{6.45}$$

where $\mathcal{P}_u$ and $\mathcal{P}_p$ correspond respectively to the partial derivative blocks $\frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{r}_p}{\partial \mathbf{x}}$ (which are non-zero because of the aforementioned dependence of $\widetilde{\mathbf{r}_u}$ and $\mathbf{r}_p$ on inertial quantities), and $\mathbf{g}_x^*$ is the adjoint right-hand side for nodal coordinates: $\left(\frac{\partial J}{\partial \mathbf{x}}\right)^T$, which will be non-zero for those nodes whose coordinates directly affect the cost function J. The variable $\mathbf{x}^*$ is thus interpreted as the field of adjoint nodal coordinates. The adjoint system (6.45) can be solved in a sequence mirroring, in reverse, that of the primal, i.e. by solving first the adjoint Navier-Stokes problem for $\widetilde{\mathbf{u}}^*$ and $\mathbf{p}^*$ (with any of the algorithms discussed in Section 6.2), and subsequently the "adjoint morpher" problem:

$$\widehat{\mathcal{M}}^T \mathbf{x}^* = \mathbf{g}_x^* - \mathcal{P}_u^T \widetilde{\mathbf{u}}^* - \mathcal{P}_p^T \mathbf{p}^* . \tag{6.46}$$

Notice that the right-hand side of (6.46) is equivalent to

$$\left(\frac{\partial J}{\partial \mathbf{x}} - (\widetilde{\mathbf{u}}^*)^T \frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{x}} - (\mathbf{p}^*)^T \frac{\partial \mathbf{r}_p}{\partial \mathbf{x}} \right)^T = \left(\frac{dJ}{d\mathbf{x}} \right)^T , \tag{6.47}$$

i.e. the (transpose) adjoint-based gradient of J with respect to all nodes coordinates; this is referred to as *nodal sensitivity field*. In practice, terms $\frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{x}}$, $\frac{\partial \mathbf{r}_p}{\partial \mathbf{x}}$ and $\frac{\partial J}{\partial \mathbf{x}}$ are evaluated by reverse assembly combined with graph colouring the same way as for the Navier-Stokes Jacobian and adjoint right-hand side (see Section 6.1). In fact, reverse-evaluation of a matrix-vector product of the form $(\widetilde{\mathbf{u}}^*)^T \frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{x}}$ can be done in a matrix-free fashion, thus economising on the overall memory footprint. The mechanism - named "reverse apply transpose" in the i-Adjoint tool - is fairly straightforward: each step of the reverse matrix assembly produces a vector which, once masked accordingly, gives all columns of $\frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{x}}$ corresponding to the degrees of freedom of the colour being treated; rather than storing these in a matrix object, the product of each with $\widetilde{\mathbf{u}}^*$ is computed on the fly, which yields directly all entries of the resulting vector for those degrees of freedom.

Finally, the gradient of J with respect to the design variables of interest is computed as

$$\mathbf{s}_A = \frac{dJ}{d\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} - (\widetilde{\mathbf{u}}^*)^T \frac{\partial \widetilde{\mathbf{r}_u}}{\partial \boldsymbol{\alpha}} - (\mathbf{p}^*)^T \frac{\partial \mathbf{r}_p}{\partial \boldsymbol{\alpha}} - (\mathbf{x}^*)^T \frac{\partial \mathbf{r}_x}{\partial \boldsymbol{\alpha}} \tag{6.48}$$

which corresponds to the generic formula for adjoint sensitivity (2.15) applied to an adjoint problem of the form (6.45). As mentioned in Section 6.3.2, in this work a SHCFP-parametrised RMMM is adopted where the shape parameters $\boldsymbol{\alpha}$ correspond to the target coordinates of handle nodes. Neither the objective function nor the Navier-Stokes operator depend directly on $\boldsymbol{\alpha}$, hence $\frac{\partial J}{\partial \boldsymbol{\alpha}} = \mathbf{0}$, $\frac{\partial \widetilde{\mathbf{r}_u}}{\partial \boldsymbol{\alpha}} = \mathbf{0}$ and $\frac{\partial \mathbf{r}_p}{\partial \boldsymbol{\alpha}} = \mathbf{0}$. Thus (6.48) simplifies to

$$\mathbf{s}_A = \frac{dJ}{d\boldsymbol{\alpha}} = -(\mathbf{x}^*)^T \frac{\partial \mathbf{r}_x}{\partial \boldsymbol{\alpha}} \ . \tag{6.49}$$

For convenience, a functionality was added to the parametrised RMMM tool providing directly the hand-derived term $\frac{\partial \mathbf{r}_x}{\partial \boldsymbol{\alpha}}$. Hand-derivation is trivial since the SHCFP parametrisation of the morpher is linear with respect to the target handle displacements $\boldsymbol{\delta\alpha}$. As mentioned, (6.40) can be written in the form

$$\widehat{\mathcal{M}}\boldsymbol{\delta}\mathbf{x} = \mathbf{m} + \mathcal{V}\boldsymbol{\delta\alpha} \ . \tag{6.50}$$

Denoting - with a slight abuse of notation - by $\boldsymbol{\alpha}_0$ the exact position of handle nodes prior to morphing such that $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0 + \boldsymbol{\delta\alpha}$, the residual of (6.50) is

$$\mathbf{r}_x = \widehat{\mathcal{M}}\boldsymbol{\delta}\mathbf{x} - \mathbf{m} - \mathcal{V}(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0) \tag{6.51}$$

which leads to

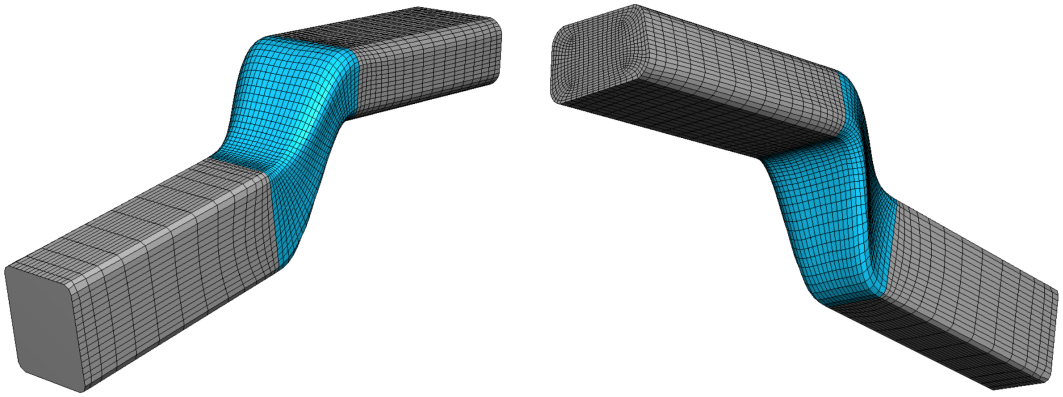
$$\frac{\partial \mathbf{r}_x}{\partial \boldsymbol{\alpha}} = -\mathcal{V}. \tag{6.52}$$

6.4 Validation of MHFV adjoint Navier-Stokes

6.4.1 Sensitivity and gradient validation

Validation of the MHFV adjoint solver shall be performed on the *S-bend* test case already described in Section 5.3.4, but with a slightly lower Reynolds number and on a coarser mesh. The model - specifically selected for the AboutFlow project - is found in adjoint literature frequently enough that it can be considered a benchmark case [120, 138, 181, 182]. The fluid is air ($\nu = 1.589\ \mathrm{E}^{-5}\ m^2/s$) and the inlet velocity is set to 0.1 m/s, leading to a $Re \approx 300$, which is the setting most often considered in the literature as it guarantees steady laminar flow with fully developed flow conditions at the outlet. The geometry and mesh are shown in Figure 6.6; the mesh is composed of 41044 hexahedral cells. The cost function is defined as the the total power loss:

$$J = -\int_{\partial\Omega} \rho \left(p + \frac{1}{2} \left\| \vec{U} \right\|^2 \right) \vec{U} \cdot \vec{n}\, dS\ . \tag{6.53}$$



(a) right side (b) left side

Figure 6.6: S-bend: geometry and mesh.

The soft handle parametrised morpher described in Section 6.3.2 is used. More specifically, all surface nodes lying on the middle section of the S-bend - highlighted in blue in Figure 6.6 - are selected as handle nodes to which a target displacement $\boldsymbol{\delta\alpha}$ is imposed, while all other boundary nodes are kept fixed. The design variables $\boldsymbol{\alpha}$ are thus the target coordinates for the handle nodes, giving a design space with a total of 5544 degrees of freedom (1848 handle nodes $\times$ 3 coordinates each).

MHFV discretisation strategies are set as follows: OVRN weight type for viscous terms; ULSQR-stabilised second-order upwinding for convective terms; PRS2 scheme for

pressure. In order to minimise discrepancies due to partially converged fields, the primal is converged down to a rather strict tolerance on scaled residuals (10^{-8}) while the adjoint systems (Navier-Stokes and RMMM) are solved to machine precision with a direct linear solver.

A first way of validating a discrete adjoint solver consists in comparing against a FD gradient projected onto the direction of the adjoint gradient itself. This is done by applying to all design variables a displacement in the direction of the gradient $\mathbf{s}_A$:

$$\boldsymbol{\delta\alpha} = \lambda \frac{(\mathbf{s}_A)^T}{\|\mathbf{s}_A\|}$$

where λ is a step-length factor. The primal is then re-run and a new cost function $J(\boldsymbol{\alpha} + \boldsymbol{\delta\alpha})$ is computed; a FD gradient $\mathbf{s}_{FD}$ projected in direction of $\mathbf{s}_A$ is subsequently computed as

$$\|\mathbf{s}_{FD}\| = \frac{J(\boldsymbol{\alpha} + \boldsymbol{\delta\alpha}) - J(\boldsymbol{\alpha})}{\lambda}$$

and its value compared with that of $\mathbf{s}_A$ itself projected onto its own direction: $\mathbf{s}_A \cdot \frac{\mathbf{s}_A}{\|\mathbf{s}_A\|} = \|\mathbf{s}_A\|$. Finally, a relative error - or better a relative discrepancy between adjoint and FD gradient - is evaluated by scaling by an average gradient value:

$$\epsilon(s) = \left| \frac{\|\mathbf{s}_A\| - \|\mathbf{s}_{FD}\|}{\frac{\|\mathbf{s}_A\| + \|\mathbf{s}_{FD}\|}{2}} \right| .$$

Running this test on the S-bend case with factor $\lambda = 10^{-4}$ gives a relative error $\epsilon(s) = 1.18\ \mathrm{E}^{-3}$, a small enough value which constitutes a first validation of the code.

A more detailed check is then performed on the values of single gradient components: five handle nodes are selected at random and, for each, a FD-based sensitivity of J (step-length factor $\lambda = 10^{-4}$) is calculated with respect to each coordinate. Results are reported in Table 6-B, where relative errors are again scaled by an averaged gradient component magnitude. Results are very positive: for all five nodes, and for each component, the FD and adjoint gradients consistently agree not only in terms of direction (sign), but also in terms of magnitude within a relative difference in the order of 10^{-2}.

Lastly, a qualitative validation of the adjoint sensitivity comes from observing the full nodal sensitivity field (Figure 6.7). As shown in Section 6.3.3, this corresponds to the gradient of J with respect to the position of all mesh nodes, surface and volume alike; therefore, the vectors shown in figure identify the direction in which each node should be moved to obtain a (first-order) maximal increase in J. A few observations are in order. Firstly, there is a noticeable high-sensitivity ring at the inlet, represented by

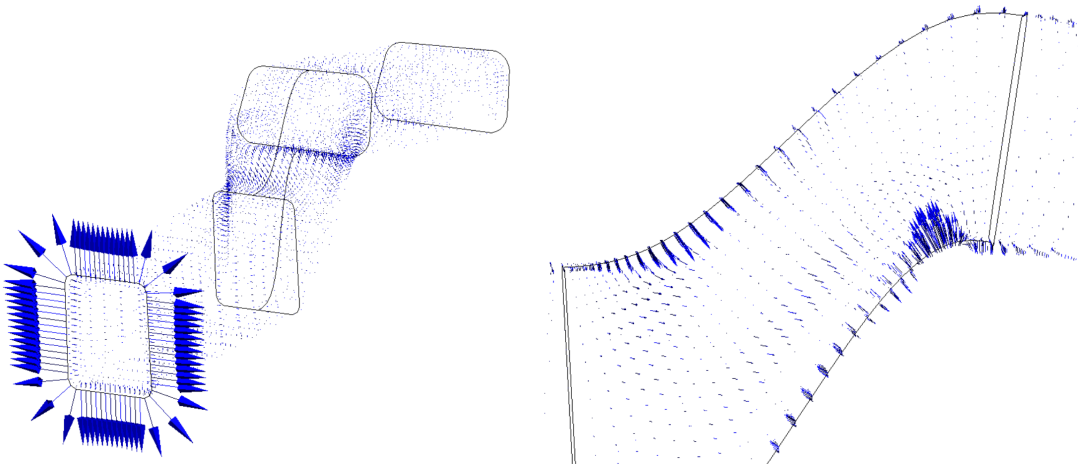
Table 6-B: S-bend test case: comparison of FD and adjoint gradient components for five randomly selected handle nodes.

		FD grad.	adjoint grad.	relative error
handle node A	x	$3.975\ \mathrm{E}^{-7}$	$4.075\ \mathrm{E}^{-7}$	$2.490\ \mathrm{E}^{-2}$
	y	$-1.016\ \mathrm{E}^{-7}$	$-1.065\ \mathrm{E}^{-7}$	$4.705\ \mathrm{E}^{-2}$
	z	$2.222\ \mathrm{E}^{-7}$	$2.221\ \mathrm{E}^{-7}$	$4.154\ \mathrm{E}^{-4}$
handle node B	x	$1.949\ \mathrm{E}^{-7}$	$1.930\ \mathrm{E}^{-7}$	$9.744\ \mathrm{E}^{-3}$
	y	$-1.015\ \mathrm{E}^{-7}$	$-1.004\ \mathrm{E}^{-7}$	$1.089\ \mathrm{E}^{-2}$
	z	$1.269\ \mathrm{E}^{-7}$	$1.241\ \mathrm{E}^{-7}$	$2.242\ \mathrm{E}^{-2}$
handle node C	x	$4.159\ \mathrm{E}^{-7}$	$4.102\ \mathrm{E}^{-7}$	$1.399\ \mathrm{E}^{-2}$
	y	$-1.642\ \mathrm{E}^{-6}$	$-1.683\ \mathrm{E}^{-6}$	$2.456\ \mathrm{E}^{-2}$
	z	$-8.783\ \mathrm{E}^{-7}$	$-8.923\ \mathrm{E}^{-7}$	$1.572\ \mathrm{E}^{-2}$
handle node D	x	$6.177\ \mathrm{E}^{-7}$	$6.067\ \mathrm{E}^{-7}$	$1.798\ \mathrm{E}^{-2}$
	y	$-2.705\ \mathrm{E}^{-6}$	$-2.768\ \mathrm{E}^{-6}$	$2.286\ \mathrm{E}^{-2}$
	z	$-1.085\ \mathrm{E}^{-6}$	$-1.097\ \mathrm{E}^{-6}$	$1.076\ \mathrm{E}^{-2}$
handle node E	x	$8.743\ \mathrm{E}^{-7}$	$8.672\ \mathrm{E}^{-7}$	$8.185\ \mathrm{E}^{-3}$
	y	$-3.000\ \mathrm{E}^{-6}$	$-3.083\ \mathrm{E}^{-6}$	$2.715\ \mathrm{E}^{-2}$
	z	$-1.535\ \mathrm{E}^{-6}$	$-1.557\ \mathrm{E}^{-6}$	$1.461\ \mathrm{E}^{-2}$

outward vectors several orders of magnitude larger than sensitivity values elsewhere in the domain. This may be interpreted as a mathematical side-effect of the adjoint approach: the cost function J, i.e. the discrete power loss (6.53), is proportional to the difference in total pressure between the inlet and the outlet; increasing the inlet area indefinitely would indeed lead to a larger integrated total pressure at the inlet, and subsequently a larger drop J. This effect is however irrelevant from an engineering viewpoint: the mesh morpher will ultimately enforce zero displacement for all boundary nodes except for those in the mid-section of the duct, hence inlet nodal sensitivities will not play a role in the final gradient computation.

