Shape and Topology Optimization using the Cut-Cell Method and its Continuous Adjoint for Single– and Two–phase Turbulent flows, in a Multiprocessor Environment

Vrionis Panayiotis–Yiannis

Parrallel CFD & Optimization Unit (PCOpt)

Laboratory of Thermal Turbomachines

School of Mechanical Engineering

National Technical University of Athens

A thesis submitted for the degree of

Doctor of Philosophy

Athens, 2022

Ευρωπαϊκή Ένωση
European Social Fund

Operational Programme
Human Resources Development, Education and Lifelong Learning
Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Laboratory of Thermal Turbomachines
Parallel CFD & Optimization Unit

Shape and Topology Optimization using the Cut-Cell Method and its Continuous Adjoint for Single– and Two–phase Turbulent flows, in a Multiprocessor Environment

Vrionis Panayiotis–Yiannis

Examination Committee:

1. Kyriakos Giannakoglou*(Supervisor),
   Professor NTUA, School of Mechanical Engineering
2. Konstantinos Mathioudakis*,
   Professor NTUA, School of Mechanical Engineering
3. Demetri Bouris*,
   Associate Professor NTUA, School of Mechanical Engineering
4. Spyridon Voutsinas,
   Professor NTUA, School of Mechanical Engineering
5. Ioannis Anagnostopoulos,
   Professor NTUA, School of Mechanical Engineering
6. Kostas Belibassakis,
   Professor NTUA, School of Naval Architecture and Marine Engineering
7. Georgios Papadakis,
   Assistant Professor NTUA, School of Naval Architecture and Marine Engineering

**Member of the Advisory Committee*

Abstract

This PhD thesis focuses on developing a highly automated, general–purpose software for the analysis and design of practical applications in the field of fluid mechanics by exploiting the merits of the Cut–Cell method. The principal idea of the Cut–Cell method lies in the generation of simple Cartesian meshes, subsequently modified by discarding parts that reside in the solid region. This process creates cut cells that conform to the body surface. The use of Cartesian meshes eases the mesh generation process of the entire workflow, regardless of the body surface complexity. Emphasis is laid on the extension of the existing Cut–Cell software, capable of numerically solving laminar, incompressible single–phase flows, via the artificial compressibility method, to turbulent, single– and two–phase flows exhibiting cavitation. In addition, the developed Cut–Cell method is applied in gradient–based optimization using the continuous adjoint method.

The Cut–Cell software is extended to the RANS equations using the standard $k-\varepsilon$ turbulence model, solved coupled with the mean flow equations. The choice of this turbulence model is its broad usage in two–phase flows. In addition, the Cut–Cell method necessitates the use of a high Reynolds turbulence model, due to difficulties in refining along the normal–to–the–wall direction to accurately resolve the turbulent boundary layers, which results in impractical meshing requirements. Thus, closure is realized via a modified wall functions technique, while also accounting for varying first cell distances, normal to the wall.

The analysis of two–phase flows that include cavitation effects follows the homogeneous mixture assumption. Thus, a mixture pseudo–fluid is introduced, the properties of which arise based on the local composition of each constituent. These are assumed to have a constant density. The RANS equations are, thus, expressed for the mixture fluid and an additional species transport equation is included. Cavitation effects are introduced using the Kunz cavitation model, which quantifies the appropriate mass transfer between constituents, to account for evaporation and

condensation. Numerically, a shock–capturing scheme is employed to account for the large flow variable gradients present at the two–phase interfaces, opting for their flexibility and the straightforward extension from the existing gas–dynamics solver. In detail, the numerical solution process uses a cell–centered finite volume method that computes inviscid numerical fluxes via the MUSCL scheme and Roe's approximate Riemann solver. Due to the homogeneous mixture assumption and the involvement of additional transport equations for the species, the resulting governing equation need be appropriately preconditioned (Kunz preconditioner). The resulting numerical scheme allows for the simulation of two–phase flows with large density gradients and benefits from the localized mesh refinement from the Cut–Cell method.

In the field of gradient–based optimization, the development of both shape and topology optimization tools based on the Cut–Cell method is pursued. The continuous adjoint equivalent to the described two–phase RANS equations and cavitation model is mathematically formulated to compute objective function gradients at a cost that is independent to the number of design variables. The method is integrated into a shape optimization framework and applied to isolated air/hydrofoils and ducts, parameterized using Bézier–Bernstein polynomials. The inclusion of the Cut–Cell method in a shape optimization framework confines mesh changes at the immediate vicinity of the body surface, circumventing the need for mesh displacement techniques, and allows large displacements without aggravating the Cut–Cell mesh quality.

The ability of the Cut–Cell method to automatically generate valid meshes around rapidly changing solid boundaries inspired the development of a new topology optimization workflow. The new method overcomes the low near–wall accuracy of porosity–based optimization methods by reconstructing the solid walls in each optimization cycle. This is achieved by modifying the process that generates the Cut–Cells. Furthermore, the GCMMA algorithm is implemented to facilitate the large number of bounded design variables and introduced constraint function, typically used in topology optimization. The developed method is assessed through comparisons with a porosity–based topology optimization solver. For the sake of fairness, porosity–based optimized solutions are re–evaluated on body–fitted Cut–Cell meshes, using a utility tool developed herein.

The extensions are integrated into the existing Cut–Cell software and allow parallel computations on many processors using the MPI protocol. Additionally, selected computationally intensive tasks are locally parallelized using threads via OpenMP directives.

Keywords: Computational Fluids Dynamics, Two–phase flow, Shape and Topology Optimization, Continuous Adjoint Method, Cut–Cell Method.

Περίληψη

Η διδακτορική αυτή διατριβή ασχολείται με την ανάπτυξη ενός ευρείας χρήσης λογισμικού ανάλυσης και σχεδιασμού πρακτικών εφαρμογών της μηχανικής των ρευστών με βάση τη Μέθοδο Τεμνόμενων Κυψελών. Η τελευταία βασίζεται στην τροποποίηση Καρτεσιανών πλεγμάτων, αποκόπτοντας τμήματα που καταλαμβάνονται από τα στερεά σώματα, ούτως ώστε να κατασκευάσει οριόδετα πλέγματα. Συνεπώς, το τελικό υπολογιστικό πλέγμα αποτελείται κυρίως από Καρτεσιανές κυψέλες και εκείνες που τέμνονται με το στερεό σώμα και τους έχει αποκοπεί ένα τμήμα. Η χρήση Καρτεσιανών πλεγμάτων εισάγει αυτοματοποίηση στη διαδικασία γένεσης πλέγματος και, συνεπώς, η ανάπτυξη εξειδικευμένων μεθόδων που εκμεταλλεύονται αυτήν την ιδιότητα είναι επίσης κύριος στόχος της διατριβής. Σημείο αναφοράς αποτελεί ένα υπάρχον λογισμικό που υλοποιεί τη Μέθοδο Τεμνόμενων Κυψελών για την επίλυση ασυμπίεστων, στρωτών, μονοφασικών ροών, με τη χρήση της μεθόδου της ψευδοσυμπιεστότητας, το οποίο επεκτείνεται τυρβώδεις διφασικές ροές που παρουσιάζουν σπηλαίωση και σε εφαρμογές σχεδιασμού με τη χρήση της συνεχούς συζυγούς μεθόδου για τον υπολογισμό παραγώγων ευαισθησίας.

Αρχικά, η επέκταση σε τυρβώδεις ροές γίνεται με την υλοποίηση του μοντέλου τύρβης $k - \varepsilon$ των Launder και Spalding. Το σύστημα εξισώσεων (μέσης ροής και τυρβωδών μεταβλητών) επιλύεται πεπλέγμενα. Η επιλογή του συγκεκριμένου μοντέλου τύρβης βασίστηκε στην ευρεία χρήση του σε διφασικές ροές, καθώς επίσης και στη δυσκολία της Μεθόδου Τεμνόμενων Κυψελών να δημιουργεί, τοπικά, κυψέλες μεγάλου λόγου διαστάσεων. Αυτό έχει ως αποτέλεσμα την αδυναμία χρήσης μοντέλων τύρβης χαμηλών αριθμών Reynolds λόγω των τεράστιων απαιτήσεων πύκνωσης. Έτσι, η χρήση συναρτήσεων τοίχου αποτελεί, ουσιαστικά, μονόδρομο. Παρόλα αυτά, η χρήση συναρτήσεων τοίχου παρουσιάζει επίσης κάποιες ιδιαιτερότητες λόγω της αρκετά μεταβαλλόμενης απόστασης των βαρύκεντρων των Τεμνόμενων Κυψελών από το στερεό όριο και τροποποιείται καταλλήλως.

Όσον αφορά την ανάπτυξη αριθμητικών μεθόδων ανάλυση διφασικών ροών που παρουσιάζουν σπηλαίωση, η αριθμητική μέθοδος βασίζεται στη θεώρηση ενός ομογενούς μείγματος. Πρακτικά, γίνεται η εισαγωγή ενός ψευδο–ρευστού, του οποίου η πυκνότητα και μοριακή συνεκτικότητα προκύπτει βάσει της τοπικής συγκέντρωσης των ουσιών που το αποτελούν. Έτσι, οι εξισώσεις ροής για τη διατήρηση όγκου–ορμής και τη μεταφορά των τυρβωδών μεταβλητών μπορούν να εκφραστούν ως προς το μείγμα και τη μεταφορά της δευτερεύουσας φάσης, μειώνοντας το συνολικό υπολογιστικό κόστος. Το φαινόμενο της σπηλαίωσης εισάγεται στην αριθμητική μέθοδο μέσω του μοντέλου

του Kunz, που μοντελοποιεί τις διαδικασίες εξάτμισης και συμπύκνωσης. Η αριθμητική μέθοδος που αναπτύχθηκε για την επίλυση των εξισώσεων, βασίζεται στην ήδη υπάρχουσα υποδομή της κεντροκυψελικής διατύπωσης πεπερασμένων όγκων. Ο υπολογισμός των ατριβών όρων γίνεται μέσω του σχήματος MUSCL και του κατά Roe επιλύτη. Στο τμήμα αυτό απαιτείται η τροποποίηση του σχήματος το οποίο λαμβάνει υπόψη τις επιπλέον ιδιοτιμές και προσταθεροποίηση κατά Kunz. Συνολικά, το αριθμητικό σχήμα που υλοποιήθηκε επιτρέπει την ανάλυση διφασικών ροών μεγάλου λόγου πυκνοτήτων και επωφελείται τοπικών τεχνικών πύκνωσης του Καρτεσιανού πλέγματος.

Στο τμήμα της διατριβής που αφορά την ανάπτυξη μεθόδων σχεδιασμού καθοδηγούμενων από την κλίση της συνάρτησης–στόχου (παραγώγων ευαισθησίας), έγινε ανάπτυξη εργαλείων βελτιστοποίησης μορφής. Για τον υπολογισμό των παραγώγων ευαισθησίας έγινε η μαθηματική διατύπωση και ανάπτυξη του συζυγούς προβλήματος, με βάση τη συνεχή συζυγή μέθοδο, που διέπεται από τις διφασικές, τυρβώδεις εξισώσεις ροής που μοντελοποιούν και φαινόμενα σπηλαίωσης. Η συζυγής μέθοδος είναι η κυρίαρχη μέθοδος υπολογισμού παραγώγων ευαισθησίας αφού μπορεί να τις υπολογίζει με κόστος ανεξάρτητο του πλήθους των μεταβλητών σχεδιασμού, επιτρέποντας την εφαρμογή του σε προβλήματα μεγάλης κλίμακας. Η αναπτυχθείσα μέθοδος εντάσσεται σε ένα πλαίσιο βελτιστοποίησης μορφής, και εφαρμόζεται σε μεμονωμένες αερο/υδροτομές και αγωγούς, παραμετροποιημένες με τη χρήση πολυωνύμων Bézier–Bernstein. Πλεονεκτήματα, που σχετίζονται με τη χρήση της Μεθόδου των Τεμνόμενων Κυψελών, είναι το ότι επιτρέπει μεγάλες μετατοπίσεις του υπό μελέτη σώματος, περιορίζει τις αλλαγές του πλέγματος κοντά στο στερεό σώμα, και εγγυάται τη γένεση καλής ποιότητας πλεγμάτων κατά τη βελτιστοποίηση.

Η δυνατότητα της Μεθόδου των Τεμνόμενων Κυψελών να παράγει καλής ποιότητας πλέγματα γύρω από στερεά σώματα που επιδέχονται μεγάλες αλλαγές, ενέπνευσε την ανάπτυξη μιας διαδικασίας βελτιστοποίησης τοπολογίας. Η αναπτυχθείσα μέθοδος προσφέρει υψηλή ακρίβεια κατά τη διάρκεια επίλυσης του προβλήματος τοπολογίας, μέσω της κατασκευής των στερεών ορίων. Για την υλοποίηση αυτής της μεθόδου, ήταν, αρχικά, αναγκαία η τροποποίηση του τρόπου κατασκευής των Τεμνόμενων Κυψελών. Τα προβλήματα βελτιστοποίησης τοπολογίας χαρακτηρίζονται από μεγάλο αριθμό οριοθετημένων μεταβλητών σχεδιασμού και την εισαγωγή συναρτήσεων περιορισμού. Συνεπώς, ήταν απαραίτητη η υλοποίηση μιας μεθόδου ανανέωσης των μεταβλητών σχεδιασμού που να μπορεί να διαχειριστεί τέτοιου είδους προβλήματα βελτιστοποίησης (GCMMA). Για την αξιολόγηση της νέας μεθόδου, έγιναν συγκρίσεις με κλασική μέθοδο βελτιστοποίησης τοπολογίας, που βασίζεται στην τεχνική του πορώδους, για να εξαχθούν αποτελέσματα. Στο τμήμα αυτό, αναπτύχθηκε εργαλείο επαναξιολόγησης λύσεων, που προκύπτουν από την τεχνική του πορώδους σε οριόδετα πλέγματα, ώστε να είναι πιο αντιπροσωπευτική η σύγκριση.