Another interesting remark is that sensitivity values are nearly zero for all internal nodes. This is perfectly logical and can be interpreted as a good sign of the correctness of the sensitivity field: displacing an internal node does not modify the shape of the duct, and therefore it should not impact the objective function at all. In practice, in a discrete setting, the movement of an internal node does have an impact on local inertial quantities, which propagates to the operators and thus to the flow variables, ultimately affecting the discrete cost function as well. This influence however remains very limited, provided that local grid quality is sufficient in terms of refinement and expansion rate (i.e. change of cell volume with respect to neighbouring cells). For similar reasons the boundary nodal sensitivity appears to be orthogonal (or almost) to the surface itself, as visible in Figure 6.7(b) where the nodal sensitivity is shown in detail in the area of

interest. This is in agreement with continuous adjoint theory: when considering the displacement of a boundary node, any component other than that in the direction of the surface normal has no first-order impact on the shape of the duct. In a discrete framework, sensitivity will most likely not be exactly orthogonal everywhere, but any non-orthogonal component will be very limited in magnitude provided that the surface mesh is sufficiently refined.



(a) full duct (b) zoom on mid-section

Figure 6.7: S-bend: nodal sensitivity field.

6.4.2 Performance of colouring algorithms and reduced assembly

The S-bend case from the previous section is also used to test how the colouring algorithms implemented in i-Adjoint compare against each other and how they respond to a change in graph connectivity. Two configurations are tested: first-order and second-order pressure scheme (PRS1 and PRS2). In both cases the convective scheme is set to ULSQR-stabilised second-order upwinding; this choice however is irrelevant here, since any other convective strategy from Chapter 4, regardless of its order of accuracy, would act on the same local stencils, and thus graph colouring would not be affected.

The comparison is done by checking the number of colours required by each algorithm to colour graphs related to each adjoint term: the Navier-Stokes Jacobian $\frac{\partial \mathbf{r}_{NS}}{\partial(\vec{\mathbf{u}},\mathbf{p})}$; the adjoint right-hand side $\frac{\partial J}{\partial(\vec{\mathbf{u}},\mathbf{p})}$; the tangent matrix with respect to nodal coordinates $\frac{\partial \mathbf{r}_{NS}}{\partial \mathbf{x}}$; the direct dependency of J on nodal coordinates $\frac{\partial J}{\partial \mathbf{x}}$. In this instance the reduced assembly option from Section 6.1.6 is not used, meaning that the Navier-Stokes Jacobian is reverse-assembled in full and therefore the colouring of its graph involves all velocity

and pressure degrees of freedom as graph vertices.

Table 6-C: S-bend, first-order scheme (PRS1): performance of colouring algorithms.

	Number of colours			
	$\frac{\partial \mathbf{r}_{NS}}{\partial(\tilde{\mathbf{u}},\mathbf{p})}$	$\frac{\partial J}{\partial(\tilde{\mathbf{u}},\mathbf{p})}$	$\frac{\partial \mathbf{r}_{NS}}{\partial \mathbf{x}}$	$\frac{\partial J}{\partial \mathbf{x}}$
Welsh-Powell	29	9	39	44
DSATUR	25	7	30	35
MDSATUR	25	7	30	36
Planar-6	26	8	30	36
Modified Planar-6	26	8	31	35

Table 6-D: S-bend, second-order scheme (PRS2): performance of colouring algorithms.

	Number of colours			
	$\frac{\partial \mathbf{r}_{NS}}{\partial(\tilde{\mathbf{u}},\mathbf{p})}$	$\frac{\partial J}{\partial(\tilde{\mathbf{u}},\mathbf{p})}$	$\frac{\partial \mathbf{r}_{NS}}{\partial \mathbf{x}}$	$\frac{\partial J}{\partial \mathbf{x}}$
Welsh-Powell	48	9	122	44
DSATUR	39	7	89	35
MDSATUR	38	7	87	36
Planar-6	40	8	90	36
Modified Planar-6	40	8	90	35

Results are reported in Tables 6-C and 6-D. The most noticeable outcome is that, as expected, switching to a second-order stencil for the pressure variable (Figure 6.3) has a significant impact on colouring, with the required number of colours almost doubled for the Navier-Stokes Jacobian and tripled for the nodal tangent matrix $\frac{\partial \mathbf{r}_{NS}}{\partial \mathbf{x}}$, a direct consequence of the decreased graph sparsity. On the other hand, the results for vectors $\frac{\partial J}{\partial(\tilde{\mathbf{u}},\mathbf{p})}$ and $\frac{\partial J}{\partial \mathbf{x}}$ are identical in both cases: this is due to the fact that the total power loss J is always computed the same way regardless of the discretisation scheme, and thus the degrees of freedom involved are the same. More specifically, the probe-wise power loss is computed at a boundary face F (adjacent to cell C) through the expression

$$j_F = -\rho \left(\frac{1}{2} \left\| \vec{u}_F \right\| + p_C + \nabla_C^{\mathcal{L},\mu} p \cdot (\vec{x}_F - \vec{x}_C) \right) U_{FC} \ , \tag{6.54}$$

which uses boundary hybrid velocity components u_F^i and the pressure value extrapolated to the boundary via the LSQ gradient $\nabla_C^{\mathcal{L},\mu} p$, hence acting on a second-order stencil around F (Figure 6.3(b)).

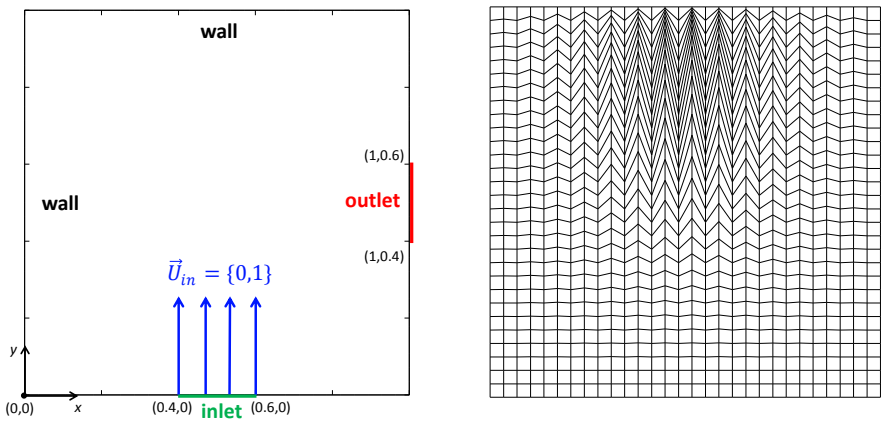
The Welsh-Powell algorithm underperforms compared to all others: it produces roughly 15% to 30% more colours in the first-order scenario and 20% to 40% more in the second-order case, indicating that, at least for this test case, not only is the algorithm less efficient, but such defect worsens when graph sparsity is reduced. Concerning DSATUR and Planar-6 there are no significant differences between the two other than a slight advantage of the former; the same can be said for the modified versions of both, which behave almost identically to their respective unmodified counterparts.

Table 6-E: Performance of full and reduced reverse assembly

	full	reduced
no. colours	25	23
DoFs per colour	3	1
CPU time (s)	44.53	16.36

One last test performed here concerns the reduced assembly described in Section 6.1.6, where only the term $\boldsymbol{\mathcal{T}} = \frac{\partial \widetilde{\mathbf{r}_u}}{\partial \mathbf{U}}$ is reverse-assembled automatically while all other (linear) Jacobian blocks are assembled by hand. The same S-bend test case is used with PRS1 pressure scheme and ULSQR convecting scheme, which is solution-independent and thus satisfies the conditions required by the reduced assembly. DSATUR is used for colouring. As shown in Table 6-E, the reduced assembly runs 2.72 times faster (close to the expected $d = 3$) than the full assembly approach, due to the fact that each vertex in the graph for $\boldsymbol{\mathcal{T}}$ carries only one degree of freedom (the convecting flux U_F).

6.4.3 Algorithm performance

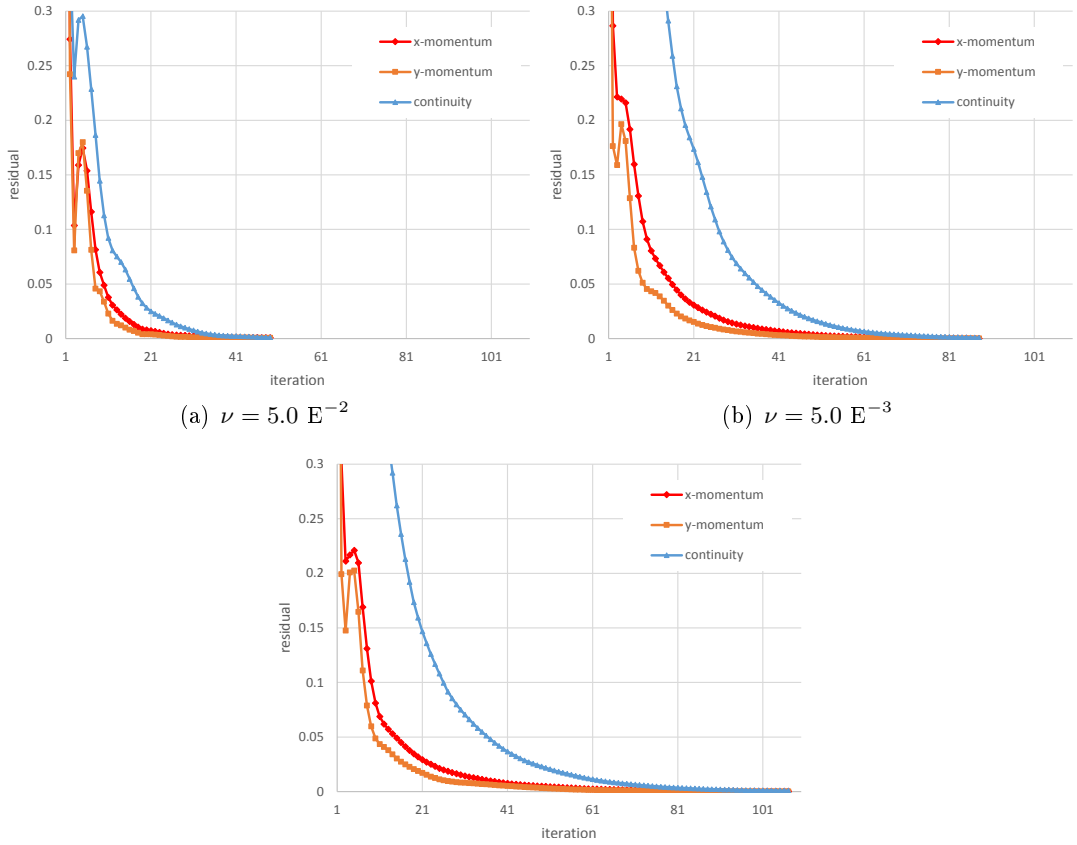


(a) boundary conditions (b) mesh type

Figure 6.8: "Inlet-outlet" test case setup.

A basic 2D "inlet-outlet" test case is devised to test how the adjoint solution algorithms presented in Section 6.2 compare against each other. The case is defined over the square domain $\Omega =]0,1[\times]0,1[$ with inlet, outlet and wall boundary conditions set as in Figure 6.8(a); the inlet y-velocity is set to $v_{in} = 1$ and three different values of kinematic viscosity, $\nu = 5.0\ \mathrm{E}^{-2}$, $5.0\ \mathrm{E}^{-3}$ and $4.0\ \mathrm{E}^{-3}$ are tested to verify how algorithm performance responds to a variation in the Reynolds number. The mesh is chosen to be a quadrilateral grid of size 30×30; applying the distortion pattern shown in Figure 6.8(b) produces a highly skewed and non-orthogonal grid (maximum non-orthogonality angle $\max(\theta) \approx 78.19°$), which allows once again to verify the mesh-independence of the overall scheme. For the primal, the adopted MHFV strategies are: OVRN weights for the viscous term; ULSQR-stabilised second-order upwinding for the convective term; PRS2 scheme for pressure. Each primal equation is solved down to a scaled residual of 10^{-6}, while adjoint tolerance for all algorithms is set to 10^{-3}; both primal and adjoint residual norms are scaled as shown in Section 5.3.3. Direct linear solvers are used throughout.

In order to have a fair comparison between adjoint SIMPLEC and VCPL, each case

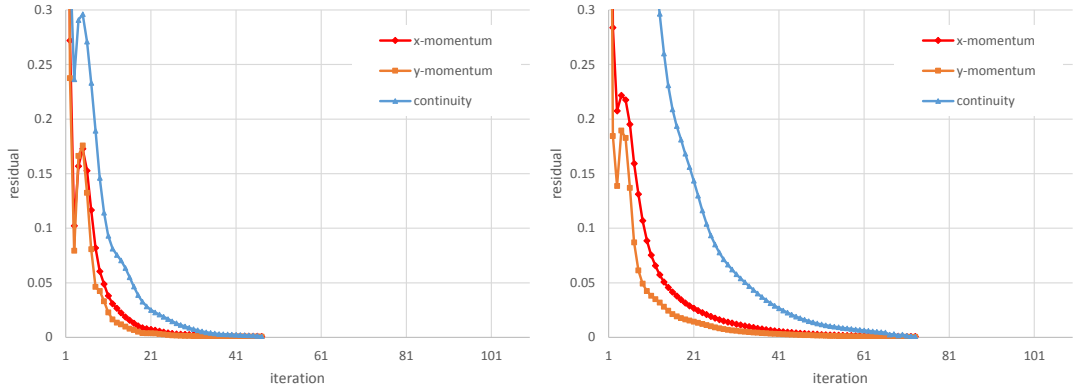


(a) $\nu = 5.0\ \mathrm{E}^{-2}$ (b) $\nu = 5.0\ \mathrm{E}^{-3}$

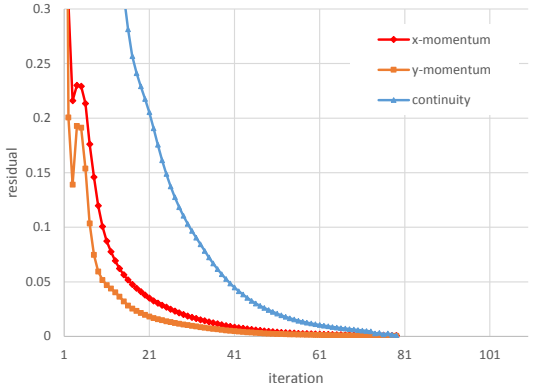
(c) $\nu = 4.0\ \mathrm{E}^{-3}$

Figure 6.9: Adjoint SIMPLEC: convergence history at different Re.

is run with its respective optimal relaxation factor α (in terms of iteration count) which is found empirically; the obtained values are reported in Table 6-F, along with the corresponding iteration count for each load case and the speedup accomplished by VCPL. Results indicate that in a low-Re scenario the two algorithms exhibit an almost identical behaviour, while for the two lower viscosity cases VCPL reduces the iteration count by roughly 17% and 26% compared to SIMPLEC. This may be attributed to the fact that



(a) $\nu = 5.0\ \mathrm{E}^{-2}$ (b) $\nu = 5.0\ \mathrm{E}^{-3}$



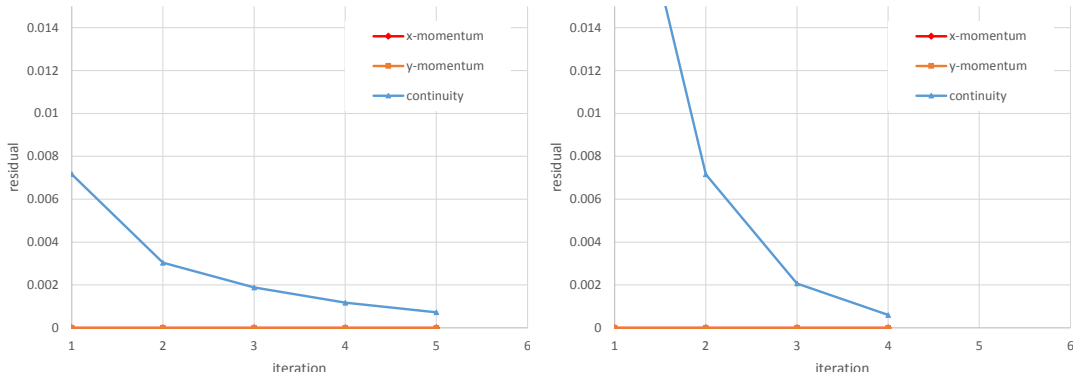
(c) $\nu = 4.0\ \mathrm{E}^{-3}$

Figure 6.10: Adjoint Velocity-Coupled: convergence history at different Re.

Table 6-F: Adjoint SIMPLEC and VCPL: optimal α values, iteration count and VCPL speedup at different Re.

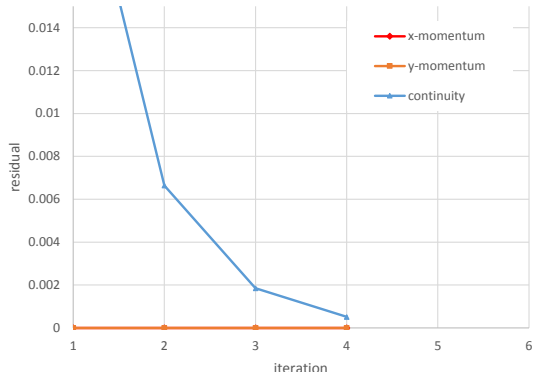
		SIMPLEC	Velocity-Coupled	speedup
$\nu = 5.0\ \mathrm{E}^{-2}$	α	$3.0\ \mathrm{E}^{-1}$	$2.9\ \mathrm{E}^{-1}$	4.08%
	no. iter.	49	47	
$\nu = 5.0\ \mathrm{E}^{-3}$	α	$1.6\ \mathrm{E}^{-1}$	$2.0\ \mathrm{E}^{-1}$	17.04%
	no. iter.	88	73	
$\nu = 4.0\ \mathrm{E}^{-3}$	α	$2.3\ \mathrm{E}^{-1}$	$2.1\ \mathrm{E}^{-1}$	26.17%
	no. iter.	107	79	

the ATC term is larger for larger Re, and thus the benefits of treating it implicitly - as in the VCPL algorithm - become more evident as the problem becomes more convection-dominated. In all scenarios, however, the convergence behaviour of both schemes remains remarkably similar, as highlighted by the convergence history plots (Figure 6.9 and 6.10). It can thus be concluded that, in the context of discrete adjoint MHFV Navier-Stokes, the implicit treatment of the ATC does not drastically modify the convergence slope, but it does yield a moderate performance improvement in higher Re cases.



(a) $\nu = 5.0\ \mathrm{E}^{-2}$

(b) $\nu = 5.0\ \mathrm{E}^{-3}$



(c) $\nu = 4.0\ \mathrm{E}^{-3}$

Figure 6.11: Adjoint Augmented Lagrangian: convergence history at different Re.

On the other hand, the adjoint AL scheme - run here with a penalisation factor $\gamma = 3.0\ \mathrm{E}^{+1}$ - outperforms by far both SIMPLEC and VCPL: as highlighted by the convergence history plots (Figure 6.11), the iteration count is reduced in all cases by roughly an order of magnitude. Adjoint AL thus performs similarly to its primal counterpart. The convergence plots also reveal a peculiar behaviour of the algorithm: momentum residuals appear to be reduced to near-machine precision from the very start of the solution process and do not change, while the continuity residual is progressively driven to zero by the AL penalisation mechanism.

Chapter 7

Applications

This chapter presents some results of full adjoint-based optimisation processes. Test cases are selected from a list suggested by the AboutFlow project. They are intended to represent simulations of industrial interest, although they remain rather academic in terms of problem size and complexity.

7.1 S-bend

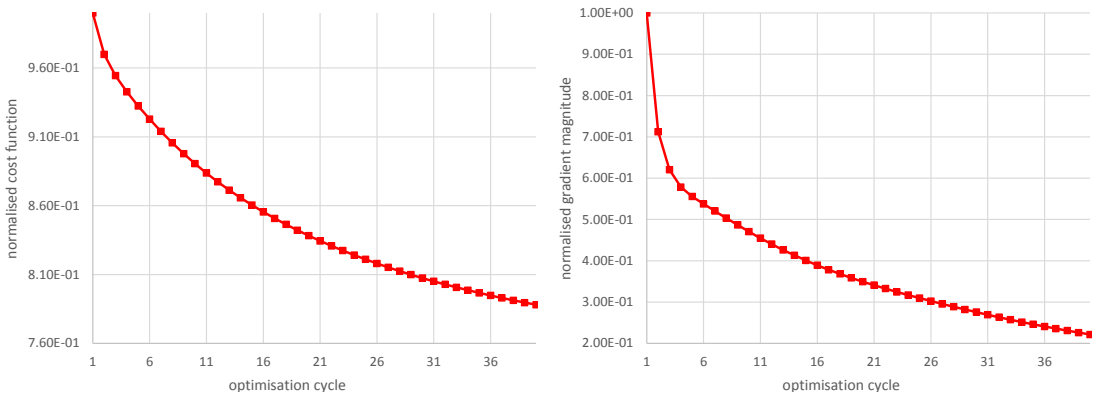
The *S-bend* test case, as described in Section 6.4.1, is hereby considered. As mentioned, this is a 3D shape optimisation case for the laminar internal flow of air through the S-shaped duct part of a car's HVAC system. The inlet velocity is set to 0.1 m/s which gives a Reynolds number $Re \approx 300$ based on the height of the duct at the inlet. The mesh is composed of 41044 hexahedral cells. The soft handle parametrised RMMM morpher is adopted, with all surface nodes lying on the "neck" (the middle section) of the S-bend (Figure 6.6) acting as handle nodes - and thus their target coordinates acting as design variables, producing a design space with a total of 5544 degrees of freedom; all other boundary nodes are kept fixed. Discretisation strategies are set as follows: OVRN weight type for viscous terms; ULSQR-stabilised second-order upwinding for convective terms; PRS2 scheme for pressure. The primal is converged to a tolerance of 10^{-6} on scaled residuals; the adjoint systems (Navier-Stokes and RMMM) are solved to machine precision with a direct linear solver. The cost function is the total power loss (6.53), computed probe-wise on each boundary face as in (6.54).

A basic optimisation algorithm is selected: the *steepest descent* method (described in e.g. [12]), which operates by displacing the shape parameters at each iteration in the

direction opposite to that of the adjoint gradient $\mathbf{s}_A$, scaled by a user-defined factor λ, i.e.

$$\boldsymbol{\delta\alpha}^{n+1} = -\lambda \left(\mathbf{s}_A{}^n\right)^T \quad . \tag{7.1}$$

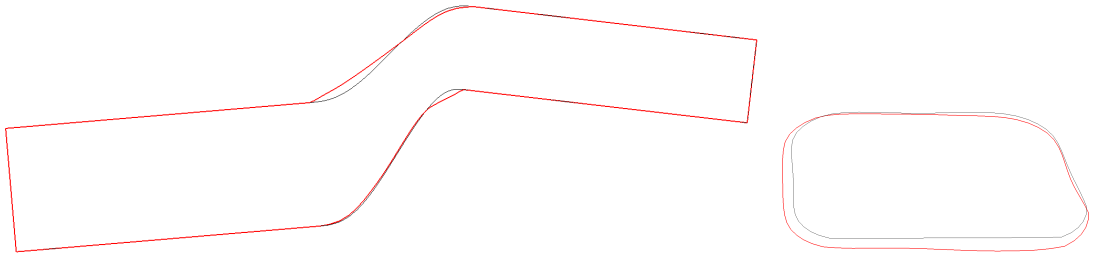
Hence the value of $\boldsymbol{\alpha}$ is displaced in the direction which, according to the gradient, will produce the strongest first-order reduction in J. Since the displacement is proportional to the gradient magnitude $\|\mathbf{s}_A\|$, it will be progressively smaller as the algorithm converges towards a local minimum. Selecting an appropriately small factor λ allows to keep the algorithm at each cycle within a scope where the linear assumption on J holds reasonably well, thus ensuring that J is consistently reduced from one cycle to the next. λ should also be selected in order to limit the target displacement for handle nodes below a threshold that the RMMM morpher can accept - the morphing tool, being itself based on a linearisation, can only handle relatively small deformations unless sub-cycling (subdivision of $\boldsymbol{\delta\alpha}$ into smaller consecutive steps) is applied. In the specific case of the S-bend, the largest gradient components at the initial state are in the order of 10^{-5}; setting $\lambda = 10^3$ produces initial target displacements in the order of 10^{-2} m, small enough with respect to the average height of the duct (≈ 0.1 m) but sufficiently large to produce a reasonable reduction of J at each cycle.



(a) normalised cost function: J/J_0 (b) normalised gradient magnitude: $\|\mathbf{s}_A\|/\|\mathbf{s}_{A0}\|$

Figure 7.1: S-bend: optimisation convergence history.

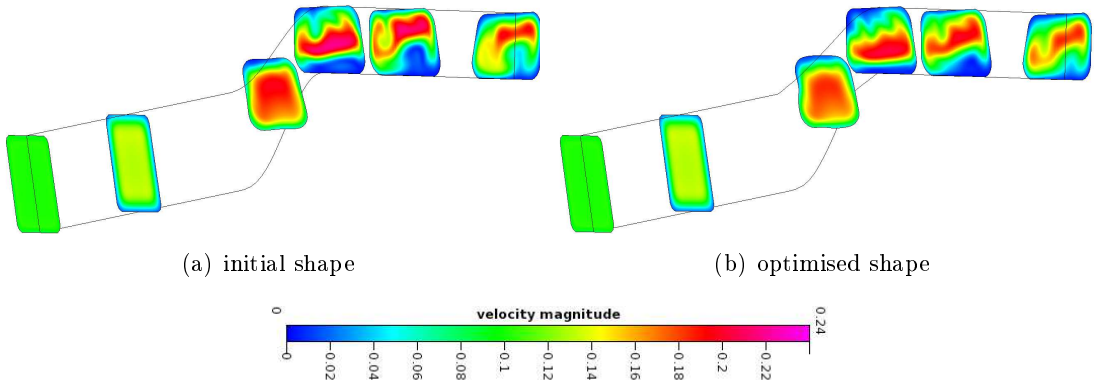
The convergence history of the optimisation process is reported in Figure 7.1(a) in terms of normalised cost function J/J_0 (J_0 being the initial value of J) over 40 optimisation iterations, where a 21.2% improvement on J is achieved. The steepest descent algorithm behaves as expected, converging towards a local minimum as confirmed by the steady decrease in gradient magnitude, Figure 7.1(b). In fact, the plots show that the at cycle 40 the algorithm is not yet fully converged since both J and gradient magnitude appear to be still decreasing, suggesting that there is still some room for improvement.



(a) longitudinal section (b) cross-section (neck exit)

Figure 7.2: S-bend: sections of the original shape (black) and optimised (red).

A comparison between the initial and optimised duct shape is shown in Figure 7.2. The optimised shape features two additional bulges: one on the upper side at the entrance of the neck, and one on the lower side of the duct at the exit of the neck (the latter is visible in the cross-section in Figure 7.2(b)). The duct appears overall inflated with respect to its original geometry. A rather abrupt change in surface curvature is also visible at both ends of the neck, where patches of free nodes are jointed to fixed ones; this is a consequence of the fact that the mesh morpher does not impose smoothness across patches, and is thus to be interpreted as a numerical limitation of the method rather than a physically significant result. It is worth noticing however that, despite the cumbersome geometry of these areas and the coarseness of the mesh, the MHFV primal solver still manages to converge with ease.



(a) initial shape (b) optimised shape

Figure 7.3: S-bend: velocity magnitude (m/s) at different cross-sections.

Results are overall in line with those presented in previous literature: the location and magnitude of the inflated areas closely match those shown by Helgason et al. [120], despite their geometry being slightly different. Xu et al. [241] also report a similar inflation, although in their work they also observe an inward hollowing pattern on both sides of the duct and consider it as an attempt at suppressing secondary flows known as *Dean vortices*. Dean vortices, first investigated by W.R. Dean [68], appear in the cross-

section of internal flows where the curvature of the duct varies - such as for the S-bend - as pairs of counter-rotating vortices superimposed to the primary flow (Figure 7.4). They are a consequence of the increase in pressure - and subsequent decrease in velocity - on the convex side of the bend caused by the adverse pressure gradient generated by the change in curvature, and vice-versa on the concave side. They are known to be a major cause of power loss [32]. By suppressing secondary flows, Xu et al. [241] do achieve larger deformations and a slightly better reduction of J (25.5%), which is likely due do the difference in the selection of shape parameters: theirs is a CAD-based approach where the boundary is represented via *Non-Uniform Rational B-Splines* (NURBS) surface patches [192], featuring a total of 1920 degrees of freedom; their design space is therefore smaller with respect to the one presented here, but it enforces better smoothness of the final shape and is less subject to gradient noisiness which appears to facilitate the optimisation process. Perhaps a similar result could be achieved with the RMMM morpher by reducing the handle nodes to a selected subset of the ones used here.

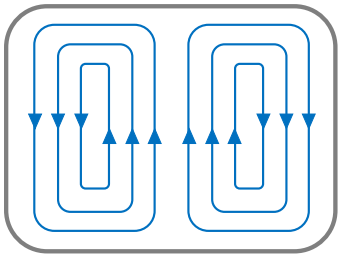


Figure 7.4: Schematics of a secondary flow (pair of counter-rotating Dean vortices) in the cross-section of a curved duct.

On the other hand, the physical effects observable in the optimised geometry remain fairly similar. This is highlighted in Figure 7.3, where velocity magnitude contours are plotted for both geometries at different cross-sections of the duct: in the optimised configuration, the separated flow on the lower side downwind of the neck (the large low-velocity bubbles visible in the last three cross-sections) is visibly reduced, and the flow is made overall more uniform.