Όλα τα παραπάνω εντάχθηκαν στο οικείο λογισμικό Τεμνόμενων Κυψελών και επι-

τρέπουν την παράλληλη εκτέλεση σε πολλούς επεξεργαστές μέσω του προτύπου MPI. Επιπρόσθετα, έγινε χρήση εντολών OpenMP που αξιοποιούν νήματα εντός των επεξεργαστών για την παραλληλοποίηση υπολογιστικά ακριβών διεργασιών.

Λέξεις κλειδιά: Μέθοδοι Υπολογιστικής Ρευστοδυναμικής, Διφασικές ροές, Βελτιστοποίηση Μορφής και Τοπολογίας, Συνεχής Συζυγής Μέθοδος, Μέθοδος Τεμνόμενων Κυψελών.

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor, Kyriakos Giannakoglou, Professor NTUA, for the opportunity to work under his guidance, his patience, consistent support and involvement throughout this thesis that were instrumental for the completion of this work.

I also wish to thank the other two members of the advisory committee Konstantinos Mathioudakis, Professor NTUA, and Demetris Bouris, Associate Professor NTUA, for their constructive remarks at the progress reports and presentation of this thesis.

I am also grateful to the State Scholarships Foundation (IKY) for funding this research through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research – 2^{nd} Cycle" (MIS–5000432), co-financed by Greece and the European Union (European Social Fund–ESF).

I would like to extend my sincere thanks to Dr. Konstantinos Samouchos for his invaluable help during my PhD research, the many thought–provoking discussions, and, of course, his friendship that made this collaboration truly pleasant. I also sincerely thank Dr. Evangelos–Papoutsis Kiachagias for his priceless advises, both of technical and personal nature. Many thanks to Dr. Konstantinos Tsiakas, Dr. Konstantinos Gkaragkounis, Dr. Morteza Monfaredi, PhD student Skamagkis Themis, with whom I spent many hours together, for their friendly chats and company that lightened the atmosphere during my PhD research.

Furthermore, I owe special thanks to my brother, Dr. Constantinos Vrionis for always being there eager to listen and help, and my girlfriend, Ioanna, for her understanding and the strength she has given me. I should also thank both of them for enduring through my grumpiness.

Last, but not least, I am forever indebted to my parents, George and Beatrice, for their unconditional support and their profound belief in my abilities that have encouraged me through each step.

Contents

Acronyms

ABCs Adjoint Boundary Conditions

AD Algorithmic Differentiation

ALM Augmented Lagrangian Method

AoA angle of attack

CC Cut-Cell

CCTopO Cut–Cell Topology Optimization

CFD Computational Fluid Dynamics

CPs Control Points

EFS Equivalent Flow Solutions

FAEs Field Adjoint Equations

FDs Finite Differences

FEM Finite Element Method

FSI Fluid—Structure Interaction

FVM Finite Volume Method

GCMMA Globally Convergent Method of Moving Asymptotes

GSs Geometric Sensitivities

HJE Hamilton–Jacobi Equation

IBM Immersed Boundary Method

IDW Inverse Distance Weighting

k–NNS k–Nearest Neighbors Search

KKT Karush—Kuhn—Tucker

MMA Method of Moving Asymptotes

MUSCL Monotonic Upstream–centered Scheme for Conservation Laws

OOP Object–Oriented Programming

PCOpt Parallel CFD & Optimization unit

PDE Partial Differential Equation

r–SP re–evaluated Standard Porosity

RANS Reynolds–Averaged Navier–Stokes

SA Spalart–Allmaras

SDs Sensitivity Derivatives

ShpO Shape Optimization

SPTopO Standard Porosity (Brinkman) Topology Optimization

STL STereoLithography

TEM Transport Equation Model

TopO Topology Optimization

TVD Total Variation Diminishing

VoF Volume of Fluid

WLSQ Weighted Least Squares

XFEM eXtended Finite Element Method

Chapter 1

Introduction

The simultaneous increase in computational power and the development of numerical methods for the solution of complex phenomena, governed by a system of Partial Differential Equations (PDE), have made computer–aided numerical predictions commonplace. In the field of fluid dynamics, Computational Fluid Dynamics (CFD) is commonly applied to provide insight into real–life problems ranging from the analysis and design of aerospace, marine and automotive applications to weather forecasting and chemical processes due to their time– and cost–effectiveness, and ease of examining different flow conditions and body surfaces. As a result, various sophisticated CFD and numerical optimization tools have been, and are continuously being, developed to provide additional flexibility, accuracy, and robustness. The ability to obtain superior designs at reduced costs by lowering the number of necessary experimental validations during the design phase [236] are credited to these tools.

However, the advanced simulation, and optimization, capabilities are often hindered by the mesh generation process. It constitutes one of the main bottlenecks in the simulation workflow. Simulations of complex body surfaces require significant human intervention for the body surface preprocessing and the mesh generation process to obtain accurate results and is a tedious, costly task that also necessitates user experience [236]. A remedy can be found in the Immersed Boundary Methods (IBM) that can potentially automate the entire mesh generation process even in complex configurations. The Cut-Cell (CC) method, a subclass of IBMs, poses the main method this thesis extends.

Following the trend for more advanced simulation capabilities, the extension of an existing CC–based Navier–Stokes equations code, developed in a previous PhD thesis [214], is pursued to support fluid problems governed by the Reynolds–Averaged Navier–Stokes (RANS) equations

and two–phase fluid problems of liquid and vapour featuring cavitation. These extensions introduce novelties not yet tackled in the associated literature, especially when considering their incorporation into a numerical optimization tool.

The following sections document the theoretical background of the components addressed throughout this thesis. More specifically, Section 1.1 establishes the concept of IBMs and, then, focuses on the CC method. Section 1.2 presents the aspects of the numerical challenges related to the simulation of cavitating flows and describes the commonly adopted numerical methods. Section 1.3 is concerned with the methods used to perform CFD–based optimization. Then, in section 1.4 the adjoint method is introduced and recounts its variations. Finally, Section 1.5 outlines this thesis structure.

1.1 Immersed Boundary Methods

The notion of IBMs was pioneered by Peskin [196] in his seminal work studying the blood flow in the human heart. The unique attribute of IBMs lies in their usage of meshes, not aligned with the body surface boundaries, but have them *immersed* within it. Instead of wrapping a mesh around the body surfaces, a fixed Cartesian mesh is created that encompasses both the fluid and solid regions. The presence of body surfaces in the flow is accounted for via force functions, acting upon the governing equations in either the continuous or discrete sense.

The use of IBMs offers several advantages and disadvantages when compared with conventional body–conforming approaches. The key feature of IBMs is the simplicity of mesh generation. It offers increased automation and flexibility in the mesh generation process that is absent when generating body–conformal structured or unstructured meshes. On the contrary, in such cases, the process is cumbersome and follows an iterative human–in–the–loop workflow to obtain good–quality body–conforming meshes [24, 164, 236]. This advantage becomes evident in case simulations of moving bodies are considered. Typically, conventional approaches require the deformation of the mesh to follow the body surface boundary in each timestep [149]. However, when simulating large body motions, or multiple bodies in relative motion, these mesh deformation techniques fail to provide good–quality meshes, necessitating re–meshing and interpolating the flow solution to the new mesh. Overall, these requirements lead to reduced accuracy and robustness and increase the computation cost of the simulation. For this reason, overset (Chimera) meshes have been developed that can partly remedy these drawbacks but still require advanced interpolation techniques between the meshes to maintain conservation [50]. In the IBMs, the Cartesian mesh remains fixed, enabling large body motions

and, in principle, need not be deformed or regenerated.

The main disadvantage of IBMs lies in their difficulty to impose boundary conditions along the body surfaces. The exact approach followed to incorporate the effects of the solid boundaries is critical and separates the IBMs into two main subclasses with multiple variants [164]. Another disadvantage of IBMs stems from their difficulty to control mesh resolution at the vicinity of body surfaces, compared with conventional approaches. Since the mesh lines are not aligned with the body boundary, creating the high–aspect ratio poses a challenge. As such, IBMs require more dense Cartesian meshes to resolve the boundary layer. This problem becomes more pronounced as the Reynolds number increases since the boundary layer thickness reduces in size.

1.1.1 Immersed Boundary Methods Approaches

There is a multitude of ways in separating IBMs into subclasses. Mittal & Iaccarino [164] approach separates IBMs based on the way the forcing function is implemented. Sotiropoulos & Yang [238] classifies them based on the representation of the fluid–solid interface. Following Sotiropoulos & Yang [238] approach, IBMs are broadly divided into *Diffused* and *Sharp Interface Methods*. The former are characterized by the smearing of the immersed boundary and have its effects act upon several computational cells in its vicinity, while the latter avoiding interface smearing by tracking its exact location and applying its effect locally.

In diffused interface methods, the force functions are represented by discrete delta functions or penalization methods. Their exact implementation further differentiates them. In case these are introduced in the continuous form of the governing equations, the continuous forcing approaches arise, whereas, in the discrete ones, the force functions are incorporated during the discretization of the governing equations.

The continuous forcing approaches are ideally suited for Fluid—Structure Interaction (FSI) problems with elastic bodies since the exerted body forces can be computed using the appropriate law, for instance, Hooke's. Therefore, this approach is widely used in the field of bio–fluid mechanics; some indicative examples are the work of Peskin [196] and Griffith *et al.* [88], who study the human heart interaction with the blood flow, and Eggleton & Popel [72], wherein the deformation of red blood cells in shear flow is presented. The accuracy these methods is significantly influenced by the smoothed functions choice used to distribute the exerted body forces. An extensive review and applications on the continuous forcing approaches in the field of bio–fluid mechanics can be found in Peskin [197]. However, continuous forcing approaches

do not fare well when considering rigid bodies. The force functions significantly stiffen the governing equations and cause numerical instabilities. Workarounds have been proposed, such as approximating rigid bodies using stiff spring and, therefore, allowing a minute elasticity to exist [140], constraining the solid region to perform rigid motion [84] or assuming a porous medium [18]. Discrete forcing approaches were introduced by Mohd-Yusof [166]. The forcing functions are incorporated during the discretization of the governing equations and are computed based on the desired velocities at the body surfaces. They avoid the introduction of user–specified parameters and exhibit better numerical characteristics and stability compared to their continuous counterpart. Moreover, even though this approach is considered a diffused interface method, it can retain a distinct representation of the immersed boundary [39, 238]. However, this amplifies non–physical spurious force oscillations in moving bodies, see Uhlmann [258] and Breugem [39]. Fadlun *et al.* [74] extended the formulation to the finite differences method. Additional improvements are documented in Sotiropoulos & Yang [238] and references therein.

The main advantage of diffused interface methods lies in their implicit treatment of the immersed boundary, i.e. it avoids computing sub–cell information and limits the additional numerical requirements to the inclusion of the forcing function. However, their main disadvantage is their inability to accurately predict the associated boundary layer. Sharp interface methods explicitly track the body surface to overcome this and include it into the discretization scheme next to the solid boundary.

The first main approach of sharp interface methods is the *ghost–cell* method [254]. In such an approach, the solid boundaries are explicitly tracked and have the computational domain extended keep an additional zone of cells inside the solid regions, namely the ghost cells. The ghost–cells are then used to compute flux balances with cells in the fluid domain. Their values are computed using an interpolation scheme that implicitly incorporates the appropriate boundary conditions. Thus, the interpolation used to compute the ghost–cell values is of high importance and has led to the development of various techniques [76, 185, 231]. A consensus used to compute the ghost–cell values lies in the use of image points [165], namely points inside the fluid domain whose location is computed by mirroring the ghost–cells along the solid boundary.

Even though the above–mentioned IBMs have seen great success and application in a wide range of fluid problems, none preserves the conservation properties in the vicinity of the immersed boundary and, hence, the entire flow domain. Spurious production/loss of mass, mo-

mentum and energy can be observed at the immersed boundary [110, 122]. The conservative issue is abolished with an alternative unique IBM that partly escapes this categorization but can broadly be classified as a sharp interface IBM, namely the CC method [75]. The CC method relies on the generation of Cartesian meshes that are subsequently modified to conform with the immersed body surface, bridging IBMs and conventional body–fitted approaches. Figure 1.1 shows an illustration of the aforementioned meshing approaches. The CC method is exclusively used and extended in this thesis and is, thus, presented separately in the following section.