7.2 U-bend

A second internal flow test case considered here is the so-called *U-bend*, initially presented by the Von Karman Institute for Fluid Dynamics (VKI) [59, 232] and subsequently become a benchmark case for CFD shape optimisation (see e.g. [81, 172]), similarly to the S-bend above. The geometry is that of the trait of a duct going trough a 180° bend, such as those typically found in serpentine ducts of internal cooling channels of

turbine blades. It has been observed that the U-bends that connect consecutive passages of such ducts represent regions of strong pressure loss. Several early experimental results (e.g. Chang et al. [51], Monson et al. [169], Cheah et al. [52]) all attribute the loss to two main physical phenomena: a) the presence of secondary flows caused by an imbalance between curvature-induced centrifugal forces and pressure along the duct's cross-section (as already observed for the S-bend in Section 7.1), and b) the formation of a separation bubble adjacent to the internal wall on the exiting part of the bend, caused by an adverse pressure gradient (see Figure 7.8). The U-bend geometry is therefore worth investigating in terms of shape optimisation.

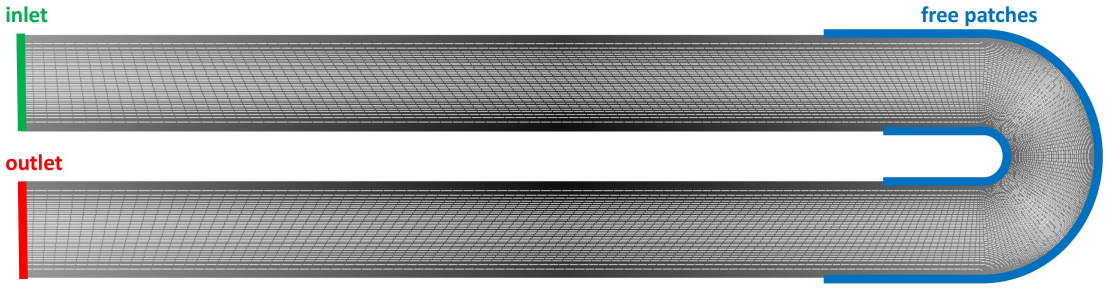
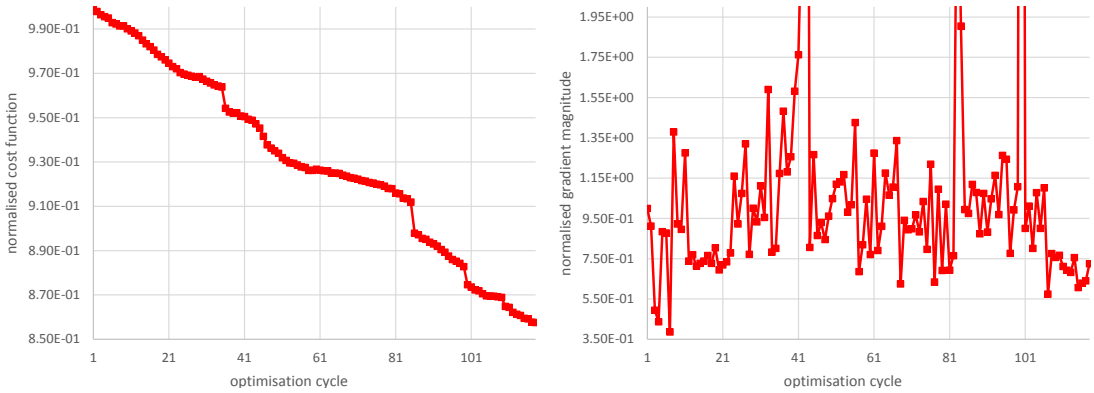


Figure 7.5: U-bend: geometry, mesh and boundary conditions.

The case may be investigated in either 2D or 3D; for reasons of computational power, the former option is hereby chosen for the present work. Geometry and mesh are shown in Figure 7.5: the mesh counts 26433 quadrilateral cells and 26800 nodes. The duct has a constant diameter of 0.075 m, and the bend is semi-circular, with an inner radius of 0.0195 m and an outer radius of 0.0945 m. Both the inlet and outlet legs have a length of 0.75 m, which was empirically found to be sufficient to guarantee a fully developed flow both at the start of the circular bend and at the outflow plane. The inlet velocity is set to 1 m/s and the fluid is air ($\nu = 1.589\ \mathrm{E}^{-5}$), which gives a Reynolds number $Re \approx 4700$. The flow regime is therefore (slightly) turbulent, which is tackled through the RANS approach combined with the *Spalart-Allmaras* (SA) turbulence model [216]. SA is a one-equation model which requires solving an additional transport equation for a viscosity-like variable ν_{sa}, from which an "eddy viscosity" ν_t is computed which models the macroscopic effects of the presence of turbulence; the fluid's physical kinematic viscosity ν in the Navier-Stokes equations is replaced with an effective viscosity $(\nu + \nu_t)$. The boundary layer mesh is refined such that the non-dimensional wall distance y^+ for the centres of wall-adjacent cells is in the range of 1, as required for this type of model (see e.g. [53, 238] for details). Further details on the MHFV implementation of SA are provided in Appendix A.

The additional SA equation in the primal implies, in theory, that a corresponding



(a) normalised cost function: J/J_0 (b) normalised gradient magnitude: $\|\mathbf{s}_A\|/\|\mathbf{s}_{A0}\|$

Figure 7.6: U-bend: optimisation convergence history.

adjoint SA equation should be assembled and solved for an adjoint turbulent viscosity ν_{sa}^*. This represents however an extra computational cost as well as a further risk of instability in the adjoint solution algorithm; it is therefore chosen to operate under the so-called *frozen turbulence* assumption: the SA equation is not differentiated and the turbulent variable is treated as a passive scalar, i.e. the variation in the ν_{sa} field induced by a perturbation $\boldsymbol{\delta\alpha}$ in shape parameters is neglected. The approach effectively invalidates the consistency of the adjoint-based gradient; however, in practice, it is found that under the assumption of sufficiently small displacements the computed gradient is close enough to the correct one in terms of magnitude and direction, and thus can safely be used within an optimisation algorithm. Examples of successful industrial applications of adjoint-based optimisation under the frozen turbulence assumption can be found in [138, 173, 181].

The objective function is the total power loss (6.53). Primal discretisation strategies are selected as: OVRN weight type for viscous terms; HUPW1 scheme for convective terms, for both Navier-Stokes and turbulence model; PRS2 scheme for pressure. The Navier-Stokes are converged down to a tolerance of 10^{-5} on scaled residuals; the SA equation is converged to 10^{-4}; the adjoint systems (Navier-Stokes and RMMM) are solved to machine precision with a direct linear solver. The handle nodes are chosen to be all boundary nodes defining the bend itself - both the inner and outer side - as well as part of the straight leg on either side of it (blue in Figure 7.5), which gives a design space with a total of 424 degrees of freedom (212 nodes $\times$ 2 coordinates each).

Concerning the optimiser, it was empirically determined that setting a step-length factor $\lambda = 1$ on the design variables produces displacements in the order of 10^{-3} m, roughly two orders of magnitude smaller than the duct's diameter. Such a small step-length is chosen for two reasons. Firstly, it facilitates autonomous primal convergence at

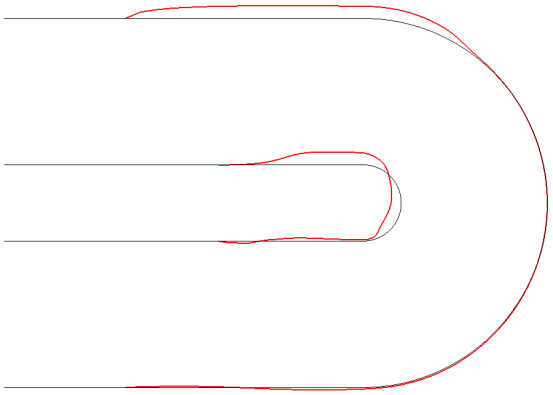


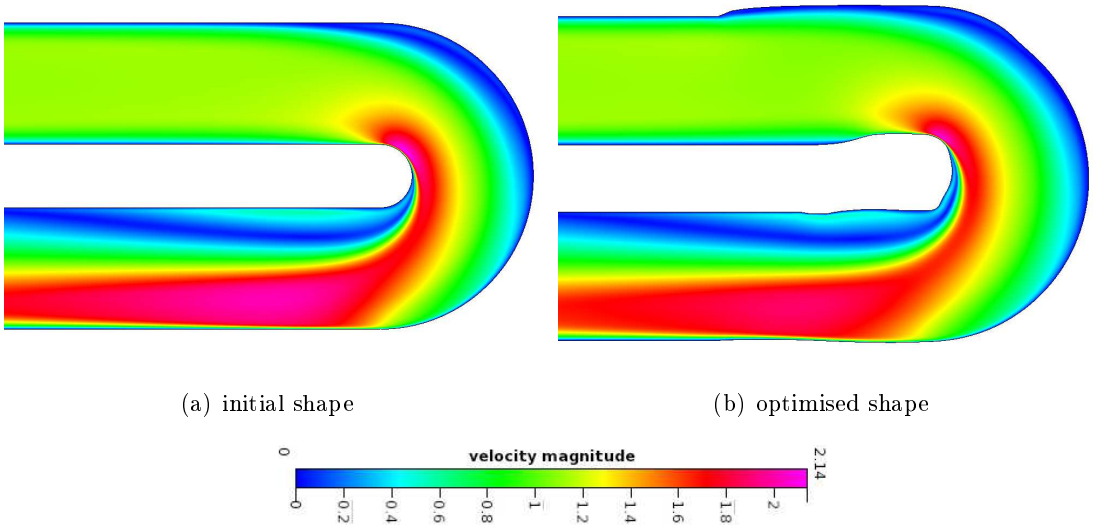
Figure 7.7: U-bend: original shape (black) and optimised (red) of the bend.

each cycle as long as the flow field from the previous cycle is used as a starting solution - high-Re cases in general have shown difficulties converging from a poor initial guess, regardless of the solution algorithm, and do require some manual intervention such as e.g. an initial set of iterations with no turbulence model and first-order-only operators. Secondly, a small step-length guarantees that the error on the gradient due to the frozen turbulence assumption remains negligible, thus ensuring that the whole optimisation process moves in the right direction. The drawback is a limited improvement on the cost function between cycles and thus a high number of cycles required to achieve a significant reduction of J.

Figure 7.6(a) shows the convergence history of the normalised cost function J/J_0 over 118 cycles, at the end of which a 14.24% improvement on J is obtained. Compared to the S-bend, results for the U-bend are not as straightforward to interpret mathematically: J does steadily decrease from one iteration to the next, but the trend is rather irregular, featuring traits of linear decrease with varying slope as well as sudden jumps in the value of J. It should also be mentioned that, for this test case, it was not possible to fully automate the optimisation process: manual intervention was required at certain iterations, namely to temporarily reduce the step-length in order to ease convergence of the morpher, occasionally hindered by too noisy gradients (possibly caused by localised instabilities in the primal field). However, such interventions do not necessarily correspond to visible effects in the convergence shown in Figure 7.6, such as jumps, which are instead caused by sudden peaks in gradient magnitude.

The gradient magnitude history (Figure 7.6(b)), on the other hand, confirms the linear-like descent of J in the sense that it does not present a definite decreasing trend, but rather oscillates around a certain range of values. This points to the fact that the optimiser is still far from approaching a local minimum of the cost function and therefore,

in theory, there is room for a much more significant reduction of J; the limitation comes unfortunately from the RMMM tool, which fails to morph past the 118-th iteration without producing degenerate/negative volume cells.



(a) initial shape (b) optimised shape

Figure 7.8: U-bend: velocity magnitude (m/s).

Figure 7.7 shows the optimised shape: similar to the S-bend case, the process is attempting to suppress or reduce secondary/recirculating flows, which for the U-bend are mostly found along the second half of the bend and caused by flow separation at the inner wall, visible in Figure 7.8(a). The optimised shape presented here is in good agreement with previous literature, in particular Coletti et al. [59] and Verstraete et al. [232]: the initially circular bend is widened and turned into a horseshoe shape, particularly evident on the trait upwind of the bend (top in Figure 7.7). However, no significant differences are visible between the original and optimised flow field, other than a slight reduction of the separated flow region along the internal side of the bend and a moderate reduction of the average flow velocity, as highlighted by the velocity magnitude contour (Figure 7.8). A bump-like feature is visible on the inlet leg at the location where the patch of fixed nodes is connected to that of handle nodes; this is due to the RMMM tool which is not yet capable of enforcing any smoothness constraint in this type of situation. The same area is also found to be responsible for the aforementioned creation of negative volume cells, i.e. the reason why the optimiser had to be stopped. In conclusion: the results on the U-bend are positive but, taking into account the difficulties encountered, cannot be considered as fully satisfactory without further ameliorations to the solver, morpher and optimiser.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

The limitations of the discrete adjoint method, and specifically the instability arising from a non-converging primal, were the motivating factor of the main body of the present work, which primarily focused on improving the primal CFD solver in terms of spatial discretisation schemes in order to indirectly alleviate the adjoint robustness issues. A new discretisation method was put forward and named Mixed Hybrid Finite Volumes (MHFV). Arguably the most innovative aspect of MHFV with respect to classical Finite Volumes and Finite Elements is the treatment of diffusion terms: the basic methodology for discretising the (anisotropic) pure diffusion equation is borrowed from the family of novel methods known as Mixed Virtual Elements, which leads to the formulation of a second-order accurate, stable diffusion operator on generic polyhedral meshes. The MHFV method is based on *mimetic* discrete gradient and divergence operators, i.e. operators that satisfy the Gauss-Green formula at the discrete level. The scheme is fully implicit and it maintains its convergence properties even on grids typically considered unsuitable for classical FV, such as those containing highly non-orthogonal, skewed or non-convex elements; this is a considerable further advantage in the context of shape optimisation. A parallelism was identified between MHFV and classical FV Non-Orthogonal Correctors, and exploited to derive a novel weighting system for the MHFV stabilisation term. The MHFV scheme for pure anisotropic diffusion was validated on manufactured test cases over sequences of highly distorted grids.

The MHFV scheme was then extended to cater for scalar convection-diffusion-reaction problems. Following suggestions from literature, multiple convective schemes - some resembling FV, others FE - were coded within a unified framework: Hybrid Centred,

Mixed Centred, Hybrid First-Order Upwind, Hybrid θ-Scheme, and Hybrid Second-Order Upwind. It was shown through estimates that, as in classical FV, centred schemes suffer from stability issues which, for high-Peclet cases, require a level of mesh refinement which in practice cannot be afforded. First-order upwinding, on the other hand, is unconditionally stable at the cost of degrading the accuracy of the scheme. A range of semi-empirical stabilisation techniques were proposed with the aim of stabilising second-order schemes while maintaining their nominal accuracy: three of them - SUPG, flux limiters and WLSQR - were adapted to MHFV from FV and/or FE; a fourth one, ULSQR, is a novelty of this work. The MHFV convection-diffusion-reaction scheme, in all of its variants, was successfully validated for a range of Peclet numbers and grids on manufactured test cases, including one (the Smith-Hutton test) specifically designed to test stabilisation techniques.

The MHFV incompressible, steady-state Navier-Stokes scheme was then derived. As suggested by previous literature focused on similar attempts, the MHFV convection-diffusion operator was used to discretise the Picard-linearised momentum equation in each direction with the addition of a pressure term to the convective-diffusive flux of velocity components. An extension of the pressure scheme to second-order accuracy is a further innovation brought about in the present work. The scheme was tested first in terms of h-convergence on a manufactured solution with particular emphasis on grid independence, and then successfully validated on the 2D lid-driven cavity benchmark test case over a range of Reynolds numbers from 10^2 to 10^4 and a highly distorted mesh.