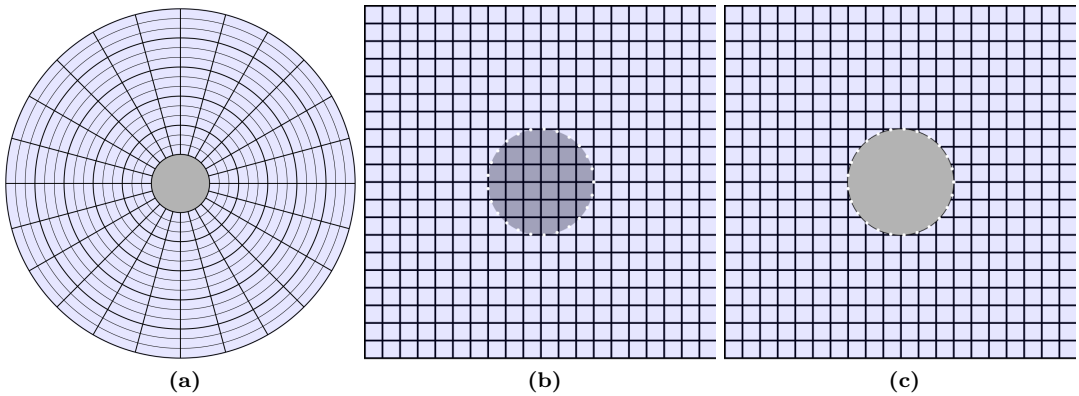


Figure 1.1: Illustration of meshing approaches around a cylinder - In conventional body–fitted approaches (a), the generated mesh is wrapped around the considered body surface. On the contrary, in IBMs, a non body–conforming Cartesian mesh is generated having the body surface immersed within it (b). In the CC method (c), a Cartesian mesh is generated and is subsequently modified to exclude the solid part by reshaping the intersected Cartesian cells. This leads to a body–conforming Cartesian mesh. Ciel coloring corresponds to the computational domain in which the governing equations are solved.

1.1.2 The Cut–Cell Method

The higher accuracy and conservation properties near the interface and high level of automated mesh generation capabilities make the CC method one of the most potent IBMs, suitable for turbulent simulation [161, 238]. As it has been already alluded to, in the CC method, the non–conforming Cartesian meshes are cut by the body surfaces, separating them into distinct fluid and solid regions. To do so, computational cells that encompass the immersed boundary retain only their fluid part to create finite volumes that conform with the boundary surface. The resulting 'Cartesian' mesh consists of the interior Cartesian and the boundary *cut* cells. Generating the cut cells poses one of the main challenges/drawbacks of the CC method due to the complex Cartesian cell–body surface intersections that can arise, even in $2D$.

Firstly introduced by Clarke *et al.* [55] for $2D$ inviscid flows and extended to $3D$ by Gaffney JR & Hassan [80], the CC method mainly focuses on finite volume discretizations and has been benefited from decades of development. These early works identify a problem that has since become synonymous to the CC method, the *small cell problem*. During the CC mesh generation, arbitrarily small CCs may arise that harm the stability and performance of the solution process. The presence of small CC has led to the development of several techniques that identify these small CCs and either physically merge them with a neighboring cell to create a *hypercell* [124, 279], or link/mix them by numerically connecting them with adjacent cells [58, 92, 168]. Another alternative lies in the *H–box method* [95, 156] that, even though it is second–order accurate, it is impeded by significant complexity and has not seen broad usage.

Following developments focused on *adaptive mesh refinement* techniques to avoid waste of computational resources and introduce refinement at areas of interest. In Quirk [205] and Pember *et al.* [194] mesh adaptation techniques are used to accurately capture shock waves, while in Melton *et al.* [158] a geometry–based approach is presented, wherein the Cartesian mesh progressively coarsens away from the body surface.

Extension of the CC method to the $2D$ Navier–Stokes equations was first attempted by Coirier & Powell [57], who investigated two different viscous flux discretization. These are applied in laminar flows with low to moderate Reynolds number to several test cases, including a coordinate–aligned flat plate, a non aligned one, and a branched duct with cooling fins. They conclude that the presence of the irregular CCs leads to a very oscillatory skin friction coefficient distribution when non mesh–aligned cases are studied, and stability concerns are raised, elucidating the challenges of extending the CC method to viscous flows. The stability issues are mitigated by Ye *et al.* [279] who introduced a polynomial interpolating function at the CCs to compute the required spatial derivatives to second–order accuracy, but is limited to the $2D$ Navier–Stokes equations. Similarly, in Hartmann *et al.* [92] a different viscous flux discretization is proposed that is second–order accurate, excluding the vicinity of the solid boundaries and extends the CC method to the $3D$ Navier Stokes equations. However, the challenge of computing a smoothly varying skin friction coefficient distribution is not tackled.

Several approaches have been developed to overcome this challenge ranging from the employment of strand meshes to altering the discretization scheme near the solid boundaries. The use of strand meshes arises from the taxing number of cells required to accurately resolve the boundary layer. This approach espouses the generation of a body–conforming mesh near the body surfaces that couples with a background Cartesian mesh [115, 275], similar to the inflation

layers used in conventional body–conforming meshes. This approach seems potent as it overcomes the accuracy issues mentioned and has seen significant improvements in automation and mesh resolution requirements. However, several challenges that persist include their application to complex/moving body surfaces, the automatic treatment of intersecting near–body meshes in multi–component body surfaces, and the additional interpolations used for intra–mesh communication [76, 115]. Approaches advocating using a Cartesian mesh up to the solid boundaries favor the simple and highly automated mesh generation process. The increased accuracy near the solid boundaries is pursued by increasing the discretization stencil [12], introducing polynomial fits [11, 31], following the principle presented in Ye *et al.* [279], or both [168]. For example, in Anagnostopoulos [11] different polynomials fits are suggested for the flow velocity and pressure. In Berger & Aftosmis [31] the velocity spatial derivatives between the CCs are computed using the *recentering* idea, proposed in [33, 109], and is coupled with a quadratic polynomial fit that computes the CC velocities and their spatial derivatives. In their work, a comparative accuracy study is performed comparing the recentering approach with the polynomial fit-based, revealing the higher accuracy of the former. Another noteworthy, more recent attempt is that of Muralidharan & Menon [168], in which they propose a new cell clustering algorithm that numerically merges the CCs with neighboring cells and averages the unknown flow variables in each cluster. This approach provides a smooth pressure and skin friction coefficient near the solid boundaries.

Extension of the CC method, and IBMs in general, to turbulent flows has been realized in the context of Wall–Modeled LES (WMLES) [85, 202, 269], DES [23], LES [52, 160, 162, 171], and DNS [65, 69] that inherently require very fine resolution. Interestingly, in such simulations, incorporating surface roughness is of high importance and, thus, the Cartesian–based methods can offer great flexibility [69]. However, in the RANS equations, turbulence is modeled to provide a more economical alternative, avoiding the stringent mesh resolution requirements. Still, a sufficiently fine refinement near the wall is required to resolve the turbulent boundary layer. However, unlike conventional body–conforming approaches, the isotropically refined Cartesian–based methods have difficulties in refining only along the wall–normal direction, leading to a rampant increase of the required number of cells [61]. Thus, wall–resolved RANS simulations become unjustifiably expensive [31, 32, 249]. The use of strand meshing can still be pursued but introduces additional complexities to the mesh generation process [115]. For this reason, approaches using pure Cartesian meshes, coupled with near–wall modeling, have seen significant research interest, invigorating near–wall modeling interest as well. This approach is motivated

by the reduced cell counts offered through the use of wall functions in body–fitted RANS and WMLES methodologies [252]. Challenges arise due to the first cells' irregular distances to the solid walls, but several recent approaches have been promising, showing significant cell count reduction, increased accuracy and smooth near–wall quantities. Limitations, such as those observed in wall functions, exist [212, 249], leading to the development of more sophisticated wall models [32, 46].

Hitherto the survey was limited to stationary body surfaces. In case moving objects are simulated, several CC–specific challenges need to be addressed. The clipping of the Cartesian mesh on the current body surface position leads to the emergence and disappearance of cells during unsteady simulations and, thus, temporal derivative discontinuity. This can result in mass loss and pressure spurious oscillations. Various attempts have been made to remedy this challenge, such as distributing the disappearing cells mass to neighboring cells [29, 217, 218] or developing space–time finite volumes [169], but a dominant technique benefiting from the flexibility of the CC method still remains. The pursuit of more advanced techniques is also being considered in the PhD thesis of Samouchos [214].

The CC method proves to be a powerful, scalable and flexible method that can deal with modern CFD–related challenges, also mentioned in the 2030 vision study [236]. Therefore, investing in the development of CC approaches that can handle flow problems of increased complexity is innate.

1.2 Cavitating Flows

Cavitating flows occur in engineering applications involving pumps, water turbines, nozzles and marine propellers. The continual bubble implosions that generate shockwaves and high-speed liquid jets near the solid walls have detrimental effects in these applications and are the reason for vibrations, load imbalances, material loss, noise pollution, and reduction in performance and life cycle [79, 130]. In contrast, cavitation turns out to be beneficial in some other applications, such as in drug delivery systems [241], underwater cleaning devices and vehicles. Cavitation has received significant research interest in the past decades. Experimental cavitation studies are challenging that require specialized equipment and measurement techniques to obtain reliable data [2, 38, 130, 243]. In Acosta & Hamaguchi [2], cavitation, and more precisely its appearance mechanism, is studied on the ITTC headform. Figure 1.2 exemplifies a case of sheet cavitation on the ITTC headform.

Numerical predictions are a faster and cheaper alternative that can provide details about the

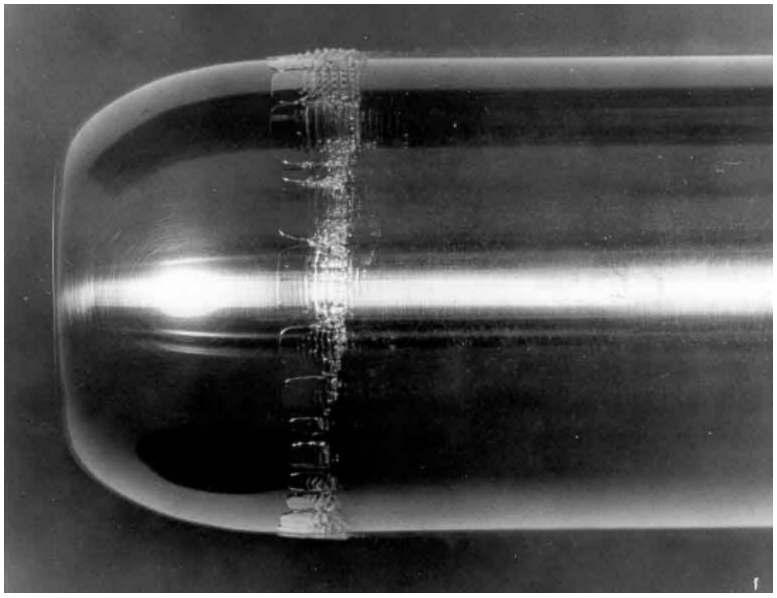


Figure 1.2: Experimental investigation of cavitation [2] - Sheet cavitation on the ITTC headform. The flow is from left to right with a speed of $12.2\frac{m}{s}$ and a cavitation number of 0.424. Image reproduced after written permission from the Oxford University Press.

onset and evolution of cavitation. Modeling challenges arise due to the interfacial dynamics, phase change and multiple time scales. Thus, single– or two–fluid numerical models with various levels of complexity exist. Two–fluid models are the less common approach; they treat the liquid and vapour phases as two different fluids [89] by adopting conservation laws (mass, momentum, energy) for both phases, providing generality at the expense of their computational cost [129]. The single–fluid, or homogeneous, models are based on conservation equations governing the homogeneous mixture, with averaged properties [153]. Away from cavitation bubbles, the mixture fluid behaves as a pure liquid. Instead, within them, its properties depend on the presence of vapour. Homogeneous models can further be classified as the Transport Equation Model (TEM), Barotropic [62] and Equilibrium [216] ones. For more details, interested readers are referred to Koop [129] and the therein cited papers.

Herein, simulations of cavitating flows are pursued following the TEM approach, wherein the mixture constituents (liquid and vapour) are considered incompressible fluids. The mass transfer rates are determined by the Kunz cavitation model [137]. Its advantages are its ability to generate baroclinic vorticity at the cavity closure region and its flexibility when dealing with multicomponent species. In TEM, species transport equations are solved for each mixture constituent mass or volume fraction, supplemented by the momentum conservation equations

of the homogeneous mixture. Cavitation, and thereby vapour generation and destruction, are modeled by explicit source terms in the transport equations. These are derived either from the Rayleigh–Plesset equation (RPE) [204], such as the Bubble Two–phase Flow (BTF) [135] and the Sauer [215] model, empirical approaches based on the evaporation and condensation processes, e.g. the Merkle [159] and Kunz [138] model, or the empiricism-free cavitation model based on interfacial dynamics of Senocak and Shy [223]. Cavitation models combining both bubble dynamics (RPE) and empiricism, such as the Zwart [281] and the Full Cavitation [234] model, exist too. The system of equations is solved in a coupled manner, using a preconditioner [138] based on the artificial compressibility method of Chorin [54] and implemented on adaptive CC meshes by extending the precursor software [214] to treat a mixture fluid with an additional transport equation for the liquid volume fraction and the cavitation model.