A few different solution algorithms for the Navier-Stokes were proposed: besides the popular SIMPLE, which was adapted to MHFV in its SIMPLEC variant, the Block-Coupled (BCPL) and Augmented Lagrangian (AL) approaches were also explored on the grounds of being able to tackle the larger, more complex linear systems arising in these strategies thanks to improvements in modern computational power and linear solver capabilities. Numerical experiments proved SIMPLEC to be rather inefficient compared to both BCPL and AL, not only in terms of iteration count but also in how algorithm performance is affected by mesh type and size as well as the physics of the problem. BCPL and AL were found to perform comparably and reduce the iteration count by a whole order of magnitude from SIMPLEC, but difficulties in solving the near-singular penalised systems arising with the AL approach did show in certain configurations. The improved behaviour of MHFV in terms of convergence to steady-state, which positively affects stability of the corresponding discrete adjoint, was demonstrated by comparison with a standard FV solver.

As a last step, the discrete adjoint MHFV Navier-Stokes system was considered. The reverse assembly approach - performed here via Finite Differencing (FD) - was identified

as a suitable way of assembling the Jacobian matrix as well as all other derivative terms required for adjoint sensitivity computation. Colouring algorithms were used to maintain the required number of FD evaluations independent of the problem size. Adjoint-adapted versions of the primal solution algorithms SIMPLEC and AL were proposed. A Velocity-Coupled algorithm (VCPL) - a version of SIMPLEC with fully implicit treatment of the Adjoint Transpose Convection (ATC) term - was also implemented. Numerical tests showed that the performance and behaviour of adjoint SIMPLEC and AL are identical to those of their primal counterparts, with AL outperforming the former in terms of iteration count by an order of magnitude. The improvements brought about by the VCPL scheme were only visible at high Reynolds numbers. Finally, a Rigid Motion Mesh Morphing tool with node-based Soft Handle Parametrisation was included in the differentiation chain, and the adjoint-produced gradient was validated by comparison with FD values on the 3D S-bend benchmark case. The S-bend and the U-bend test cases (both internal flow cases, the first laminar, the second turbulent) were then used to illustrate the results of a full shape optimisation process, achieving a reduction in the total power loss of 21.2% and 14.24% respectively. While the process was fully automated for the S-bend and resulted in a smooth decrease in objective function and corresponding gradient magnitude, the U-bend was more irregular and required manual intervention in some places, notably to temporarily adjust the steepest descent step-length factor in order to get past iterations with seemingly rogue or noisy gradient components.

8.2 Future work

Several limitations were encountered throughout the research presented here. They serve to suggest future axes of development for primal and adjoint MHFV towards full-scale industrial applications.

- *Code parallelisation*: the prototype MHFV solver developed for this thesis was written within the framework of ESI's in-house *Fortran Template Library* (FTL). FTL is a powerful FORTRAN library which extends the key concepts of C++ - such as object-oriented programming and templating capabilities - to the FORTRAN language. FTL was originally developed specifically for ESI's i-Adjoint tool, but its potential applications extend beyond adjoint computation. For instance, it contains a number of basic classes allowing to instantiate objects such as topologies, matrix graphs, block-sparse matrices etc. many of which were used for implementing the MHFV solver. FTL, however, is yet to be parallelised (in either DMP or SMP), and as a consequence it was not possible in the present work to tackle test cases of industrial size without incurring in prohibitive CPU costs, which would have

jeopardised the overall research progress.

- *Linear solvers*: linear systems were solved with the in-house solver *DFGM*, which can be used either as an iterative solver (GMRES) preconditioned via traditional ILU or a two-level additive DDM Schwarz method, or as a direct solver. It was found however that DFGM systematically failed to precondition systems with non-homogeneous block-sparse matrices, i.e. matrices containing blocks of different sizes. This is notably the case for the full Oseen system arising from the MHFV Navier-Stokes operator, where the underlying hybrid topology leads to a distinction between blocks linked to faces, carrying d degrees of freedom each (one per velocity component), and blocks linked to cells, carrying one degree of freedom each (pressure). As a consequence, direct solvers had to be used wherever a hybrid Oseen-type system arose, which was notably the case for the BCPL solution algorithm. The usage of a direct solver is justifiable if the goal is a mere demonstration of the efficiency of the outer algorithm, but it is too costly to be considered for full-sized industrial cases. Similar issues were also encountered in the preconditioning of the penalised momentum equations arising in the AL approach, although in this case the problem was expected: previous literature does mention the appearance of near-indefinite systems, and research towards a solution or workaround is ongoing. Hence the two alternative solution algorithms as presented in this thesis remain at a "proof of concept" level.

- *Analysis and validation*: it was shown how numerical tests on the schemes and strategies presented produce encouraging numerical results; for completeness, however, it would be interesting to conduct a more thorough mathematical analysis. For example, one may attempt to derive stability and error estimates for the ULSQR convective scheme, or the PRS2 second-order pressure gradient operator. A similar observation holds for the Spalart-Allmaras implementation presented in Appendix A, for which there is no a priori guarantee of convergence. A more exhaustive campaign of numerical experiments, including a wider range of geometries and problem physics, would also be highly beneficial to further prove the validity of the MHFV scheme and the several options described in this work.

- *Mesh morpher*: the Rigid Motion Mesh Morphing tool, combined with the Soft Handle CAD-Free Parametrisation, was chosen for the practical reason of being implemented within the same FTL framework, which greatly facilitated the coupling with the MHFV solver. The morpher however is still in its development phase and currently suffers from limitations, such as the inability to cope with too large deformations without sub-cycling or the lack of smoothness at the joints between fixed and moving patches of nodes. Some of these issues are currently being inves-

tigated (in the context of the EC-funded IODA project) and positive results have begun to emerge, but not timely enough to be included in the present thesis.

- *Further developments*: practical time constraints dictated that only a few "basic" options could be included in the MHFV Navier-Stokes solver: three types of boundary conditions (inlet, outlet and wall), and one turbulence model (Spalart-Allmaras). As a consequence, while the potential of the scheme was demonstrated on academic test cases, it was not possible to test it on more complex industrial cases which often require more sophisticated models. However, the aforementioned modularity of the prototype will simplify the future task of developing new features, extend the (primal and adjoint) solver to unsteady and/or compressible flows, as well as the implementation of a higher than second-order MHFV scheme.

Appendix A

The Spalart-Allmaras Turbulence Model

The Spalart-Allmaras (SA) turbulence model [216] is a one-equation model which requires solving an additional convection-diffusion-reaction problem for a viscosity-like variable ν_{sa}, subsequently used to compute a *turbulent eddy viscosity* ν_t which, when added to the fluid's physical kinematic viscosity, produces a macroscopic simulation of the effects of turbulence. The model was shown to work particularly well for applications involving wall-bounded flows with boundary layers subjected to adverse pressure gradients. The original SA formulation, presented below, belongs to the so-called family of *low-Reynolds models*, meaning that the dimensionless wall distance y^+ of grid points closest to boundary walls must be in the order of 1; for a more insightful explanation, see e.g. [53, 238].

In its basic, steady-state version, the SA equation is defined as

$$\begin{aligned} & \nabla \cdot \left(-\frac{(\nu + \nu_t)}{\sigma} \nabla \nu_{sa} + \vec{U} \nu_{sa} \right) \\ + \quad & \left(c_{w1} f_w + \frac{c_{b1}}{\kappa^2} f_{t2} \right) \left(\frac{\nu_{sa}}{d} \right)^2 - c_{b1} \left(1 - f_{t2} \right) S \nu_{sa} \\ - \quad & \frac{c_{b2}}{\sigma} \nabla \nu_{sa} \cdot \nabla \nu_{sa} \qquad\qquad = 0 \end{aligned} \tag{A.1}$$

with the eddy viscosity computed as

$$\nu_t = f_{v1} \nu_{sa} \tag{A.2}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \qquad \text{and} \qquad \chi = \frac{\nu_{sa}}{\nu} \ . \tag{A.3}$$

Additional definitions are given by

$$S = \Omega + \frac{\nu_{sa}}{\kappa^2 d^2} f_{v2} \tag{A.4}$$

where Ω is the magnitude of the vorticity and d is the distance from the field point to the nearest wall, and

$$\begin{array}{lll} f_{v2} = 1 - \dfrac{\chi}{1 + \chi f_{v1}} & ; & f_w = g\left(\dfrac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right)^{1/6} \quad ; \\ g = r + c_{w2}\left(r^6 - r\right) & ; & r = \min\left(\dfrac{\nu_{sa}}{S\kappa^2 d^2}, 10\right) \quad ; \\ f_{t2} = c_{t3}^{-c_{t4}\chi^2} & & . \end{array} \tag{A.5}$$

Finally, the model constants are

$$\begin{array}{llllll} c_{b1} = 0.1355 & ; & \sigma = 2/3 & ; & c_{b2} = 0.622 & ; \quad \kappa = 0.41 \quad ; \\ c_{w2} = 0.3 & ; & c_{w3} = 2 & ; & c_{v1} = 7.1 & ; \quad c_{t3} = 1.2 \quad ; \\ c_{t4} = 0.5 & ; & c_{w1} = \dfrac{c_{b1}}{\kappa^2} + \dfrac{1 + c_{b2}}{\sigma} & . & & \end{array} \tag{A.6}$$

There is a clear resemblance between the SA equation (A.1) and a generic transport problem. More specifically: the first term is a simple convection-diffusion operator with convecting field $\vec{U}$ and (isotropic) diffusivity $\frac{(\nu+\nu_t)}{\sigma}$; the second and third terms - often referred to as *production-destruction* in turbulence modeling jargon - can be rewritten as a non-linear reaction term, i.e. one with a solution-dependent reaction coefficient η:

$$\eta\left(\nu_{sa}\right) = \left(c_{w1} f_w + \frac{c_{b1}}{\kappa^2} f_{t2}\right) \frac{\nu_{sa}}{d^2} - c_{b1}\left(1 - f_{t2}\right) S \ . \tag{A.7}$$

Both these terms are straightforwardly discretised with the MHFV operators described in Chapter 4. At a given CFD iteration, a cell-averaged vorticity magnitude Ω_C is computed based on a Gauss gradient of the current MHFV hybrid velocity field $\widetilde{\mathbf{u}}$, while an approximated nearest-wall distance field d can be obtained via any of the methods described in [225] - which in turn require solving a diffusion or convection-diffusion problem, once and for all at the beginning of the simulation. The cell-averaged field ν_{sa} from the previous iteration is then used to compute cell-based values of all model functions in (A.4) and (A.5), allowing to compute a (linearised) cell-averaged reaction coefficient in the form (A.7). As for the diffusive and convective coefficients, they are linearised by using respectively the current eddy viscosity field ν_t and convecting field $\mathbf{U}$.

The fourth term in (A.1): $(-\frac{c_{b2}}{\sigma}\nabla\nu_{sa} \cdot \nabla\nu_{sa})$, on the other hand, is a form not yet encountered and thus requires further attention. To that end, it is useful to consider a

convection-diffusion-reaction problem of the form:

$$\nabla \cdot \left(-\mathbb{K}\nabla\phi + \vec{U}\phi\right) + \eta\phi + \underbrace{\vec{W}\cdot\nabla\phi}_{\text{additional term}} = f \quad \text{in } \Omega \tag{A.8}$$

where $\vec{W}$ is a generic vector field and all other symbols carry their usual connotation from Chapter 4. Applying the Gauss-Green formula to the additional term in (A.8) over a cell C yields

$$\int_C \vec{W}\cdot\nabla\phi\, dV = -\int_C \phi\nabla\cdot\vec{W}\, dV + \int_{\partial C}\phi\vec{W}\cdot\vec{n}\, dS \tag{A.9}$$

which justifies the following mixed formulation of (A.8):

$$\begin{cases} \vec{V} = -\mathbb{K}\nabla\phi + \left(\vec{U}+\vec{W}\right)\phi \\ \nabla\cdot\vec{V} + \left(\eta - \nabla\cdot\vec{W}\right)\phi = f \end{cases} \quad \text{in } \Omega\ . \tag{A.10}$$

Problem (A.10) is now in a form that, in practice, can be discretised with the already existing MHFV operators: let $\mathbf{W}$ be the de Rham map of $\vec{W}$ in X^h; it is then sufficient to replace the MHFV convecting field $\mathbf{U}$ with a modified one:

$$\widehat{\mathbf{U}} = \mathbf{U} + \mathbf{W} \tag{A.11}$$

and similarly modify the reaction coefficient in Q^h:

$$\widehat{\boldsymbol{\eta}} = \boldsymbol{\eta} - \mathcal{D}\mathbf{W}\ . \tag{A.12}$$

This procedure allows to discretise the fourth term in (A.1), namely by setting $\vec{W} = -\frac{c_{b2}}{\sigma}\nabla\nu_{sa}$. Hence for the SA model, the MHFV field $\mathbf{W}$ should correspond to a diffusive flux of ν_{sa} with constant diffusivity $\frac{c_{b2}}{\sigma}$. This is obtained by using a MHFV-like flux operator, i.e. by defining $\mathbf{W}$ as

$$(\mathbf{W})_{\partial C} = \mathbb{M}^{-1}_{sa,C}\left(\nu_{sa,C} - \nu_{sa,F}\right)_{\partial C} \tag{A.13}$$

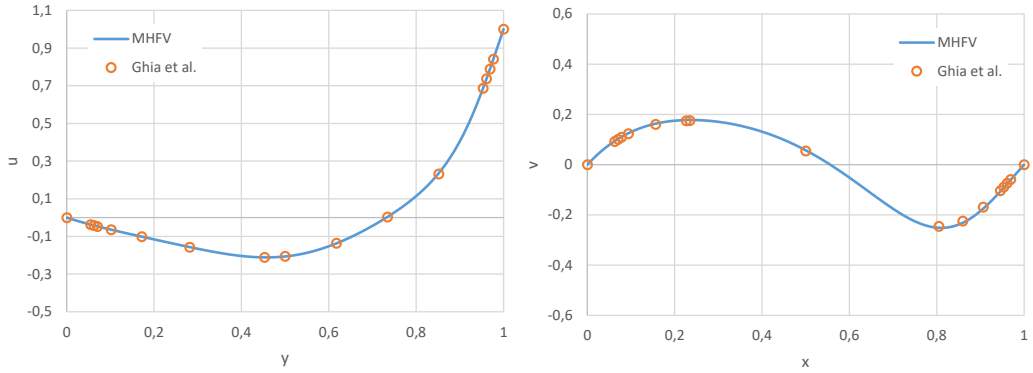
where $\mathbb{M}_{sa,C}$ corresponds to the local scalar product matrix from Chapter 3, computed by replacing the diffusivity tensor $\mathbb{K}$ with the (isotropic) constant $\frac{c_{b2}}{\sigma}$. Since this yelds two fluxes per face, W_{FC+} and W_{FC-}, which are in general different, the average value is taken. Of course (A.13) implies a solution-dependent $\mathbf{W}$ and thus, by virtue of (A.11), a non-linear convective term; the equation is thus linearised at each CFD iteration by computing (A.13) from previous values of ν_{sa}.

Appendix B

Under-Resolved Lid-Driven Cavity

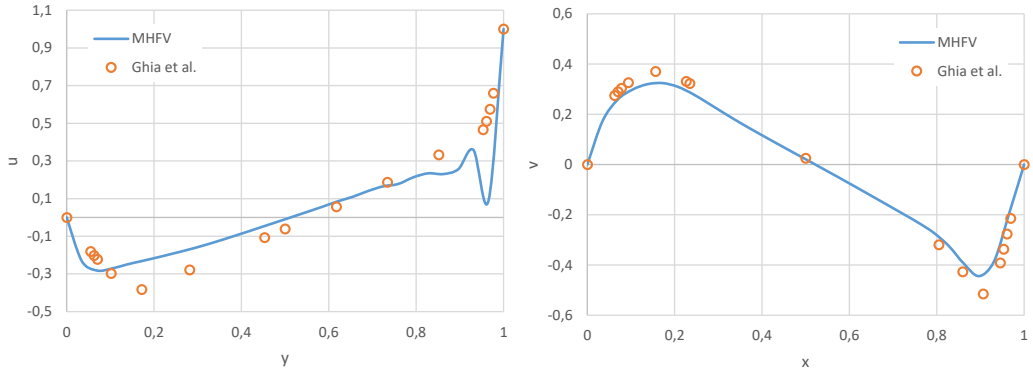
Figures B.1 through B.3 report MHFV results for the under-resolved lid-driven cavity test case for $Re = 10^2$, 10^3 and 10^4 respectively. The solver setup (second-order for all variables) and mesh type (quadrilateral distorted) are the same as described in Section 5.3.2, but the mesh is coarser: it is generated by distorting a 30×30 Cartesian mesh, leading to an averaged cell-to-cell distance $h_{avg} \approx 3.51\ \mathrm{E}^{-2}$, approximately four times larger than that used for the reference results.

The scheme is stable for all Re values, although some oscillations can be observed in more convection-dominated regimes; these can be attributed to the ULSQR stabilising strategy (Section 4.2.5) which bounds such oscillations but does not suppress them entirely. At $Re = 10^2$ the results are in perfect agreement with the reference values despite the under-resolution. For higher Re values the error remains within acceptable bounds across most of the domain, however the scheme fails to capture with sufficient accuracy steep gradient zones, notably close to boundaries due to the under-resolved boundary layer.



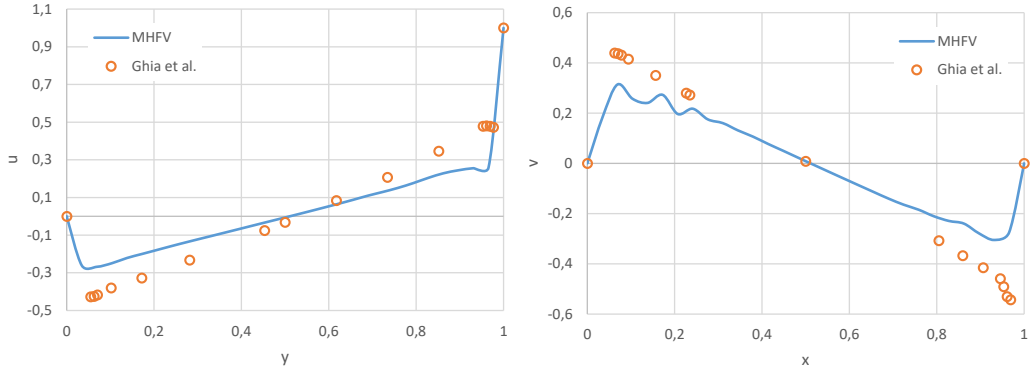
(a) u along y-axis (b) v along x-axis

Figure B.1: Under-resolved lid-driven cavity, $Re = 10^2$: results comparison.



(a) u along y-axis (b) v along x-axis

Figure B.2: Under-resolved lid-driven cavity, $Re = 10^3$: results comparison.



(a) u along y-axis (b) v along x-axis

Figure B.3: Under-resolved lid-driven cavity, $Re = 10^4$: results comparison.

Appendix C

Author's Publications

Conference papers

1. **M. Oriani** and G. Pierrot , "Alternative solution algorithms for primal and adjoint incompressible Navier-Stokes," *ECCOMAS Congress*, Crete Island, Greece, June 2016.

2. **M. Oriani** and G. Pierrot , "A Mixed Hybrid Finite Volume Scheme for incompressible Navier-Stokes," *NAFEMS World Congress*, San Diego, California, USA, June 2015.

3. **M. Oriani** and G. Pierrot , "Alleviating adjoint solver robustness issues via mimetic CFD discretization schemes," *OPT-i Conference*, Kos Island, Greece, June 2014.

Bibliography

[1] I. Aavatsmark. An introduction to multipoint flux approximations for quadrilateral grids. *Computational Geosciences*, 6(3):405–432, 2002.

[2] R. Abgrall. On Essentially Non-Oscillatory schemes on unstructured meshes: Analysis and implementation. *Journal of Computational Physics*, 114(1):45–58, 1994.

[3] D. Adak and E. Natarajan. A unified analysis of nonconforming Virtual Element Methods for convection diffusion reaction problem. *ArXiv e-print 1601.01077*, 2016.

[4] S. Akbarzadeh, Y. Wang, and J-D. Müller. Fixed point discrete adjoint of SIMPLE-like solvers. *AIAA Paper 2015-2750*, 2015.

[5] W.K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers and Fluids*, 28(4):443–480, 1999.

[6] M.I. Andreou, S.E. Nikoletseas, and P.G. Spirakis. Algorithms and experiments on colouring squares of planar graphs. *Second International Workshop on Experimental and Efficient Algorithms, WEA 2003, Ascona, Switzerland*, pages 15–32, 2003.

[7] P. Antonietti, L. Beirão da Veiga, N. Bigoni, and M. Verani. Mimetic Finite Differences for non-linear and control problems. *Mathematical Models and Methods in Applied Sciences*, 24(8):1457–1493, 2014.

[8] P. Antonietti, N. Bigoni, and M. Verani. Mimetic Finite Difference method for shape optimization problems. *Lecture Notes in Computational Science and Engineering*, 103:125–132, 2015.

[9] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. I: Discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.

[10] B.M. Averick, J.J. Moré, C.H. Bischof, A. Carle, and A. Griewank. Computing large sparse Jacobian matrices using automatic differentiation. *SIAM Journal on Scientific Computing*, 15(2):285–294, 1994.

[11] T.J. Barth and D.C. Jespersen. The design and application of upwind schemes on unstructured meshes. *AIAA Paper 89-0366*, 1989.

[12] M. Bartholomew-Biggs. *Nonlinear Optimization with Engineering Applications.* Springer, 2005.

[13] P. Batten, C. Lambert, and D.M. Causon. Positively conservative high-resolution schemes for unstructured elements. *International Journal for Numerical Methods in Engineering*, 39:1821–1838, 1996.

[14] O. Baysal and M.E. Eleshaky. Aerodynamic sensitivity analysis methods for the compressible Euler equations. *Journal of Fluids Engineering*, 113(4):681–688, 1991.

[15] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L.D. Marini, and A. Russo. Basic principles of Virtual Element Methods. *Mathematical Models and Methods in Applied Sciences*, 23(1):199–214, 2013.

[16] L. Beirão da Veiga, F. Brezzi, L.D. Marini, and A. Russo. The hitchhiker's guide to the Virtual Element Method. *Mathematical Models and Methods in Applied Sciences*, 24(8):1541–1573, 2014.

[17] L. Beirão da Veiga, F. Brezzi, L.D. Marini, and A. Russo. Mixed Virtual Element Methods for general second order elliptic problems on polygonal meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 50(1):727–747, 2016.

[18] L. Beirão da Veiga, J. Droniou, and G. Manzini. A unified approach to handle convection terms in Finite Volumes and Mimetic Discretization Methods for elliptic problems. *IMA Journal of Numerical Analysis*, 31(4):1357–1401, 2011.

[19] L. Beirão da Veiga, K. Lipnikov, and G. Manzini. Arbitrary-order nodal mimetic discretizations of elliptic problems on polygonal meshes. *SIAM Journal on Numerical Analysis*, 49(5):1737–1760, 2011.

[20] L. Beirão da Veiga, K. Lipnikov, and G. Manzini. *The Mimetic Finite Difference method for Elliptic Problems.* Springer, 2014.

[21] L. Beirão da Veiga, C. Lovadina, and G. Vacca. Divergence free virtual elements for the Stokes problem on polygonal meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 51(2):509–535, 2017.

[22] L. Beirão da Veiga, C. Lovadina, and G. Vacca. Virtual Elements for the Navier-Stokes problem on polygonal meshes. *ArXiv e-print 1703.00437*, 2017.

[23] L. Beirão da Veiga and G. Manzini. A higher-order formulation of the Mimetic Finite Difference method. *SIAM Journal on Scientific Computing*, 31(1):732–760, 2008.

[24] M.F. Benedetto, S. Berrone, A. Borio, S. Pieraccini, and S. Scialò. Order preserving SUPG stabilization for the Virtual Element formulation of advection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 311:18–40, 2016.

[25] M. Benzi. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[26] M. Benzi and X. Guo. A dimensional split preconditioner for Stokes and linearized Navier–Stokes equations. *Applied Numerical Mathematics*, 61(1):66–76, 2011.

[27] M. Benzi, M. Ng, Q. Niu, and Z. Wang. A Relaxed Dimensional Factorization preconditioner for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 230(16):6185–6202, 2011.

[28] M. Benzi and M.A. Olshanskii. An Augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing*, 28(6):2095–2113, 2006.

[29] M. Benzi, M.A. Olshanskii, and Z. Wang. Modified Augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 66(4):486–508, 2011.

[30] M. Benzi and Z. Wang. Analysis of Augmented Lagrangian-based preconditioners for the steady incompressible Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 33(5):2761–2784, 2011.

[31] M. Berger and M.J. Aftosmis. Analysis of slope limiters on irregular grids. *AIAA Paper 2005-0490*, 2005.

[32] S.A. Berger, L. Talbot, and L.S. Yao. Flow in curved pipes. *Annual Review of Fluid Mechanics*, 15(1):461–512, 1983.

[33] C.H. Bischof, P.M. Khademi, A. Bouaricha, and A. Carle. Effcient computation of gradients and jacobians by transparent exploitation of sparsity in automatic differentiation. *Optimization Methods and Software*, 7:1–39, 1996.

[34] F. Blom. Considerations on the Spring Analogy. *International Journal for Numerical Methods in Fluids*, 32(6):647–668, 2000.

[35] J. Bonelle. *Compatible Discrete Operator schemes on polyhedral meshes for elliptic and Stokes equations.* PhD thesis, École doctorale MSTIC, 2014.

[36] R. Bott and L.W. Tu. *Differential Forms in Algebraic Topology*. Springer-Verlag, Berlin, New York, 1982.

[37] J. Brezillon and R.P. Dwight. Aerodynamic shape optimization using the discrete adjoint of the Navier-Stokes equations: Applications toward complex 3d configutations. *KATnet II Conference on Key Aerodynamic Technologies, Bremen, Germany*, 2009.

[38] F. Brezzi, A. Buffa, and G. Manzini. Mimetic scalar products of discrete differential forms. *Journal of Computational Physics*, 257:1228–1259, 2014.

[39] F. Brezzi, R.S. Falk, and L.D. Marini. Basic principles of Mixed Virtual Element method. *Mathematical Modelling and Analysis*, 48(4):1227–1240, 2014.

[40] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer-Verlag, New York, 1991.

[41] F. Brezzi, K. Lipnikov, and M. Shashkov. Convergence of the Mimetic Finite Difference method for diffusion problems on polyhedral meshes. *SIAM Journal on Numerical Analysis*, 43(5):1872–1896, 2005.

[42] F. Brezzi, K. Lipnikov, and M. Shashkov. A family of Mimetic Finite Difference methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 15(10):1533–1551, 2005.

[43] F. Brezzi, L.D. Marini, S. Micheletti, P. Pietra, and R. Sacco. Stability and error analysis of mixed finite-volume methods for advection dominated problems. *Computers & Mathematics with Applications*, 51(5):681–696, 2006.

[44] D. Brélaz. New methods to color the vertices of a graph. *Communication of ACM*, 22(4):251–256, 1979.

[45] N. Brooks and T.J.R. 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*, 32(1):199–259, 1982.

[46] E. Burman and A. Linke. Stabilized finite element schemes for incompressible flow using Scott–Vogelius elements. *Applied Numerical Mathematics*, 58(11):1704–1719, 2008.

[47] M.S. Campobasso and M.B. Giles. Stabilization of a linear flow solver for turbomachinery aeroelasticity by means of the recursive projection method. *AIAA Journal*, 42(9):1765–1774, 2004.

[48] A. Cangiani, V. Gyra, and G. Manzini. The non conforming Virtual Element Method for the Stokes equations. *SIAM Journal on Numerical Analysis*, 54(6):3411–3435, 2016.

[49] A. Cangiani and G. Manzini. Flux reconstruction and solution post-processing in Mimetic Finite Difference methods. *Computer Methods in Applied Mechanics and Engineering*, 197:933–945, 2008.

[50] A. Cangiani, G. Manzini, and A. Russo. Convergence analysis of the Mimetic Finite Difference method for elliptic problems. *SIAM Journal on Numerical Analysis*, 47(4):2612–2637, 2009.

[51] S.M. Chang, J.A.C. Humphrey, and A. Modavi. Turbulent flow in a strongly curved U-bend and downstream tangent of square cross-sections. *Physicochemical Hydrodynamics*, 4:243–269, 1983.

[52] S.C. Cheah, H. Iacovides, D.C. Jackson, H.H. Ji, and B.E. Launder. LDA investigation of the flow development through rotating U-ducts. *Journal of Turbomachinery*, 118:590–596, 1996.

[53] C.J. Chen and S. Jaw. *Fundamentals of turbulence modeling.* Taylor & Francis, 1998.

[54] F. Christakopoulos, D. Jones, and J-D. Müller. Pseudo-timestepping and verification for automatic differentiation derived CFD codes. *Computers & Fluids*, 46(1):174–179, 2011.

[55] B. Christianson. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3:311–326, 1994.

[56] B. Christianson. Reverse accumulation and implicit functions. *Optimization Methods and Software*, 9:307–322, 1998.

[57] B. Cockburn, G. Kanschat, and D. Schötzau. A locally conservative LDG method for the incompressible Navier-Stokes equations. *Mathematics of Computation*, 74:1067–1095, 2005.

[58] T.F. Coleman and J.J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):187–209, 1983.

[59] F. Coletti, T. Verstraete, J. Bulle, T. Van der Wielen, N. Van den Berge, , and T. Arts. Optimization of a U-bend for minimal pressure loss in internal cooling channels - part II: Experimental validation. *ASME Journal of engineering for Gas Turbines and Power*, 135(5), 2013.

[60] Y. Coudière, J.P. Vila, and P. Villedieu. Convergence rate of a Finite Volume scheme for a two dimensional convection-diffusion problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 33(3):493–516, 1999.

[61] F. Courty, A. Dervieux, B. Koobus, and L. Hascoët. Reverse Automatic Differentiation for optimum design: from adjoint state assembly to gradient computation. *Optimization Methods and Software*, 18(5):615–627, 2003.

[62] A.R. Curtis, M.J. Powell, and J.K. Reid. On the estimation of sparse Jacobian matrices. *IMA Journal of Applied Mathematics*, 13(1):117–119, 1974.

[63] P. Cusdin. *Automatic sensitivity code for Computational Fluid Dynamics*. PhD thesis, School of Aeronautical Engineering, Queen's University Belfast, 2005.

[64] P. Cusdin and J-D. Müller. Deriving linear and adjoint codes for CFD using automatic differentiation. *School of Aeronautical Engineering, Queen's University Belfast*, 2003.

[65] M. Darwish and F. Moukalled. A fully coupled Navier-Stokes solver for fluid flow at all speeds. *Numerical Heat Transfer*, 65(5):410–444, 2014.

[66] M. Darwish, I. Sraj, and F. Moukalled. A coupled finite volume solver for the solution of incompressible flows on unstructured grids. *Journal of Computational Physics*, 228(1):180–201, 2009.