The advantages of the CC method, and their ability to locally increase mesh resolution, make it an attractive alternative to traditional body-fitted meshes. Furthermore, their ability to guarantee conservation near the solid boundaries reinforces their application, considering the sensitivity of cavity predictions to near–wall resolution [222]. In general, CFD simulations of cavitating flows have gained enough maturity to predict cavitating flows in complex applications, such as surface piercing hydrofoils [123], moving fuel injector control valves [180], water–jet pumps [145], cavitating propellers [167], external gear pumps [163], and supercavitating spheres [195]. However, relevant numerical optimization tools are rather limited, encouraging their development through the present thesis.

1.3 Numerical Optimization Tools

The establishment of CFD tools for performance analysis has motivated the development of numerical optimization procedures to facilitate industrial design needs. These can be classified into two main categories based on the strategy followed to iteratively compute the optimal design, namely the global (stochastic) [240] and local (deterministic) approaches [34, 178]. The former, also named zero–order methods, can operate in a *black–box* framework avoiding the computation of the objective function(s) gradient(s). These approaches are ideal for optimization problems with a cheap objective function evaluation process and have seen great success in constrained and multi–objective optimization problems. They are (mostly) population–based, i.e. starting with several arbitrarily generated candidate solutions, new candidate solutions are generated based on natural evolution operations, such as mutation and crossover, organized in generations. The introduction of randomized perturbations allows for increased exploration

properties enabling them to find the global minimum/maximum, given that sufficient computational resources are provided. Representative such approaches are the Genetic Algorithms (GAs), Evolutionary Strategies (ESs), Evolutionary Algorithms (EAs). The cost of these methods scales non–linearly with the number of design variables, i.e. degrees of freedom of the optimization problem, and are, thus, plagued by the *curse of dimensionality*. In CFD applications, the objective function is computed by solving a costly non–linear system of PDEs and often exhibit a large number of design variables, making the use of ingenuous approaches problematic. However, when refined, via surrogate or dimensionality–reduction models [112, 113], it enables their selective application [111, 139]. Altogether, this thesis is exclusively concerned with deterministic optimization methods. Hence, stochastic approaches shall not be further discussed.

Deterministic approaches embody all optimization algorithms that rely on the computation of the (local) objective function gradient w.r.t. the design variables (also referred to as Sensitivity Derivatives (SDs)), and in some cases second–order derivatives to obtain curvature information (*Hessian* matrix), to direct the optimization problem towards extrema. The use of local data, however, leads to their main disadvantage. They are dependent on the starting guess, can become trapped at local extrema and, thus, an optimized, instead of the optimal, solution. In addition, deterministic approaches have difficulties in multi–objective optimization problems. These disadvantages, however, dwarf their primary advantage, especially for CFD applications, due to the high–cost objective function evaluation. The additional information makes these approaches significantly more computationally efficient per objective function evaluation [78], since they march towards the extrema direction, continuously obtaining better performing solutions. An extensive review on deterministic optimization methods can be found in Nocedal & Wright [178].

The efficiency of deterministic approaches strongly relies on the objective function gradient computation efficiency. In the field of fluid mechanics, the objective function gradient cannot be computed analytically and requires the introduction of numerical methods [198]. The easiest, most straightforward approach is Finite Differences (FDs). In an optimization problem defined by its objective function J, controlled by the design variables $\boldsymbol{b}$ of size N, the objective function gradient $\frac{\partial J}{\partial \boldsymbol{b}}$ can be approximated by consecutively permuting each design variable ($\boldsymbol{e}$) infinitesimally by ϵ in both directions ($\pm$), i.e.

$$\frac{\partial J\left(\boldsymbol{b}\right)}{\partial b_i} = \left(\frac{J\left(\boldsymbol{b} + e_i\epsilon\right) - J\left(\boldsymbol{b} - e_i\epsilon\right)}{2\epsilon}\right) + O(\epsilon^2), \qquad i = 1, ..., N \tag{1.1}$$

The main advantage of this method is its ease of implementation. Since the re–evaluation of the objective function occurs with each design variable increase or decrease, this method can be considered a black–box method. However, the total cost of computing the objective function gradient scales linearly w.r.t. the number of design variables ($2N$ Equivalent Flow Solutions (EFS)), making it inefficient when considering multi–variable optimization problems. In addition, the objective function gradient approximation is sensitive to the permutation size selected. Large values lead to low accuracy approximations, while small ones can lead to roundoff errors due to minute denominators. The silver lining is usually found through trial and error, inadvertently increasing the total cost. Other alternatives exist that can overcome the ϵ–dependency, such as the Complex Variable [15], and the Direct Differentiation [227] methods but are hindered by significant implementation investment and linear, w.r.t. the number of design variables, scaling (N EFS).

An alternative method for computing the objective function gradient is found in the adjoint method, with a cost that is independent to the number of design variables. This independence makes the adjoint method suitable for large–scale optimization problems, commonly seen in CFD applications, outshining the aforementioned optimization methods. Thus, the adjoint method has been in the spotlight over the past decades by relevant academic and industrial communities. In the adjoint method, a dual problem is created by forming the Lagrangian, namely converting the constrained (by the governing equations) optimization problem to an unconstrained one that includes Lagrangian multipliers (adjoint variables). The Lagrangian shares the same extrema as the objective function and, therefore, the required gradient expression, comprising flow and adjoint variables, results after differentiating the Lagragian w.r.t. the design variables. To make the gradient expression independent of the derivatives of flow quantities w.r.t. the design variables that are associated with immense computational effort, a set of adjoint PDEs arises. The cost of solving the adjoint equations is similar to that of the governing equations. As such, the total cost of computing the objective function gradient is approximately equal to numerically solving the governing equations twice (2 EFS) and irrespective of the number of design variables.

1.4 The Adjoint Method

The adjoint method was first introduced by Lions [146] and was first applied in the field of fluid dynamics by Pironneau [203], who studied potential flows problems. Then, the adjoint method was formulated and applied to problems governed by the Euler equations [102, 105, 207]. In

the following years, the method was progressively extended [104, 117, 177] and refined [116], resulting in different objective function gradient expressions, and applied in fluid problems of increasing complexity, such as aircraft [208], automotive [182, 192], turbomachinery [188], and wind turbine [66] applications. A recent extensive review on adjoint methods can be found in Kenway *et al.* [119].

The adjoint method can be classified into two main approaches based on how they define the Lagrangian [48, 170, 198]. In the discrete approach [103], the Lagrangian is formulated based on the discretized governing equations, necessitating the differentiation of the discretized governing equations. This can be accomplished either *by hand* or using an Algorithmic Differentiation (AD) tool [87], such as TAPENADE [100], ADIFOR [19], ADOL-C [270], etc. The key benefits of the discrete approach lie in their consistency of computing the objective function gradient regardless of the coarseness of the mesh and their capability of partially automating the implementation of adjoint methods in CFD software [119]. Their main disadvantages spring from their severe computational and memory demands that, in some cases, are prohibitive.

In the continuous adjoint approach [14], which is the chosen approach in this thesis, the Lagrangian is formulated by the governing equations in continuous form and results in a set of (linear) adjoint PDEs that share similar numerical aspects with the governing equations. It enables the use of the same numerical solution process for the adjoint equations, reducing implementation and memory requirements. Development of the adjoint problem, in its continuous form, can result in different objective function gradient expressions. Employing the Leibnitz rule during the differentiation of the Lagrangian, the resulting gradient expression is dependent only on Surface Integrals (SI approach) [187] that have a negligible cost to compute. The SI approach is shown to provide lower accuracy computations since it only accounts for displacement variations at the boundaries [101]. Instead, if the Leibnitz rule is not used, the resulting expression is also dependent on Field Integrals (FI approach) [190] that require the computation of the computationally intensive mesh displacement variations throughout the computational domain but yield higher accuracy. The accuracy difference of the two approaches is attributed to the difficulty of computing boundary terms that arise from the Leibnitz rule. However, this is circumvented in Kavvadias *et al.* [118] wherein the Enhanced-Surface Integral (E–SI approach) is proposed that accounts for the mesh displacement variations via the concept of adjoint mesh displacement and solving an additional adjoint equation. Since, the CC method is used, for which mesh displacement variations exist only at the CCs, this thesis focuses on the SI approach.

Significant research has been invested in the implications of accounting for eddy viscosity variations when extending the adjoint method to turbulent flows. The inclusion of the turbulence models is often associated with significant implementation requirements that initially lead to the *frozen turbulence* assumption, purporting that eddy viscosity variations can be neglected and avoid its differentiation. Investigations of one– and two–equation turbulence models reveals that, in some flow problems, this assumption could lead to inaccurate gradient computations, in both magnitude and sign, significantly affecting the optimization path [71]. However, cases have also been reported in which differentiating the turbulence model could potentially be avoided, such as [154] wherein the frozen turbulence assumption has a minute impact on gradient accuracy. Overall, even though the effects of this assumption have been documented in both the continuous [42, 189, 282, 283] and discrete [120, 121] approach, it is yet not possible to a priori evaluate the impact of the frozen turbulence assumption. In light of the aforementioned, the development of the adjoint formulation in turbulent flows includes the differentiation of the turbulence model, implemented specifically in the CC method, to avoid cases of inaccurate gradient computations. Due to the intricate extension of the CC method to turbulent flows, several different/new aspects arise and are presented in this thesis.

In the literature, the development of the adjoint method is focused mainly on single–phase flows, whereas its application to two–phase flows is relatively limited. Adjoint two–phase methods incorporating cavitation effects are even fewer [36]. Early efforts applying the adjoint methods in two–phase flows focus on free–surface potential flows, minimizing wave resistance of ships and submarines [206]. In Palacios *et al.* [183], the continuous adjoint method to the two–phase Euler equations supported by the level set method is presented; these are extended to the RANS equations in Palacios *et al.* [184] and are applied in obstacles in free surface channel flows to minimize the wave generated by the obstacle at the free surface. More recently, a hybrid adjoint approach focusing on the derivation of the adjoint Volume of Fluid (VoF) equation is proposed [134, 136] and employed for the resistance minimization of a $3D$ container ship hull. Furthermore, in Alexias & Giannakoglou [7] the continuous adjoint method is implemented and applied in the optimization of a static mixing device. Applications of adjoint methods in cavitating flows can be found in Boger & Paterson [36], wherein they develop the continuous adjoint method for the barotropic mixture model to compute gradients of various objective functions for inviscid, cavitating flows. Moreover, in [35] a quasi$-1D$ TEM, the continuous adjoint formulation is derived for several objective functions, investigated in terms of suitability and effectiveness for a series minimization problems. More recently, relative work can be found

in Anevlavi & Belibassakis [17], in which the continuous adjoint method is introduced through an inverse problem to improve upon the predicted cavity features of panel method potential flow solutions. Based on the above, the derivation of a continuous adjoint method for inviscid and laminar flows exhibiting cavitation, modeled by a TEM using the cut-cell method, has yet to be formulated and implemented for Shape Optimization (ShpO). Thus far, ShpO methods accounting for cavitation, excluding [35, 36], are treated using a single–phase approach with an objective function that quantifies areas of pressure below the vapour pressure. However, in such approaches, much of the underlying physics is neglected since cavitation does not physically occur, rather, it is inferred. The single–phase approach introduces limitations with regards to the objective function expression, as well as the inability to quantify two–phase effects. The arising challenges are two–fold. Firstly, mass transfer rates must be differentiated according to the selected cavitation model and included in the adjoint two-phase solver. Secondly, variations of the molecular density and viscosity that appear due to the two-phase nature of the flow need to be incorporated.

1.4.1 The Adjoint Cut–Cell Method for Shape Optimization

Typically, during the ShpO algorithm, the computed objective function gradient guides the design variables that control the considered body surface to provide better performing designs. In each subsequent optimization cycle, the current body surfaces change along with the generated body–conforming mesh, similar to problems with moving objects. Thus, mesh deformation techniques are necessary to adjust body–conforming mesh for subsequent simulations. However, they come with limitations, namely the inability to deal with large displacements and gradually worsening mesh quality. On the contrary, in the CC method, and IBMs in general, the mesh generation process enables a fully automated re–meshing procedure and facilitates displacements of arbitrary magnitude. Indeed, modifying the body surface in ShpO is effortless when compared to cases with moving objects since the appearance and disappearance of finite volumes need not be treated. Additionally, the absence of mesh deformation techniques circumvents the necessity of computing mesh deformation variations away from the body surface and the accompanying increase in computational cost.

Even though adjoint CC–based implementations can provide significant advantages, the related literature is relatively sparse and confined to a single research group. The first adjoint CC–based application is realized in Nemec *et al.* [175] who implemented the discrete approach by hand–differentiation and studied transonic and supersonic steady–state flow problems gov-

erned by the $3D$ Euler equations. Their work mainly focuses on validating the discrete adjoint formulation by comparing the obtained objective function gradient with finite differences but performed no ShpO. In a subsequent study by the same group [173], shape sensitivities are computed by differentiating the CC generator and are incorporated into the adjoint formulation to perform ShpO. A later study [174] focuses on the implications of neglecting the gradient limiter in the discrete adjoint formulations by assessing the gradient accuracy in test cases that require no limiting and show good agreement with FDs, and test cases that require the use of a limiter and exhibit significant deviations. These deviations are, thus, construed as the omission of the limiter in the adjoint formulation and signifying its importance.