[67] A.C. De Niet and F.W. Wubs. Two preconditioners for saddle point problems in fluid flows. *International Journal for Numerical Methods in Fluids*, 54(4):355–377, 2007.

[68] W.R. Dean. The stream-line motion of fluid in a curved pipe. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 5(7):673–695, 1928.

[69] P. Deuring. Eigenvalue bounds for the Schur complement with a pressure convection–diffusion preconditioner in incompressible flow computations. *Journal of Computational and Applied Mathematics*, 228(1):444–457, 2009.

[70] J. Droniou. Remarks on discretizations of convection terms in Hybrid Mimetic Mixed methods. *Networks and Heterogeneous Media*, 5(3):545–563, 2010.

[71] J. Droniou and R. Eymard. A Mixed Finite Volume scheme for anisotropic diffusion problems on any grid. *Numerische Mathematik*, 105(1):35–71, 2006.

[72] J. Droniou and R. Eymard. Study of the Mixed Finite Volume method for Stokes and Navier-Stokes equations. *Numerical Methods for Partial Differential Equations*, 25(1):137–171, 2009.

[73] J. Droniou, R. Eymard, T. Galloüet, and R. Herbin. A unified approach to Mimetic Finite Difference, Hybrid Finite Volume and Mixed Finite Volume methods. *Mathematical Models and Methods in Applied Sciences*, 20(2):265–295, 2010.

[74] R.P. Dwight and J. Brezillon. Effect of various approximations of the discrete adjoint on gradient-based optimization. *AIAA Paper 2006-0690*, 2006.

[75] R.P. Dwight and J. Brezillon. Efficient and robust algorithms for solution of the adjoint compressible Navier–Stokes equations with applications. *International Journal for Numerical Methods in Fluids*, 60(4):365–389, 2009.

[76] M. Gad El-Hak, A. Pollard, and J. Bonnet. *Flow Control: Fundamentals and Practices.* Springer, 1998.

[77] G. Eleftheriou and G. Pierrot. Rigid Motion Mesh Morpher: a novel approach for mesh deformation. *OPT-I: International Conference on Engineering and Applied Sciences Optimization, Kos, Greece*, 2014.

[78] H.C. Elman. Preconditioning for the steady-state Navier–Stokes equations with low viscosity. *SIAM Journal of Scientific Computing*, 20(4):1299–1316, 1999.

[79] H.C. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. Block preconditioners based on approximate commutators. *SIAM Journal of Scientific Computing*, 27(5):1651–1668, 2006.

[80] H.C. Elman, V.E. Howle, J. Shadid, D. Silvester, and R. Tuminaro. Least Squares preconditioners for stabilized discretizations of the Navier–Stokes equations. *SIAM Journal of Scientific Computing*, 30(1):290–311, 2007.

[81] K. Elsayed, J.D. Carrilho Miranda, and C. Lacor. Minimization of the pressure loss in internal cooling channels using the adjoint method. *First Aviation Engineering Innovations Conference, Luxor, Egypt*, 2015.

[82] R. Eymard, T. Gallouet, and R. Herbin. Discretization of heterogeneous and anisotropic diffusion problems on general nonconforming meshes SUSHI: a scheme using stabilization and hybrid interfaces. *IMA Journal of Numerical Analysis*, 30(4):1009–1043, 2010.

[83] R. Eymard, R. Herbin, and J.C. Latché. Convergence analysis of a colocated finite volume scheme for the incompressible Navier–Stokes equations on general 2D or 3D meshes. *SIAM Journal on Numerical Analysis*, 45(1):1–36, 2007.

[84] C. Farhat, C. Degant, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163(1):231–245, 1998.

[85] P.E. Farrell, D.A. Ham, S.F. Funke, and M.E. Rognes. Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4):369–393, 2013.

[86] J.H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 1997.

[87] O. Friedrich. Weighted Essentially Non-Oscillatory schemes for the interpolation of mean values on unstructured grids. *Journal of Computational Physics*, 144(1):194–212, 1998.

[88] T.P. Fries and H.G. Matthies. A review of Petrov-Galerkin stabilization approaches and an extension to meshfree methods. *Technical University Braunschweig*, 2004.

[89] J. Fürst. A Finite Volume scheme with Weighted Least Square reconstruction. *ICCFD: 4th International Conference on Computational Fluid Dynamics, Ghent, Belgium*, 2006.

[90] J. Fürst. The Weighted Least Square scheme for multidimensional flows. *ECCOMAS CFD: 4th European Conference on Computational Fluid Dynamics, Delft, Netherlands*, 2006.

[91] N.R. Gauger, V. Schulz, S. Schmidt, and C. Ilic. Shape gradients and their smoothness for practical aerodynamic design optimization. *RWTH Aachen University, Report no. SPP-1253-10-03*, 2008.

[92] A.H. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.

[93] A.H. Gebremedhin, A. Pothen, and A. Walther. Exploiting sparsity in Jacobian computation via coloring and automatic differentiation: A case study in a simulated moving bed process. *Advances in Automatic Differentiation; Lecture Notes in Computational Science and Engineering*, 64:327–338, 2008.

[94] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982.

[95] K.C. Giannakoglou. Continuous adjoint methods in shape, topology, flow-control and robust optimization. *ICON-CFD: Open Source CFD International Conference, London, UK*, 2012.

[96] J.C. Gilbert. Automatic Differentiation and iterative processes. *Optimization Methods and Software*, 1:13–21, 1992.

[97] M.B. Giles. Collected matrix derivative results for forward and reverse mode Algorithmic Differentiation. *Advances in Automatic Differentiation; Lecture Notes in Computational Science and Engineering*, 64:35–44, 2008.

[98] M.B. Giles, M.C. Duta, J-D. Müller, and N.A. Pierce. Algorithm developments for discrete adjoint methods. *AIAA Journal*, 41(2):198–205, 2003.

[99] M.B. Giles, D. Ghate, and M.C. Duta. Using Automatic Differentiation for adjoint CFD code development. *Oxford University Computing Laboratory, Oxford, UK*, 2005.

[100] M.B. Giles and N.A. Pierce. Adjoint equations in CFD: duality, boundary conditions and solution behaviour. *AIAA Paper 97-1850*, 1997.

[101] M.B. Giles and N.A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65:393–415, 2000.

[102] V. Girault and P.A. Raviart. *Finite element methods for the Navier-Stokes equations: theory and algorithms.* Springer, 1986.

[103] V. Girault, B. Rivière, and M.F. Wheeler. A discontinuous Galerkin method with non-overlapping domain decomposition for the Stokes and Navier-Stokes problems. *Mathematics of Computation*, 74:53–84, 2005.

[104] R. Glowinski. Finite element methods for Navier-Stokes equations. *Annual Review of Fluid Mechanics*, 24:167–204, 1992.

[105] P.M. Gresho. Some current CFD issues relevant to the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 87(3):201–252, 1991.

[106] A. Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse Automatic Differentiation. *Optilization Methods and Software*, 1(1):35–54, 1992.

[107] A. Griewank and A. Walther. Algorithm 799: Revolve: an implementation of checkpointing for the reverse or adjoint mode of Computational Differentiation. *ACM Transactions on Mathematical Software*, 26(1):19–45, 2000.

[108] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM, second edition, 2008.

[109] M. Gugala, O. Mykhaskiv, and J-D. Müller. A comparison of node-based and cad-based parametrisations in shape optimisation. *NOED International Conference on Numerical Optimisation Methods for Engineering Design, Munich, Germany*, 2016.

[110] M. Gugala, S. Xu, and J-D. Müller. Node-based vs. CAD-based approach in CFD adjoint-based shape optimisation. *ECCM V: 5th European Conference on Computational Mechanics, Barcelona, Spain*, 2014.

[111] M.D. Gunzburger. *Finite element methods for viscous incompressible flows – a guide to theory, practice, and algorithms.* Computer Science and Scientific Computing (Academic Press), 1989.

[112] M.D. Gunzburger. *Perspectives in Flow Control and Optimization.* SIAM, 2003.

[113] P. Hansbo. Generalized Laplacian smoothing of unstructured grids. *International Journal for Numerical Methods in Biomedical Engineering*, 11(5):455–464, 1995.

[114] H. Harborth and M. Möller. Minimum integral drawings of the platonic graphs. *Mathematics Magazine*, 67(5):355–358, 1994.

[115] E. Hardee, K.H. Chang, J. Tu, K.K. Choi, I. Grindeanu, and X. Yu. A CAD-based design parametrization for shape optimization of elastic solids. *Advances in Engineering Software*, 30:185–199, 1999.

[116] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy. Uniformly high order accurate Essentially Non-Oscillatory schemes III. *Journal of Computational Physics*, 71(2):231–303, 1987.

[117] L. Hascoët. Automatic Differentiation by program transformation. *INRIA Sophia Antipolis, France, TROPICS team*, 2007.

[118] L. Hascoët, U. Naumann, and V. Pascual. TBR analysis in reverse-mode Automatic Differentiation. *INRIA Sophia Antipolis, France*, 2003.

[119] T. Heister and G. Rapin. Efficient Augmented Lagrangian-type preconditioning for the Oseen problem using Grad-Div stabilization. *International Journal for Numerical Methods in Fluids*, 71(1):118–134, 2013.

[120] E. Helgason and S. Krajnović. Aerodynamic shape optimization of a pipe using the adjoint method. *International Mechanical Engineering Congress & Exposition, Houston, USA*, 2012.

[121] J.G. Heywood, R. Rannacher, and S.Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 22(5):325–352, 1996.

[122] C. Hinterberger and M. Olesen. Industrial application of continuous adjoint flow solvers for the optimization of automotive exhaust systems. *CFD & OPTIMIZATION, ECCOMAS Thematic Conference, Antalya, Turkey*, 2011.

[123] M. Hinze and J. Sternberg. A-Revolve: an adaptive memory and run-time reduced procedure for calculating adjoints; with an application to the instationary Navier-Stokes system. *Optimization Methods and Software*, 20:645–663, 2005.

[124] M. Hojjat, E. Stavropoulou, and K.U. Bletzinger. The Vertex Morphing method for node-based shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268(1):494–513, 2014.

[125] C. Hu and C-W. Shu. Weighted Essentially Non-Oscillatory schemes on triangular meshes. *Journal of Computational Physics*, 150(1):97–127, 1999.

[126] T.J.R. Hughes and A. Brooks. A multidimensional upwind scheme with no crosswind diffusion. *Finite Element Methods for Convection Dominated Flows, ASME*, pages 19–35, 1979.

[127] T.J.R. Hughes, L.P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics. V. Circumventing the Brezzi-Babuška condition: a stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59:85–99, 1986.

[128] T.J.R. Hughes, L.P. Franca, and G.M. Hulbert. A new finite element formulation for computational fluid dynamics. VIII. The Galerkin/Least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73:163–189, 1989.

[129] D. Irsarri. Virtual element method stabilization for convection-diffusion-reaction problems using the link-cutting condition. *Calcolo*, 54(1):141–154, 2017.

[130] S. Jakobsson and O. Amoignon. Mesh deformation using basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, 36(6):1119–1136, 2007.

[131] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988.

[132] A. Jameson. Optimum aerodynamic design using CFD and control theory. *AIAA Paper 95-1729*, 1995.

[133] A. Jameson and S. Kim. Reduction of the adjoint gradient formula in the continuous limit. *AIAA Paper 2003-0040*, 2003.

[134] A. Jameson, N.A. Pierce, and L. Martinelli. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and Computational Fluid Dynamics*, 10:213–237, 1998.

[135] H. Jasak. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, Imperial College, 1996.

[136] A. Jemcov and J.P. Maruszewski. Algorithm stabilization and acceleration in computational fluid dynamics: exploiting recursive properties of fixed point algorithms. *Computational Fluid Dynamics and Heat Transfer*, 23:459–486, 2010.

[137] D. Jones, J-D. Müller, and S. Bayyuk. CFD development with Automatic Differentiation. *AIAA Paper 2012-0573*, 2012.

[138] G.K. Karpouzas, E.M. Papoutis-Kiachagias, T. Schumacher, E. De Villiers, K.C. Giannakoglou, and C. Othmer. Adjoint optimization for vehicle external aerodynamics. *International Journal of Automotive Engineering*, 7:1–7, 2016.

[139] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier-Stokes equations. *SIAM Journal of Scientific Computing*, 24(1):237–256, 2002.

[140] R.A. Klausen and A.F. Stephansen. Convergence of Multi-Point Flux Approximations on general grids and media. *International Journal of Numerical Analysis & Modeling*, 9(3):584–606, 2012.

[141] J.A. Krakos and D.L. Darmofal. Effect of small-scale output unsteadiness on adjoint-based sensitivity. *AIAA Journal*, 48(11):2611–2623, 2010.

[142] D. Krentel, R. Muminovic, A. Brunn, W. Nitshe, and R. King. Application of active flow control on generic 3D car models. *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, 108:223–239, 2010.

[143] Y. Kuznetsov, K. Lipnikov, and M. Shashkov. The Mimetic Finite Difference method on polygonal meshes for diffusion-type problems. *Computational Geosciences*, 8:301–324, 2004.

[144] S. Langer, A. Schwöppe, and N. Kroll. The DLR flow solver TAU – status and recent algorithmic developments. *AIAA Paper 14-0080*, 2014.

[145] B.P. Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer Methods in Applied Mechanics and Engineering*, 19(1):59–98, 1979.

[146] R. Lewis. *A Guide to Graph Colouring: Algorithms and Applications.* Springer International Publishers, 2015.

[147] R.W. Lewis, P. Nithiarasu, and K.N. Seetharamu. *Fundamentals of the Finite Element Method for Heat and Fluid Flow.* John Wiley & Sons, 2004.

[148] A. Liatsikouras, G. Eleftheriou, G. Pierrot, and M. Megahed. Soft handle CAD-free parametrization tool for adjoint-based optimization methods. *NOED International Conference on Numerical Optimisation Methods for Engineering Design, Munich, Germany*, 2016.

[149] A. Linke. *Divergence-Free Mixed Finite Elements for the Incompressible Navier-Stokes Equation.* PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2007.

[150] K. Lipnikov and G. Manzini. A high-order mimetic method on unstructured polyhedral meshes for the diffusion equation. *Journal of Computational Physics*, 272(1):360–385, 2014.

[151] K. Lipnikov, G. Manzini, F. Brezzi, and A. Buffa. The Mimetic Finite Difference method for the 3D magnetostatic field problems on polyhedral meshes. *Journal of Computational Physics*, 230(2):305–328, 2011.

[152] K. Lipnikov, G. Manzini, and M. Shashkov. Mimetic Finite Difference method. *Journal of Computational Physics*, 257:1163–1227, 2014.

[153] K. Lipnikov, M. Shashkov, and D. Svyatskiy. The Mimetic Finite Difference discretization of diffusion problem on unstructured polyhedral meshes. *Journal of Computational Physics*, 211:473–491, 2006.

[154] X. Liu, S. Osher, and T. Chan. Weighted Essentially Non-Oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.

[155] D. Loghin and A.J. Wathen. Schur complement preconditioners for the Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 40(4):403–412, 2002.

[156] D. Lynch. Unified approach to simulation on deforming elements with applications to phase change problems. *Journal of Computational Physics*, 47(3):387–411, 1982.

[157] S. Majumdar. Role of underrelaxation in momentum interpolation for calculation of flow with nonstaggered grids. *Numerical Heat Transfer*, 13(1):125–132, 1988.

[158] L. Mangani, M. Buchmayr, and M. Darwish. Development of a novel fully coupled solver in OpenFOAM: Steady-state incompressible turbulent flows. *Numerical Heat Transfer, Part B: Fundamentals*, 66(1):1–20, 2014.