1.4.2 Topology Optimization in Fluid Mechanics

Topology Optimization (TopO) [28] involves the pursuit of obtaining a better–qualified design for a specific application by altering the material composition within its design space while also satisfying the governing equations and design constraints. Introduced and flourished in solid mechanics by Bendsøe & Kikuchi [27], TopO has been extended to various scientific fields due to the flexibility it offers, as it is not bounded by an a priori defined connectivity. Instead of explicitly modifying the shape of a given body surface as in ShpO, its main idea is to modify the material composition of the computational domain (density, for instance) to describe solid and void (empty) regions. In fluid mechanics in particular, TopO approaches are first realized by Borrvall & Petersson [37] and deal with Stokes flows, but have been extended and applied to laminar [148, 181], and turbulent [68, 128], steady, and unsteady [63, 176, 264] flows. The predominant method used for TopO in fluid mechanics is analogous to that in solid mechanics, namely via the porosity–based approach (or else the Brinkman penalization method). In this approach, a porous material is introduced and has its permeability α in each computational cell or node controlled. Solid regions are characterized by low porosity or high impermeability, impeding fluid flow by enforcing a zero velocity, whereas fluid regions are characterized by high porosity, allowing the fluid to move freely. TopO, seeks to answer which computational cells should *solidify* and which should remain porous by altering their porosity in a finite range, usually $0 \leq \alpha \leq 1$, to minimize the objective function. This is illustrated in Figure 1.3. As such, in TopO, the design variables consist of the porosity of every cell or node of the computational domain, making adjoint methods ideal. Furthermore, the optimization problem is usually subject to additional (in)equality constraints, such as the maximum allowable fluid region volume or specific massflow at the outlets [191, 264]. Numerically, TopO problems are

characterized by a very high number of bounded design variables and additional constraints, leading to a complex optimization process that calls for more advanced optimizing procedures.

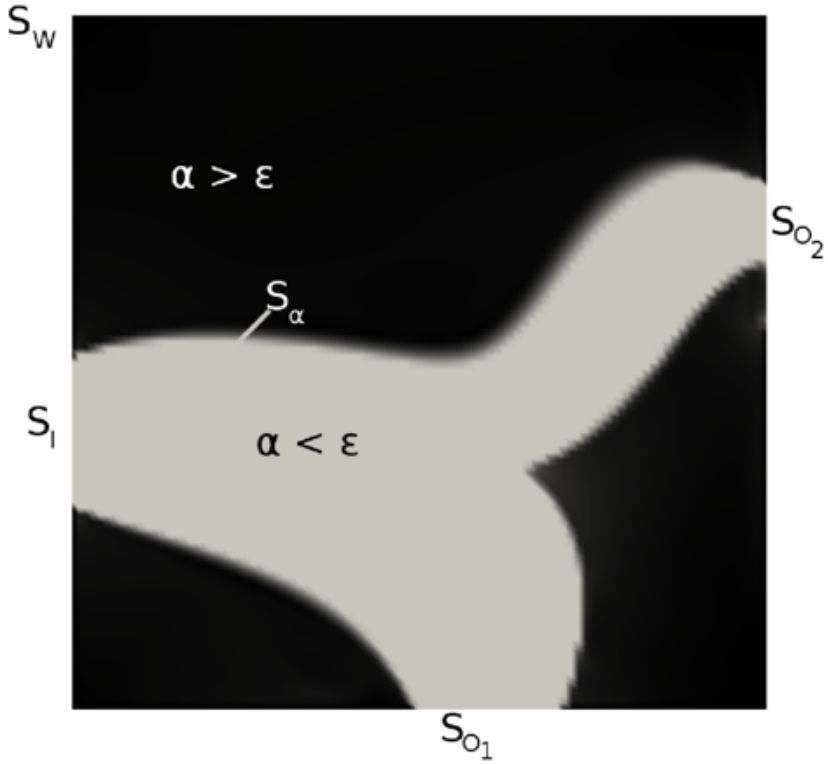


Figure 1.3: Schematic representation of the porosity–based approach - The dark region corresponds to areas with low porosity (high impermeability) that imitate the presence of solid, while gray regions are characterized by high porosity (low impermeability) and, thus, imitate fluid regions*. Note however that the interface between the two regions is implicitly described and requires additional post–processing to extract the shape of the resulting duct. Taken from Papoutsis-Kiachagias [191].

The ability to radically change the topology of the solution during the optimization cycles and the relatively simple implementation are the main advantages of porosity–based TopO. Its main weakness relates to accuracy issues, as it cannot account for the accurate effect of solid walls on the computed flow solution and regions of intermediate porosity (partly fluid/solid) can appear, breaking the distinction between the two. Usually, (porosity) interpolation functions are introduced to partly mitigate this weakness. Furthermore, stair–case effects instead of smooth curved boundaries can appear, influencing the solution accuracy. Another difficulty of

*Without being pedantic, the design variables α in porosity–based approaches characterize the porosity of each cell (or node) by describing its impermeability. As such, high α suggest an impermeable material that does not allow the fluid to pass through it and, thus, absence of porosity.

porosity–based approaches lies in the inability to simulate narrow solid regions with accuracy as pressure diffuses through them [131, 151]. For these reasons, optimized solutions computed by porosity–based TopO are usually followed by a re–evaluation process using flow solvers running on body–fitted meshes. However, the extraction of the body shape from the optimized porosity field is not trivial and may, among others, impair the quality of the optimized solution. This process usually leads to a (slightly) different performance value that is, however, more accurate. For this reason, combined methods, such as Koch *et al.* [127], are being developed that can parameterize the so–extracted boundaries and follow–up a ShpO to further refine the topology–optimized solution.

The shortcomings of the porosity–based approaches in conjunction with the demand of running more complex TopO applications that necessitate sharp fluid–solid interfaces [107, 114, 151, 268], have rejuvenated surface–capturing approaches. In these approaches, solid boundaries are reconstructed during the optimization process rather than upon its completion, usually via level-set computations on fixed background meshes [49, 70, 131]. Capturing the solid surface anew within each optimization cycle introduces several benefits. Firstly, it enables the imposition of accurate boundary conditions. Secondly, it can improve upon turbulent TopO applications for which near–wall accuracy is of the essence. Thirdly, it enables objective functions to be expressed directly at the solid boundaries, such as heat transfer or boundary deformation. In level–set TopO methods, the SDs, expressed w.r.t. the distance to the interface (ϕ), are computed and used to convect the zero–level iso–surfaces; these support topological changes such as boundary merging, splitting and disappearance [43, 259]. However, for the generation of new solid boundaries, i.e. the appearance of new fluid–solid interfaces inside the fluid domain, additional treatment is required [10, 213]. Naturally, level–set approaches are also afflicted with some disadvantages. For instance, they generally require the solution of the *Eikonal*, or Hamilton–Jacobi Equation (HJE), to compute the signed–distance function that describes the fluid–solid interface exact position, which translates to numerically solving a stiff PDE that is well–known for its strict stability properties and small timestep requirements. This process introduces computational overhead and can lead to slow optimization convergence rates. The associated computational costs can be reduced via a Narrow Band technique that updates the level–set to a Signed–Distance function at a small zone near the fluid–solid interface [127, 224]. Other alternatives also exist, such as the use of parametric level–set formulations that entirely circumvent the need of numerically solving a PDE [60, 201, 273].

A more traditional usage of TopO application has led to advanced Finite Element Method

(FEM)–based TopO approaches combining the IBMs and CC concepts with FEM discretizations, in an attempt to circumvent mesh generation of complex body surfaces and have seen application in the field of fluid mechanics. Such approaches are found in the eXtended Finite Element Method (XFEM) [81, 108, 131] and CutFEM [44, 45, 264] approaches, which are the IBMs and CC method counterparts adapted for FEM discretization. These approaches were developed to overcome mesh generation challenges for complex body surfaces and have found extensive TopO applications in both fluid and structural mechanics (and FSI problems) as an alternative to the porosity– and density–based approaches. In XFEM, the interface conditions along the intersection are accurately enforced by locally using enrichment functions to capture the discontinuous solutions within the finite elements. In CutFEM face–oriented penalty methods are also included to stabilize the formulation of the flow equations. Their use in TopO applications is combined with a level–set description of the fluid–solid interface, allowing them to solve the governing equations only in the fluid domain to reduce the associated computational cost. However, such TopO approaches that are based on the Finite Volume Method (FVM) discretization have not been yet developed and constitutes another contribution of the current thesis.

1.5 Thesis Outline

The CC method proves to be a powerful and flexible method that can overcome modern CFD–related challenges by introducing automation in the analysis and optimization framework for both ShpOs and TopOs. Motivated by the literature gap that exists regarding the application of the CC method to turbulent and cavitating flows and their adjoint–based counterparts, the development of computational methods are pursued to extend their applicability to these flow types. The integration and parallelization of the wall functions technique along the irregular CCs is investigated and assessed along with the usage of a cavitation model. Furthermore, to utilize the CC method in TopO applications, a novel accurate method that relies on a modified CC mesh generation process is developed and assessed. The new method is prompted by a preliminary comparative study between two flow analysis methods, namely the Brinkmann and CC, that illustrates the need for more accurate TopO formulations, especially when near–wall effects are of interest. The Globally Convergent Method of Moving Asymptotes (GCMMA) that proved to be superior to the Augmented Lagrangian Method (ALM) in both convergence rate and ease of constraint satisfaction performs the design variable updates. A byproduct of the development of a new TopO method that had to be assessed through fair comparisons with

porosity–based approaches is a porosity–field re–evaluation tool that can track the fluid–solid interfaces. All of the above are integrated on the precursor CC–based software to provide a general CFD–oriented analysis and ShpO and TopO tool. The thesis is structured as follows:

In Chapter 2, a brief, high–level description of the main components that constitute the CC method are presented. These include the method used to recursively decompose the computational domain into non body–conforming Cartesian cells that exhibits low memory requirements and to generate and describe valid body–conforming CCs of any complexity, both to be used as finite volumes during the flow solution process. Furthermore, details about adaptive techniques used to refine the resulting CC mesh quality are presented along with the cell–merging used to treat the *small cell problem*. Finally, this chapter describes the resulting mesh decomposition approach that allows for a parallel flow solution process and touches upon several parallelization aspects.

Chapter 3 is concerned with the flow models employed for the analysis of fluid problems throughout this thesis. Initially, the governing equations for single–phase, incompressible flows are described to acquaint readers with the existing FVM discretization aspects and numerical solution process employed in the CC method. These are used as the baseline methods for the extension to turbulent and two–phase cavitating flows. A brief theoretical background and the approach used to integrate these extensions follows. To this end, the necessary modifications, such as the CC–specific wall function technique used or the choice of cavitation model are also reasoned. Finally, $2D/3D$ test cases are studied that aim to assess and validate all of the CC software extensions.

In Chapter 4, the mathematical formulation of the continuous adjoint method for turbulent, two–phase flows exhibiting cavitation based on the CC method is presented. Initially, the objective function is augmented to form the Lagrangian and introduce the adjoint variables using a general objective function expression. Subsequently, a term–by–term analysis is carried out to derive the PDEs and boundary conditions for the adjoint problem and leads to a computationally cheap objective gradient expression that consists of only boundary integrals. Finally, the objective function expressions considered in this thesis are documented, along with their differentiation that is included in the adjoint problem.

Chapter 5 is concerned with the application of the developed continuous adjoint formulation in a ShpO workflow. Initially, a method that links the body–conforming Cartesian mesh with the discretized body surface description is described; this enables the accurate computation of geometric sensitivities without perturbing the body surface. Then, the ability to accurately

compute the considered objective function gradient is demonstrated through comparisons with FDs, and ShpOs are performed that also exemplify benefits of a CC approach.

Finally, in Chapter 6, a new CC–based TopO approach is proposed that is inspired by the independent, from the body surface, mesh generation process of the CC method. The new approach is also motivated by the ability of the CC method to enforce boundary conditions precisely on the fluid–solid interface, illustrated through comparative flow analysis accuracy study between the CC and porosity–based method. In the CC–based TopO, an optimization problem, similar to that used in typical porosity–based approaches, is defined but has the design variables mapped onto intermediate variables that can implicitly describe the fluid–solid interfaces. Hence, since in TopO no body surface description is provided, the process of generating the CCs is completely modified to *cut* on the fluid–solid interfaces. Aiming to assess the developed CC–based TopO, a standard porosity–based approach is used to make comparisons on several benchmark cases. Finally, a $3D$ TopO problem is studied, with an intent to showcase its ability to deal with large scale applications.

Additionally, 4 appendices support the theoretical background and implementation details. More precisely, Appendix A supplements Chapter 3 by providing the forms of the inviscid Jacobian matrices and eigenvectors for the employed flow models. Appendix B is concerned with the mathematical development of the boundary term that arises during the adjoint formulation in Chapter 4 that requires the differentiation of the implemented wall functions technique. Appendix C exemplifies the concept and provides implementation details of the GCMMA algorithm that proved to be indispensable for the iterative solution of the TopO problems in Chapter 6. Finally, in Appendix D, the process of re–evaluating porosity–based optimized solutions on CC–based body–fitted meshes, used in Chapter 6, is presented.