[159] G. Manzini, A. Cangiani, and O. Sutton. The conforming Virtual Element Method for the convection-diffusion-reaction equation with variable coeffcients. *Los Alamos National Laboratory, Report no. LA-UR-14-27710*, 2014.

[160] D. Marx. Graph colouring problems and their applications in scheduling. *Periodica Polytechnica, Electrical Engineering*, 48(1):11–16, 2004.

[161] G. Matthies and L. Tobiska. Mass conservation of finite element methods for coupled flow-transport problems. *International Journal of Computing Science and Mathematics*, 1(2):293–307, 2007.

[162] D.W. Matula, Y. Shiloach, and R.E. Tarjan. Two linear-time algorithms for five-coloring a planar graph. *Stanford University, Report no. STAN-CS-80-830*, 1980.

[163] D. Mavriplis. Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. *AIAA: 16th Computational Fluid Dynamics Conference, Orlando, USA*, 2003.

[164] D. Mavriplis. Solution of the unsteady discrete adjoint for three-dimensional problems on dynamically deforming unstructured meshes. *AIAA Paper 2008-0727*, 2008.

[165] S. May and M. Berger. Two-dimensional slope limiters for Finite Volume schemes on non-coordinate-aligned meshes. *SIAM Journal on Scientific Computing*, 35(5):2163–2187, 2013.

[166] K. Michalak and C. Ollivier-Gooch. Limiters for unstructured higher-order accurate solutions of the Euler equations. *AIAA Paper 2008-0776*, 2008.

[167] J-D. Müller. Efficient sensitivity computation using Automatic Differentiation. *Introduction to Optimization and Multidisciplinary Design, Von Karman Institute, Sint-Genesius-Rode, Belgium*, 2016.

[168] J-D. Müller and P. Cusdin. On the performance of discrete adjoint CFD codes using automatic differentiation. *International Journal for Numerical Methods in Fluids*, 47(8-9):939–945, 2005.

[169] D.J. Monson and H.L. Seegmiller. An experimental investigation of subsonic flow in a two dimensional U-duct. *NASA report TM-103931*, 1992.

[170] A. Montlaur, S. Fernandez-Méndez, and A. Huerta. Discontinuous Galerkin methods for the Stokes equations using divergence-free approximations. *International Journal for Numerical Methods in Fluids*, 75(9):1071–1092, 2008.

[171] A. Moraes, P. Lage, G. Cunha, and L.F. Lopes Rodrigues da Silva. Analysis of the non-orthogonality correction of Finite Volume discretization on unstructured meshes. *22nd International Congress of Mechanical Engineering (COBEM), São Paulo, Brazil*, 2013.

[172] O. Mykhaskiv, J-D. Müller, S. Auriemma, H. Legrand, M. Banovic, and A. Walther. Shape optimisation with differentiated CAD-kernel for U-bend testcase. *NOED International Conference on Numerical Optimisation Methods for Engineering Design, Munich, Germany*, 2016.

[173] S.K. Nadarajah. *The Discrete Adjoint Approach to Aerodynamic Shape Optimization*. PhD thesis, Stanford University, 2003.

[174] S.K. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA Paper 2000-0667*, 2000.

[175] A. Nemili, E. Ozkaya, N. Gauger, A. Carnarius, and F. Thiele. Automatic generation of discrete adjoints for unsteady optimal flow control. *CFD & OPTIMIZATION, ECCOMAS Thematic Conference, Antalya, Turkey*, 2011.

[176] C.F. Ollivier-Gooch. Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction. *Journal of Computational Physics*, 133(1):6–17, 1997.

[177] M.A. Olshanskii and M. Benzi. An augmented Lagrangian approach to linearized problems in hydrodynamic stability. *SIAM Journal on Scientific Computing*, 30(3):1459–1473, 2008.

[178] M.A. Olshanskii and A. Reusken. Grad-div stablilization for Stokes equations. *Mathematics of Computation*, 73:1699–1718, 2004.

[179] M.A. Olshanskii and Y.V. Vassilevski. Pressure Schur complement preconditioners for the discrete Oseen problem. *SIAM Journal on Scientific Computing*, 29(6):2686–2704, 2007.

[180] L. Osusky, H. Buckley, T. Reist, and D. Zingg. Drag minimization based on the Navier–Stokes equations using a Newton–Krylov approach. *SIAM Journal on Scientific Computing*, 53(6):1555–1577, 2015.

[181] C. Othmer. Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, 4(6), 2014.

[182] C. Othmer and T. Grahs. Approaches to fluid dynamic optimization in the car development process. *ECCOMAS conference, Munich, Germany*, 2005.

[183] C. Paniconi and M. Putti. A comparison of Picard and Newton iteration in the numerical solution of multidimensional variably saturated flow problems. *Water Resources Journal*, 30(12):3357–3374, 1994.

[184] E.M. Papoutsis-Kiachagias, K.C. Giannakoglou, and C. Othmer. Adjoint wall functions: validation and application to vehicle aerodynamics. *ECCM V: 5th European Conference on Computational Mechanics, Barcelona, Spain*, 2014.

[185] E.M. Papoutsis-Kiachagias, N. Magoulas, J-D. Müller, C. Othmer, and K.C. Giannakoglou. Noise reduction in car aerodynamics using a surrogate objective function and the continuous adjoint method with wall functions. *Computers & Fluids*, 122:223–232, 2015.

[186] E.M. Papoutsis-Kiachagias, A.S. Zymaris, I.S. Kavvadias, D.I. Papadimitriou, and K.C. Giannakoglou. The continuous adjoint approach to the $k\breve{}\varepsilon$ turbulence model for shape optimization and optimal active control of turbulent flows. *Engineering Optimization*, 47(3):370–389, 2015.

[187] J.S. Park, S.H. Yoon, and C. Kim. Multi-dimensional limiting process for hyperbolic conservation laws on unstructured grids. *Journal of Computational Physics*, 229(3):788–812, 2010.

[188] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow.* Minkowycz and Sparrow Eds. (Mc Graw Hill), 1980.

[189] J. Peter and R.P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391, 2010.

[190] S. Petropoulou. Industrial optimisation solutions based on OpenFOAM technology. *ECCOMAS CFD: 5th European Conference on Computational Fluid Dynamics, Lisbon, Portugal*, 2010.

[191] L. Piar, F. Babik, R. Herbin, and J.C. Latché. A formally second order cell centered scheme for convection-diffusion equations on unstructured non-conforming grids. *International Journal for Numerical Methods in Fluids*, 71(7):873–890, 2013.

[192] L. Piegl and W. Tiller. *The NURBS Book.* Springer, 1997.

[193] G. Pierrot. A gentle introduction to ESI i-Adjoint library. *ESI Group, Rungis, France*, 2012.

[194] G. Pierrot. Bridging the gap between continuous and discrete adjoint solvers for incompressible Navier-Stokes equations. *OPT-I: International Conference on Engineering and Applied Sciences Optimization, Kos, Greece*, 2014.

[195] G. Pierrot. Equational Differentiation of incompressible flow solvers as a middle ground between continuous and discrete adjoint methodologies. *ECCM V: 5th European Conference on Computational Mechanics, Barcelona, Spain*, 2014.

[196] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64:97–110, 1974.

[197] E. Potočar, B. Širok, M. Hočevar, and M. Eberlinc. Control of separation flow over a wind turbine blade with plasma actuators. *Journal of Mechanical Engineering*, 58(1):37–45, 2012.

[198] M. Rehman, C. Vuik, and G. Segal. A comparison of preconditioners for incompressible Navier-Stokes solvers. *International Journal for Numerical Methods in Fluids*, 57(12):1731–1751, 2008.

[199] M. Rehman, C. Vuik, and G. Segal. Preconditioners for the steady incompressible Navier-Stokes problem. *IAENG International Journal of Applied Mathematics*, 38(4), 2008.

[200] D. Rempfer. On boundary conditions for incompressible Navier-Stokes problems. *Applied Mechanics Reviews*, 59(3):107–125, 2006.

[201] J. Reuther and A. Jameson. Control based airfoil design using the Euler equations. *AIAA Paper 94-4272*, 1994.

[202] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. *AIAA Paper 99-0094*, 1996.

[203] C.M. Rhie and W.L. Chow. Numerical study of the turbulent fow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532, 1983.

[204] T.T. Robinson, C.G. Armstrong, H.S. Chua, C. Othmer, and T. Grahs. Optimizing parametrized CAD geometries using sensitivities based on adjoint functions. *Computer Aided Design & Applications*, 9(3):253–268, 2012.

[205] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[206] F. Cayré S. Boivin and J.M. Hérard. A finite volume method to solve the Navier-Stokes equations for incompressible flows on unstructured meshes. *International Journal of Thermal Sciences*, 39(8):806–825, 2000.

[207] J.A. Samareh. Survey of shape parametrization techniques for high-fidelity multi-disciplinary shape optimization. *AIAA Journal*, 39(5):877–884, 2001.

[208] D. L. Scharfetter and H. K. Gummel. Large signal analysis of a silicon read diode. *IEEE Transactions on Electron Devices*, 16(1):64–77, 1969.

[209] O. Schmitt and P. Steinmann. On curvature control in node–based shape optimization. *Proceedings in Applied Mathematics and Mechanics*, 15(1):579–580, 2015.

[210] J.J. Schneider and S. Kirkpatrick. *Stochastic Optimization.* Springer, 2006.

[211] A. Segal, M. Rehman, and C. Vuik. Preconditioners for incompressible Navier-Stokes solvers. *Numerical Mathematics: Theory, Methods and Applications*, 3(3):245–275, 2010.

[212] G.M. Shroff and H.B. Keller. Stabilization of unstable procedures: The Recursive Projection Method. *SIAM Journal on Numerical Analysis*, 30(4):1099–1120, 1993.

[213] C-W. Shu. Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory schemes for hyperbolic conservation laws. *NASA/CR-97-206253, ICASE Report No. 97-65*, 1997.

[214] D. Silvester, H. Elman, D. Kay, and A. Wathen. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics*, 128(2):261–279, 2001.

[215] R. Smith and A. Hutton. The numerical treatment of advection: a performance comparison of current methods. *Numerical Heat Transfer*, 5:439–461, 1982.

[216] P.R. Spalart and S.R. Allmaras. A one-equation turbulence model for aerodynamic flows. *Recherche Aerospatiale*, 1:5–21, 1994.

[217] A. Stück. *Adjoint Navier–Stokes Methods for Hydrodynamic Shape Optimisation.* PhD thesis, Technischen Universität Hamburg-Harburg, 2011.

[218] A. Stück and T. Rung. Adjoint complement to viscous finite-volume pressure-correction methods. *Journal of Computational Physics*, 248(1):402–419, 2013.

[219] K. Stein, T. Tezduyar, and R. Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *Journal of Applied Mechanics*, 70(1):58–63, 2003.

[220] Q.A. Ta. A short description of the i-Adjoint benchmark. *ESI Group, Rungis, France*, 2013.

[221] T. Tezduyar and S. Sathe. Stabilization parameters in SUPG and PSPG formulations. *Journal of Computational and Applied Mathematics*, 4(1):71–88, 2003.

[222] M. Towara, A.Sen, and U. Naumann. An effective discrete adjoint model for OpenFOAM. *OPT-I: International Conference on Engineering and Applied Sciences Optimization, Kos, Greece*, 2014.

[223] M. Towara and U. Naumann. A discrete adjoint model for OpenFOAM. *Procedia Computer Science*, 18:429–438, 2013.

[224] M. Towara, M. Schanen, and U. Naumann. MPI-parallel discrete adjoint OpenFOAM. *Procedia Computer Science*, 51:19–28, 2015.

[225] P.G. Tucker. Differential equation-based wall distance computation for DES and RANS. *Journal of Computational Physics*, 190(1):229–248, 2003.

[226] J.P. Van Doormal and G.D. Raithby. Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 7(2):147–163, 1984.

[227] B. van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101–136, 1979.

[228] I. Vasilopoulos, P. Flassig, and M. Meyer. CAD-based aerodynamic optimization of a compressor stator using conventional and adjoint-driven approaches. *ASME Turbo Expo 2017: Turbomachinery Technical Conference and Exposition, Charlotte, USA*, 2017.

[229] H. Veersteg and W. Malalasekera. *An introduction to Computational Fluid Dynamics: the Finite Volume method.* Pearson Education, 2007.

[230] V. Venkatakrishnan. On the accuracy of limiters and convergence to steady-state solution. *AIAA Paper 93-0880*, 1993.

[231] V. Venkatakrishnan. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, 118:120–130, 1995.

[232] T. Verstraete, F. Coletti, J. Bulle, T.V. der Wielen, and T. Arts. Optimization of a U-bend for minimal pressure loss in internal cooling channels - part I: Numerical method. *ASME Journal of engineering for Gas Turbines and Power*, 135(5), 2013.

[233] C. Vezyris, E. Papoutsis-Kiachagias, I. Kavvadias, and K. Giannakoglou. Steady & unsteady continuous adjoint method using a pseudo-compressibility block coupled solver in OpenFoam. *ECCOMAS Congress: 7th European Congress on Computational Methods in Applied Sciences and Engineering, Crete, Greece*, 2016.

[234] C. Vuik and A. Segal. Solution of the coupled Navier-Stokes equations. *Notes on Numerical Fluid Mechanics*, 51:186–197, 1995.

[235] Q. Wang and J-H. Gao. The drag-adjoint field of a circular cylinder wake at Reynolds numbers 20, 100 and 500. *Journal of Fluid Mechanics*, 730:145–161, 2013.

[236] Q. Wang and P. Moin. Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *Center for Turbulence Research - Annual Research Briefs*, pages 55–73, 2008.

[237] D.J.A. Welsh and M.B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[238] D.C. Wilcox. *Turbulence Modeling for CFD.* DCW industries, Inc., second edition, 2002.

[239] S.Ø. Wille and A.F.D. Loula. A priori pivoting in solving the Navier-Stokes equations. *Communications in Numerical Methods in Engineering*, 18(10):691–698, 2002.

[240] S.Ø. Wille, O. Staff, and A.F.D. Loula. Efficient a priori pivoting schemes for a sparse direct Gaussian equation solver for the mixed finite element formulation of the Navier-Stokes equations. *Applied Mathematical Modelling*, 28(7):607–616, 2004.

[241] S. Xu, W. Jahn, and J-D. Müller. CAD-based shape optimisation with CFD using a discrete adjoint. *International Journal for Numerical Methods in Fluids*, 74(3):153–168, 2014.

[242] S. Xu, D. Radford, M. Meyer, and J-D. Müller. CAD-based adjoint shape optimisation of a one-stage turbine with geometric constraints. *ASME Turbo Expo 2015: Turbine Technical Conference and Exposition, Montreal, Canada*, 2015.

[243] S. Xu, D. Radford, M. Meyer, and J-D. Müller. Stabilisation of discrete steady adjoint solvers. *Journal of Computational Physics*, 299:175–195, 2015.

[244] S. Xu and S. Timme. Robust and efficient adjoint solver for complex flow conditions. *Computers and Fluids*, 148:26–38, 2017.

[245] G. Yu, J-D. Müller, D. Jones, and F. Christakopoulos. CAD-based shape optimisation using adjoint sensitivities. *Computers and Fluids*, 46(1):512–516, 2011.

[246] X. Zeng. A general approach to enhance slope limiters in MUSCL schemes on nonuniform rectilinear grids. *SIAM Journal on Scientific Computing*, 38(2):789–813, 2016.

[247] S. Zhang. On the P1 Powell-Sabin divergence-free finite element for the Stokes equationss. *Journal of Computational Mathematics*, 26(3):456–470, 2008.