Chapter 2

The Cut–Cell method

This chapter outlines the unique features of the CC method [164], a subclass of IBMs, being one of the core aspects this thesis builds upon. It refrains, however, from entering into deep algorithmic descriptions since these were developed in the PhD thesis by Samouchos [214] which has recently been accomplished by the same group (Parallel CFD & Optimization unit (PCOpt)). The chapter focuses on the fundamental building blocks that result in an automatically generated, geometry–adapted Cartesian mesh, suitable for solving the governing equations around bodies of arbitrary complexity using the FVM. These include the decomposition of the computational domain into Cartesian cells, extra refinement techniques used to improve upon the generated CC mesh quality, and the cell–cutting procedure followed to create the CCs from the Cartesian cells intersected by the body surface. In addition, CC–specific challenges concerning mesh generation robustness, accurate representation of the boundary surfaces, handling of tiny CCs, and adaptive mesh refinement that significantly reduces the number of cells [47] are discussed. Finally, several remarks regarding the relevant data structure and the mesh decomposition process for the purpose of parallel computations are made.

The CC mesh generation process initially defines and decomposes the computational domain using a uniform Cartesian mesh. Cartesian cells close to the body surface are then refined, leading to a non–uniform Cartesian mesh that progressively coarsens away from it. CCs are created by demarcating the non–fluid parts of Cartesian cells intersected by the body surface. The CC mesh comprises the so–defined CCs and the Cartesian cells lying entirely into the fluid domain.

2.1 Cartesian Mesh Generation

The generation of the Cartesian mesh begins, assuming that the body surface is given in STereoLithography (STL) format. The rectangular cuboid (hyper–rectangular) computational domain that encompasses the provided body surface, is defined by its barycenter $\boldsymbol{C}_0$ and dimensions $\boldsymbol{d}_0$ and coincides with the root Cartesian cell. In the next step, the root cell, and all subsequently generated Cartesian cells, are isotropically subdivided into *child* cells of higher refinement level (4 cells in $2D$ and 8 in $3D$) with the same aspect ratio, creating a hierarchical parent–child tree structure. Initially, all Cartesian cells should reach a predefined refinement level, corresponding to a uniform Cartesian mesh. All subsequent divisions either depend on the body surface discretization by limiting each Cartesian cell to encompass at most two surface nodes or the user–defined cell lower volume bound. In the latter case, cells that are intersected by the body surface must subdivide until the lower volume bound be nearly met, but not outstepped, since in most cases the isotropic refinement makes the exact lower bound unreachable. Similarly, non–intersected cells must subdivide until the upper bound is no longer exceeded, except for cases in which neighboring cells' refinement levels differ by more than one. This is necessary to both balance the generated tree structure and to avoid steep changes in mesh resolution. This process finally creates Cartesian meshes, which are adequately refined close to the immersed solid bodies and coarsen progressively far from them.

To further clarify the $3D$ Cartesian mesh generation process, the mechanism recursively generating the tree structure and the additional conditions imposed to refine the generated Cartesian mesh are briefly discussed.

2.1.1 Recursive Octree Generation

One of the main aspects of the Cartesian Mesh generator is its use of an Octree (Quadtree in $2D$) data structure [51] to automatically decompose the computational domain into Cartesian cells of different refinement levels, Figure 2.1. Starting from the root Cartesian cell, isochoric subdivisions create smaller Cartesian cells, referred to as children cells. To accomplish that, during the decomposition of the computational domain, each Cartesian cell is labeled using a special integer indexing system (i, j, k) [4, 47, 214], used to traverse the hierarchical tree structure, locate the position of each Cartesian cell, and compute the necessary geometrical quantities.

The root Cartesian cell is given the index $(1, 1, 1)$ and, then, each child's unique index is

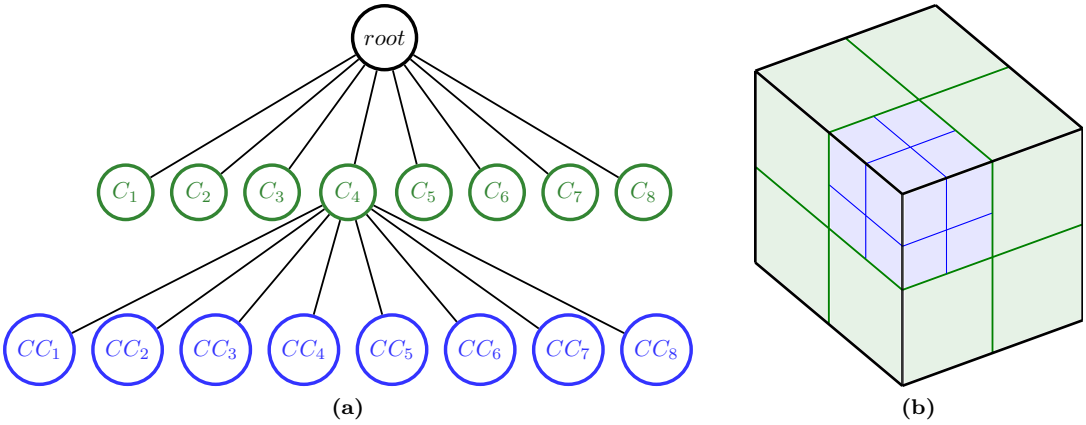


Figure 2.1: Octree Data Structure - (a) Graphical and (b) visual representation of the recursive subdivisions occurring when generating the Octree data structure. The root cell (black) is subdivided into eight children (green). For illustrative purposes, the fourth child C_4 is then subdivided into eight more children (ciel).

obtained considering its local position w.r.t. its parent cell and its parent index. As shown in Figure 2.2, each cell is divided into 8 *octants* and appointed a local index (i_l, j_l, k_l) that is used compute its (unique) global index (i_c, j_c, k_c). The exact recursive rule that computes unique indices for each child is

$$(i_c, j_c, k_c) = (2i_p + i_l, 2j_p + j_l, 2k_p + k_l) \tag{2.1}$$

where subscripts c, p, l denote child, parent, and local, respectively. As a result, the index of each cell's parent is found using

$$(i_p, j_p, k_p) = \left(\left\lfloor \frac{i_c}{2} \right\rfloor, \left\lfloor \frac{j_c}{2} \right\rfloor, \left\lfloor \frac{k_c}{2} \right\rfloor \right) \tag{2.2}$$

where $\lfloor x \rfloor = \max(n \in \mathbb{Z} \mid n \leq x)$ denotes the floor function, i.e. the greatest integer less than or equal to x. Considering that, at each subdivision, the parent cell indices are computed by dividing by 2, Eq. (2.2), it follows that the refinement level L of each Cartesian cell, i.e. the necessary subdivisions starting from the root cell to reach that descendant, is given by

$$L = \lfloor \log_2(i) \rfloor = \lfloor \log_2(j) \rfloor = \lfloor \log_2(k) \rfloor \tag{2.3}$$

Eqs (2.1), (2.2), and (2.3) arise from the recursive nature of the tree structure, and are useful since the connectivity of each Cartesian cell can be found by its index. Building upon the last

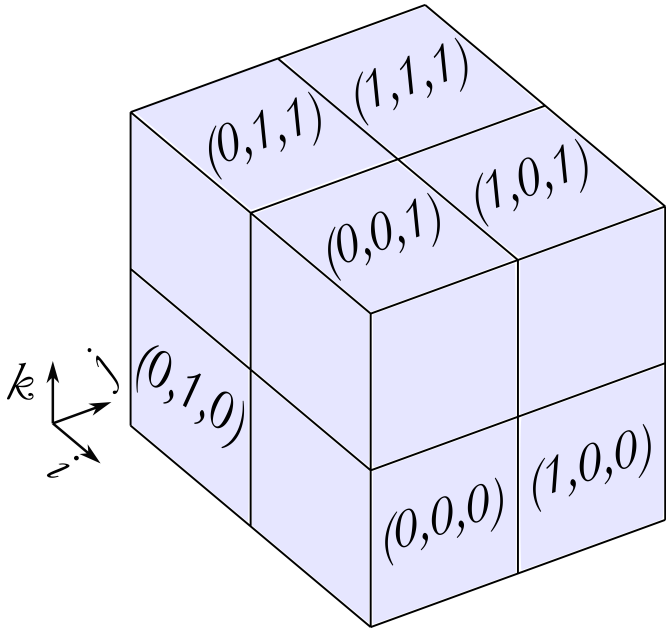


Figure 2.2: Subdivision of a Cartesian cell - The local position of each subcell/child used to compute their unique index.

remark, the integer indexing also allows for the computation of several geometrical quantities of the Cartesian cells. The edge size $\Delta\mathbf{x}$ and barycenter $\mathbf{x}_c$ of each cell can be computed using

$$\Delta\mathbf{x} = \frac{1}{2^L}\boldsymbol{d}_0 \tag{2.4}$$

$$\mathbf{x}_c = \boldsymbol{C}_0 - \frac{3}{2}\boldsymbol{d}_0 + \begin{bmatrix} i+\frac{1}{2} \\ j+\frac{1}{2} \\ k+\frac{1}{2} \end{bmatrix} \circ \Delta\mathbf{x} \tag{2.5}$$

where $\boldsymbol{C}_0, \boldsymbol{d}_0$ are, respectively, the root cell barycenter and dimensions and $\circ$ denotes element–wise multiplication (Hadamard product). Subsequently, additional geometrical quantities such as the Cartesian cell volume, edge area and barycenter, and the coordinates of its vertices can be computed in a straightforward manner.

2.1.2 Body Surface Subdivisions

The procedure dictating the body surface subdivisions follows a fast and robust tagging algorithm. Starting from the root cell, a list containing all body surface triangles that are encompassed within or intersected, following a series of single–elimination geometric checks, is generated. After each subdivision, the same series of checks is performed on the inherited list for each subcell, discarding triangles that are not encompassed or intersected. Thus, after

each division, the inherited list decreases in size until either the list is emptied or the Cartesian cell volume reaches the lower size bound, terminating the process. The benefit of using the hierarchical data structure is evident since the list of triangles quickly wanes based on the refinement level, minimizing the subsequent geometric checks. A thorough description of the involved geometric checks along with the method used to prioritize the computationally less intensive ones is given in Samouchos [214].

2.1.3 Cartesian Cell Post–refinement

The resulting Cartesian mesh, generated when considering only intersections of the Cartesian cells with the body surface, does not account for a number of features that are natural in conventional body–conforming computational meshes. More specifically, the generated Cartesian mesh shown in Figure 2.3 does not feature a smooth cell size change; the cell size increases by a factor of 2 in each direction between refinement levels. Therefore, the shown mesh could introduce notable numerical errors in the presence of large flow gradients that can appear, for instance, in the wake of solid bodies, the development of the boundary layer, or at shock waves [5, 47]. To overcome these issues, two additional refinement conditions, that can act independently or synergistically, are used.

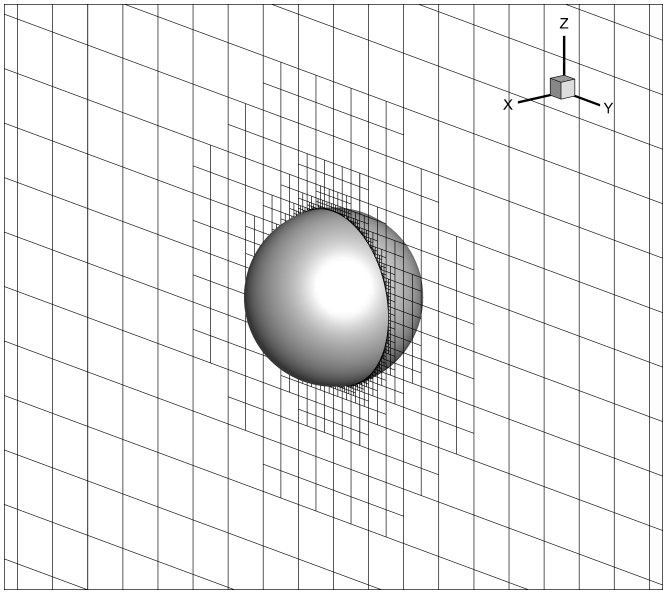


Figure 2.3: Generated Cartesian mesh around a sphere - Extracted yz–plane slice of the Cartesian mesh using body surface refinement only.

The first condition implements a distance–based sensor that requires the additional refine-

ment of Cartesian cells close to the body surface. For this step, Cartesian cells that are not intersected by the body surface are required to be decomposed into smaller ones to meet a new cell upper volume bound V_l, computed based on their distance to the body surface. This ensures a smooth transition between the refinement levels up to a cut–off distance r_0. V_l is computed for each cell through a cubic interpolation between the volume of the Cartesian cells intersected by the body surface and the original cell upper volume bound. Computing the exact distance of each Cartesian cell barycenter from the body surface is associated with a high computational cost and offers no significant advantage. Thus, the distance between each Cartesian cell barycenter and the body surface is approximated using a marching front technique. The initial front is formed by the intersected Cartesian cells, that have their distance from the body surface prescribed to zero. Then, the front is *marched* to their neighboring cells, which have yet approximated their distance from the body surface. These cells find their closest neighbor, that belongs to the previous front and has its distance from the body surface already computed, and cumulatively adds the Euclidean distance between their barycenters to approximate their distance from the body surface. Even though the cell size change between refinement levels still remains the same, the additional subdivisions regularize the smoothness of the Cartesian mesh by creating ordered *buffer* zones of equivolume cells [5], shown in Figure 2.4. The additional mesh resolution allows for more accurate computations of large flow gradients close to the body surface [47].

The second condition applies a window–based refinement to impose subdivision on the encompassed Cartesian cells. In effect, a bounding box is prescribed inside the computational domain, with a new upper and lower volume bound overriding the original ones (Section 2.1). Thus, Cartesian cells within the bounding box must split into smaller cells to reach the new volume bounds. This condition is used to accurately simulate the wake behind a body or to increase mesh resolution locally [47], Figure 2.5. In this thesis, the window–based refinement is used to capture the wake of air/hydrofoils and at the location of the expected cavity locations.

2.2 Cut–Cell Mesh Generation

The creation of the CCs, starting from the previously generated Cartesian mesh, is a key part of the CC method. As already mentioned, the CC method is appealing since its mesh generation process can facilitate a high level of automation, even when considering complicated body surfaces. It is, therefore, only natural that the cell–cutting procedure is robust and fail–safe. Typically, and especially in $2D$, the majority of cell(plane)–triangle intersections lead

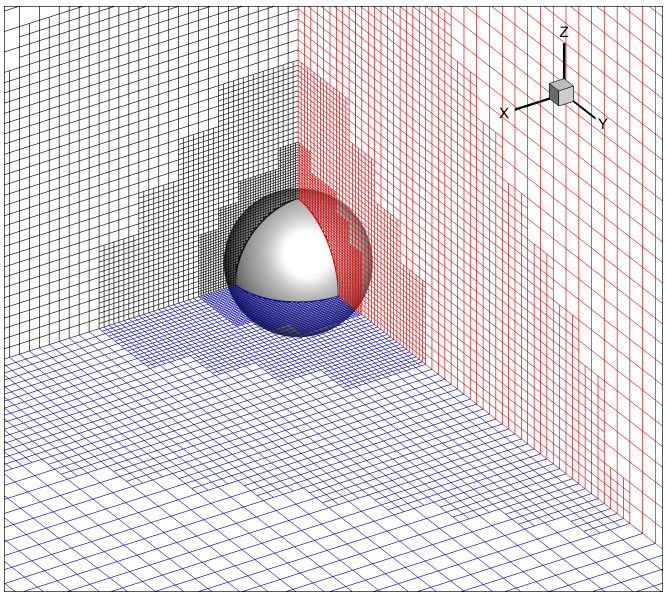


Figure 2.4: Generated Cartesian mesh around a sphere - Application of the distance–based condition results in the additional refinement of the Cartesian cells in some ordered zones of equivolume Cartesian cells (cf. Figure 2.3). Extracted slices of the Cartesian mesh along the yz–plane (red), zx–plane (black), and xy–plane (blue).

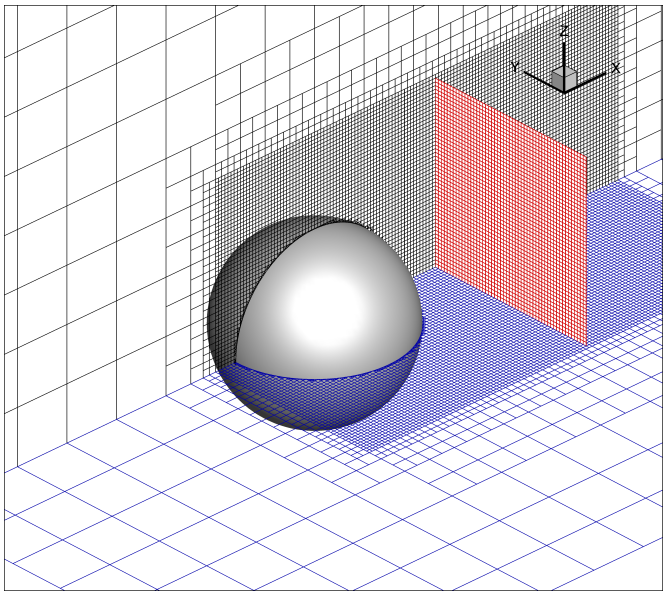


Figure 2.5: Generated Cartesian mesh around a sphere - Application of the window–based condition results in the increase of mesh resolution at the specified boxed region (cf. Figure 2.3). Extracted slices of the Cartesian mesh along the yz–plane (red), zx–plane (black), and xy–plane (blue).

to few additional faces and nodes resulting in CCs that are characterized by a rather simple topology. However, several cases often arise that result in very complex CC topology and require special attention [157, 180]. For instance, cases arise where the intersected cell leads to the creation of two separate CCs. These cases are usually treated with additional refinement that inadvertently leads to increased computational resources requirements without fully resolving the problem or by simplifying (smoothening) the body surface [56, 161, 162]. The CC mesh generation process developed in [214], and employed in this thesis, recognized limitations of such approaches and aims to handle intersections of arbitrary complexity, accurately representing the provided body surface. Inability to do so would lead to non–watertight meshes, blowing–up the mesh generation process, or even erroneous representations of the body surfaces.

2.2.1 Construction of Cut–Cells

The cell–cutting algorithm that creates the CCs follows a two–step procedure. First, each intersecting triangle is clipped inside the Cartesian cell. To do so, the Sutherland–Hodgman algorithm [5, 244] is used to compute the new polygon that is entirely encompassed within the Cartesian cell. In this algorithm, the faces of the Cartesian cell act as *clippers* that create two half–spaces. By successively evaluating each face, a new polygon is created consisting of the nodes that share the same half–space with the Cartesian cell and the intersection points of the current polygon with the current *clipper*. This process requires the relative position of each node w.r.t. each clipper. Space is split into 3 regions per dimension, viz. 9 in $2D$ and 27 in $3D$ and each region is represented by an *outcode* encoding that positions each node into one region, similar to the one used in the Cohen–Sutherland algorithm [77], to avoid multiple spatial comparisons. This step is repeated for each intersecting triangle and results in adjoining polygons that also constitute the solid faces of the CCs. This procedure is illustrated in Figure 2.6, where a Cartesian cell is intersected by 7 STL triangles at different angles.

In the next step, the CC is created first by identifying the solid and fluid parts of the Cartesian cell. Fluid and solid faces are created by connecting the computed intersection points with the Cartesian cell edge points. To do this, fluid polylines are created along the Cartesian faces comprising subsequent line segments of intersection points that arise from clipping the same intersecting triangle until a Cartesian edge is reached. These ordered polylines stand for the exact intersection of the provided triangulated body surface with the Cartesian cell face. At this point, duplicate intersection points, if any, are removed. Then, the Cartesian cell vertices lying in the fluid part of the CC are identified as 'fluid' using the obtained polylines, while the

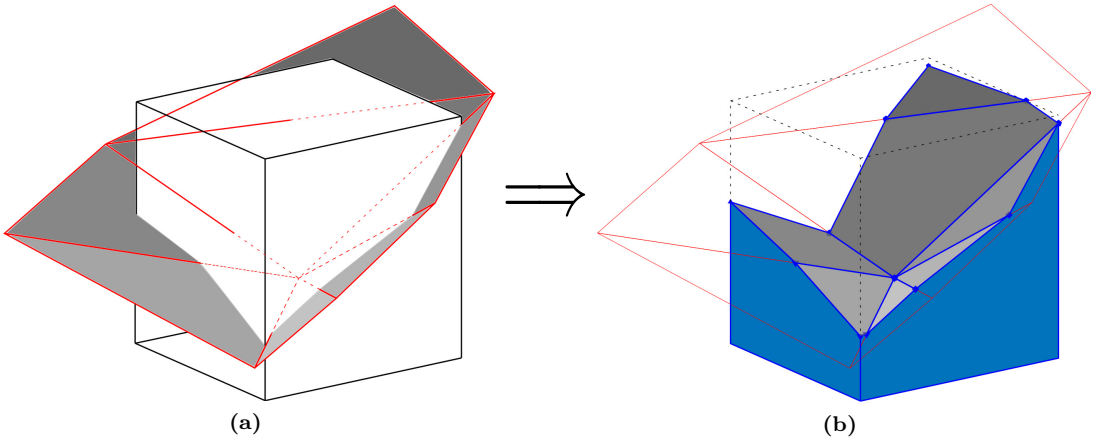


Figure 2.6: Implementation of the Sutherland–Hodgman polygon clipping algorithm - (a) The Cartesian cell is intersected by 7 differently shaded triangles. The dotted part of the triangle edges resides inside the Cartesian cell. (b) The clipping algorithm is performed for each triangle that results in 7 adjoining polygons, encompassed within the Cartesian cell. Each of these polygons represents a solid face of the CC.

remaining are identified as 'solid'. Finally, the CC fluid faces are created by connecting Cartesian cell fluid vertices and polylines on the same Cartesian cell face. This process results in CCs of arbitrary size and number of faces that can facilitate cases of split cells, non–convex polyhedra, as well as void cuboids. More details regarding the implementation of the aforementioned can be found in Samouchos [214] along with several illustrative examples.

2.2.2 Geometric Quantities of the generated Cut–Cells

The generated CCs will subsequently be used as finite volumes to solve the flow equations numerically. This necessitates the computation of several face–related geometric quantities, $\mathcal{X}^f$, where $\mathcal{X}^f = \left\{\overline{\mathbf{x}}^f,\ \hat{\mathbf{n}}^f,\ S^f\right\}$ refers to the barycenter, unit normal vector, and area of each non–Cartesian face comprising the CC, as well as the CC volume Ω^c and barycenter $\overline{\mathbf{x}}^c$. The geometric quantities of Cartesian faces (that can also exist in CCs), i.e. faces that coincide with those of the original Cartesian cells, can be straightforwardly computed using Eqs. (2.4) and (2.5), exploiting the cuboid topology. Thus, the following apply to faces generated due to intersections with the considered body surface.

Considering that the CC faces can have both convex or concave polygons with an arbitrary number of vertices $\mathbf{v}_n, n = 1, .., N$ their geometric quantities are computed by triangulating the face into N triangles that share a common (Steiner) point $\mathbf{s}$ at the arithmetic mean of the coordinates of the face vertices. Then, the normal vector of each triangle $\mathbf{n}^t$, with a magnitude

equal to its area, is computed using the cross product of the two vectors originating from the said Steiner point. The normal vector of the considered face is the sum of the constituent triangles normal vectors, i.e.

$$\mathbf{n}^f = \sum_1^N \mathbf{n}^t \qquad \mathbf{n}^t = \frac{1}{2}\mathbf{t}^1 \times \mathbf{t}^2 \tag{2.6}$$

The area and the unit normal vector of the face are then computed as $S = |\mathbf{n}|$ and $\hat{\mathbf{n}} = \frac{\mathbf{n}}{S}$. The face barycenter follows the same approach, i.e. the barycenters of comprising triangles $\overline{\mathbf{x}}^t$ are computed separately and are, then, area–averaged to give the face barycenter. This is expressed as

$$\overline{\mathbf{x}}^f = \frac{1}{S^f}\sum_1^N \overline{\mathbf{x}}^t S^t \qquad \overline{\mathbf{x}}^t = \frac{1}{3}\left(\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{s}\right) \tag{2.7}$$

Following an identical approach, the CC volume and barycenter are computed by decomposing the generated CC with P faces into P pyramids. The pyramids are created by connecting each CC face with a common Steiner point $\mathbf{s}^c$ at the arithmetic mean of the CC vertices. Thus, the CC volume is the sum of all comprising pyramid volumes Ω_p, while its barycenter is computed by volume–averaging the comprising pyramid barycenters $\overline{\mathbf{x}}^p$. Their expressions are given by

$$\Omega^c = \sum_1^P \Omega_p \qquad \Omega_p = \frac{1}{3}S^f\left(\overline{\mathbf{x}}^f - \mathbf{s}^c\right)\cdot\hat{\mathbf{n}}^f \tag{2.8}$$

$$\overline{\mathbf{x}}^c = \sum_1^P \Omega_p\overline{\mathbf{x}}^p \qquad \overline{\mathbf{x}}^p = \frac{3}{4}\left(\overline{\mathbf{x}}^{\mathbf{f}} - \mathbf{s}^{\mathbf{c}}\right) + \mathbf{s}^c \tag{2.9}$$

2.2.3 Small Cell Treatment

The cell–cutting procedure overviewed in Section 2.2.1 may result in arbitrary small CCs neighboring larger–sized cut or Cartesian cells. In the CC–related literature, this is commonly referred to as the small cell problem, as the appearance of such cases is inevitable and necessitates exceedingly small timesteps for stable explicit time–integration schemes, or it results in a stiff discretized system that can cause convergence issues in implicit schemes [124]. In both cases, the presence of small cells leads to increased simulation times, and thus several techniques have been developed to treat them. Such techniques include cell–merging [124, 279], in which the small cells are merged geometrically to create *hypercells*, cell linking/mixing [58, 92, 168], wherein small cells are connected with adjacent cells numerically, and special CC–specific dis-

cretizations [95, 156]. Their differences are their implementation complexity and obtained order of accuracy [156]. In this thesis, the cell–merging approach is followed.

The cell–merging algorithm identifies CCs that are considered *small*, i.e. exhibit a relative to a neighboring cell volume ratio of less than a threshold value; herein the value of 5% is used. It then requires them to merge with a neighboring cell to create a hypercell. To this end, each small cell locates a suitable neighbor, prioritizing, if possible, those which have not already been selected by other small cells to avoid the creation of elongated hypercells. All connectivity information and geometrical data are unified and passed on to the hypercell that now acts as a single CC. This process is illustrated in Figure 2.7.

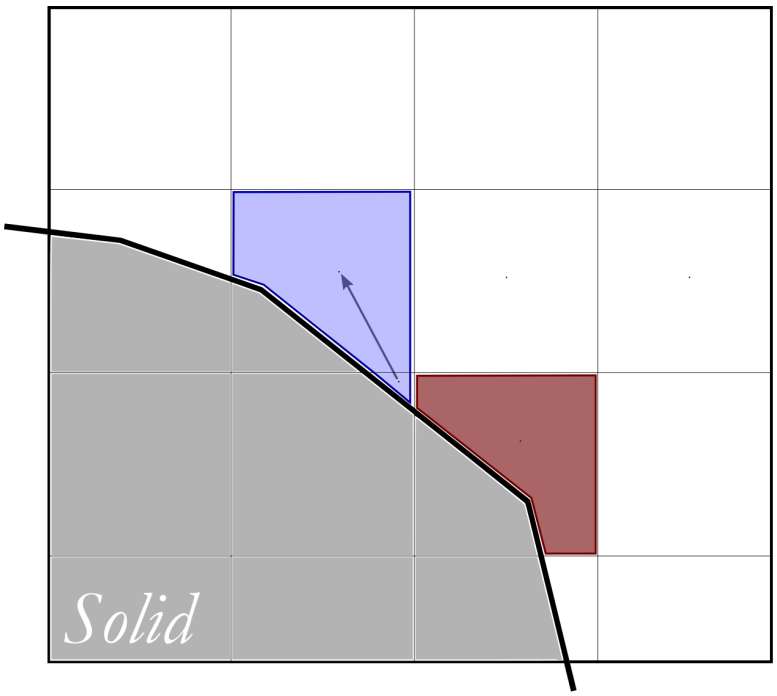


Figure 2.7: $2D$ schematic of the cell–merging algorithm for the treatment of small cells - Part of a CC mesh close to the body surface. The small cell is merged with its neighbor to create a cut–hypercell (ciel). After the merge, constituent cells are discarded and the merged cell is used instead. The created merged cell is treated similarly to a common CC (brown).

2.3 The Cut–Cell Data Structure

Even though the thus–far generated data structure can be used to create the necessary connectivity for flow simulations, it is associated with increased computational overhead due to tree traversal since it contains all descendants originating from the root cell [5, 47]. Instead, the tree

data structure is used to create a new face–based data structure, typically used in unstructured meshes, which results in a compact structure. To this end, Cartesian cells lying entirely inside the fluid domain are identified and have hanging nodes and faces created between Cartesian cells of different refinement levels. Then, fluid cells have their faces globally numbered, and each face is associated with the two corresponding cells it resides on, excluding those lying over the domain boundary and the body surface.

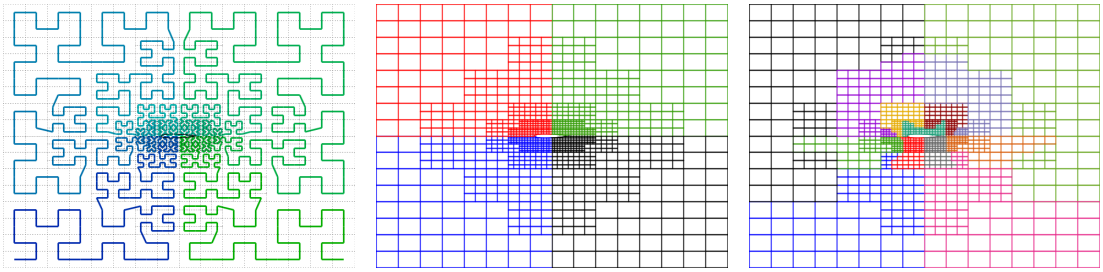
2.3.1 Cut–Cell Mesh Decomposition for Parallel Computing

The recursive nature of the Octree data structure and its growth–dependency on the inputted body surface make the parallelization of the CC mesh generation process significantly complex. Parallelization on distributed memory architectures, namely each processor generating their CC mesh in situ, would require a substantial amount of time investment to accomplish [96]. The main reasons are the complexity of impromptu balancing the generated tree and enforcing the locality of the generated subdomains (refined areas could be scattered among the processors). On the other hand, the associated data structures can be generated following a *master–slave* model, decompose the generated CC mesh and partition the resulting subdomains into the slave processors (including the appointed master) for the subsequent flow simulations. Even though this model is sub–optimal, the mesh generation process takes a small fraction of the total simulation time and proves to be very efficient ($O(10^6)$ cells/min), [214]. Throughout the extensions made in this thesis, the parallel behavior of the slave processors is retained, while the master processor is enriched following a shared memory paradigm. Extensions, executed by the master processor, are parallelized locally using OpenMP directives to utilize its threads. OpenMP directives avoid the need to exchange data among processors, allow significant speed–up of selected computationally intensive tasks but are limited to hardware specifications; hardware made available for use in this PhD thesis by the PCOpt allows the use of up to 24 threads. The underlying extensions that are parallelized using OpenMP directives are mentioned along with their implementations (see Sections 3.2.2.2, 6.3.6 and Appendix D).

Thus, in the existing software, the (master) generated CC mesh is decomposed into overlapping subdomains to exploit the benefits of parallel programming on distributed memory architectures. The MPI protocol is used to communicate between these subdomains when performing flow simulations. Domain decomposition is achieved via the Hilbert Space–Filling Curve [97], wherein the 3(2)–dimensional space is mapped onto a $1D$ space. Then, dividing the $1D$ space into equally weighted sub–spaces of any number is straightforward, favoring load

balancing and locality [9]. After decomposing the CC mesh into subdomains, these are associated with different processors, along with the necessary data structure, to initiate the numerical solution process of the governing equations using the FVM, described in Chapter 3.

Mesh decomposition using the Hilbert curve is exemplified in a dummy CC mesh over the isolated NACA 0012 airfoil. Figure 2.8a shows the resulting Hilbert curve on the non–uniform mesh, while in Figures 2.8b and 2.8c the same mesh is decomposed into 4 and 13 subdomains, respectively.



(a) (b) (c)

Figure 2.8: Mesh Decomposition using the Hilbert curve - (a) Hilbert curve on the generated CC mesh, colored by its length (blue → green). The curve starts at the south–west corner, passes through each Cartesian cell, and ends at the south–east corner. Mesh decomposition into (b) 4 and (c) 13 subdomains, with different colors for each subdomain.

Chapter 3

Flow Models for Single– and Two–Phase Incompressible Flows

THE main purpose of this chapter is to describe the flow model(s) used to perform aero/hydro-dynamic simulations using the CC mesh generation method for single– and cavitating two–phase flows. The presented flow models and numerical solution process subsequently serve as the building blocks to formulate their continuous adjoint counterpart (Chapter 4), to be integrated into a numerical optimization workflow based on the CC method.

Initially, the governing equations for single–phase laminar flows of incompressible fluids are presented, accompanied by details regarding the discretization scheme used, the enforcement of boundary conditions and the numerical solution process. These also describe the precursor CC solver that was developed in a recently accomplished PhD thesis [214]. Then, the extension to turbulent flows using the standard $k - \varepsilon$ model is presented, along with the CC–specific implementation of the wall functions technique that utilizes a pure Cartesian mesh [31, 249]. This is followed by the equations governing laminar two-phase flows that exhibit cavitation with the associated theoretical background. Finally, the extension to turbulent two–phase flows is described.

3.1 The Navier–Stokes Equations For Single–Phase Incompressible Flows

This section presents the steady–state governing equations of motion for incompressible flows, i.e. with constant density ϱ, that express the conservation of mass and momentum in each

direction of the Cartesian space $x_j,\ j = 1, 2\,(, 3)$, for a fluid with dynamic viscosity μ as

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{3.1}$$

$$\frac{\partial\,(u_k u_j)}{\partial x_j} + \frac{\partial \breve{p}}{\partial x_k} = \frac{\partial}{\partial x_j}\left(\frac{\mu}{\varrho}\left[\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k}\right]\right), \qquad k = 1, 2\,(, 3) \tag{3.2}$$

where $\breve{p} = \frac{p}{\varrho}$ is the kinematic pressure, p is the static pressure, and $\boldsymbol{u}$ the fluid velocity vector.

3.1.1 Artificial Compressibility Method

In incompressible flows, the mass conservation equation is reduced to a divergence–free flow velocity constraint due to the constant density. Furthermore, an equation of state no longer holds, i.e. pressure changes do not affect the density and, thus, directly applying time–marching techniques to solve the incompressible Navier–Stokes equations numerically is not possible. To circumvent this difficulty, Chorin [54] defined an artificial equation of state, $p = \varrho\beta^2$, coupling pressure and density using an artificial compressibility parameter β^2. The definition of an artificial equation of state and the corresponding speed of sound renders the system of PDEs hyperbolic in time and space that allows for the application of a time–marching solution method such as the one used in compressible flows [253]. Thus, the modified system of equations is marched in the pseudo(artificial)–time τ to asymptotically reach the steady–state solution. Once the said solution is reached, this artificial term vanishes.

According to the artificial compressibility method, the modified mass conservation equation becomes

$$\frac{1}{\beta^2}\frac{\partial \breve{p}}{\partial \tau} + \frac{\partial u_j}{\partial x_j} = 0 \tag{3.3}$$

In $3D$, the modified system of equations Eqs. (3.2) and (3.3) can be written in vector form as

$$\mathscr{R}_i := \Gamma_{in}\frac{\partial U_n}{\partial \tau} + \frac{\partial f^I_{ij}}{\partial x_j} - \frac{\partial f^V_{ij}}{\partial x_j} = 0 \tag{3.4}$$

where $\boldsymbol{\mathscr{R}}$ is the residual vector and

$$\boldsymbol{\Gamma} = \begin{bmatrix} \frac{1}{\beta^2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{f}^I_j = \begin{bmatrix} u_j \\ u_1\ u_j + \delta^1_j\ \breve{p} \\ u_2\ u_j + \delta^2_j\ \breve{p} \\ u_3\ u_j + \delta^3_j\ \breve{p} \end{bmatrix}, \qquad \boldsymbol{f}^V_j = \begin{bmatrix} 0 \\ \breve{\tau}_{1j} \\ \breve{\tau}_{2j} \\ \breve{\tau}_{3j} \end{bmatrix} \tag{3.5}$$

In Eq. (3.5), $\boldsymbol{\Gamma}$ is the artificial compressibility preconditioner matrix [54], $\boldsymbol{U} = \begin{bmatrix}\breve{p}\ u_1\ u_2\ u_3\end{bmatrix}^T$ the

vector of unknown flow variables, and $\boldsymbol{f}_j^I$, $\boldsymbol{f}_j^V$ the inviscid and viscous flux vectors, respectively. Furthermore, δ_j^k denotes the Kronecker delta and $\breve{\tau}_{kj} = \frac{1}{\varrho}\tau_{kj} = \frac{\mu}{\varrho}\left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k}\right)$ the viscous stress tensor. In the literature, several recommended values for the artificial compressibility parameter β can be found, based on either the local velocity [152, 257], or the flow problem reference velocity U_∞ [138, 251]. Herein, the latter is opted and, thus, $\beta = \sqrt{10}U_\infty$ is usually used.

3.1.2 The Navier–Stokes Equations

The governing equations (Eq. (3.4)) are discretized using a cell–centered FVM to be solved on the generated CC meshes. The finite volumes coincide with the cut and Cartesian cells and have the vector of unknown variables $\boldsymbol{U}$ stored at their barycenter, Figure 3.2. In $2D$, finite volumes can be rectangles or polygons with an arbitrary number of faces, while in $3D$, these become cuboids or polyhedrons.

In the FVM, the governing equations are integrated over each finite volume Ω^P. The flux integrals are transformed to surface integrals, computed over the cell faces, using the Gauss divergence theorem. Therefore, Eq. (3.4) becomes

$$\int_{\Omega^P} \Gamma_{in}\frac{\partial U_n}{\partial \tau} d\Omega + \int_{S(\Omega^P)} \left(f_{ij}^I - f_{ij}^V\right)\hat{n}_j dS = 0 \tag{3.6}$$

where $\hat{\mathbf{n}}$ is the outwards–facing unit normal vector along the surface dS.

Different discretization approaches are followed for the inviscid and viscous terms, based on stability criteria [253]. Ideally, both terms would be discretized using the central difference scheme, but it is well–known that the inviscid terms prove to be unconditionally unstable and, thus, require an upwind scheme.

3.1.2.1 Discretization of the Inviscid Flux Vector

The inviscid flux vector is discretized using a second–order accurate Total Variation Diminishing (TVD) Monotonic Upstream–centered Scheme for Conservation Laws (MUSCL) scheme [260], in which the flux vectors at the finite volume boundaries are computed using slope–limited reconstructed states $\boldsymbol{U}^{L/R}$ [248]. The discretized inviscid flux vector for Ω^P reads

$$\int\limits_{S(\Omega^P)} f_{ij}^I \hat{n}_j dS \approx \sum_{Q}^{Nei(P)} \tilde{f}_i^{PQ} + \sum_{B}^{\mathscr{B}(P)} \tilde{f}_i^B \tag{3.7}$$

where $\mathscr{B}(P)$ are the faces of Ω^P lying on the domain boundaries and $\tilde{\boldsymbol{f}}$ numerical flux vectors.

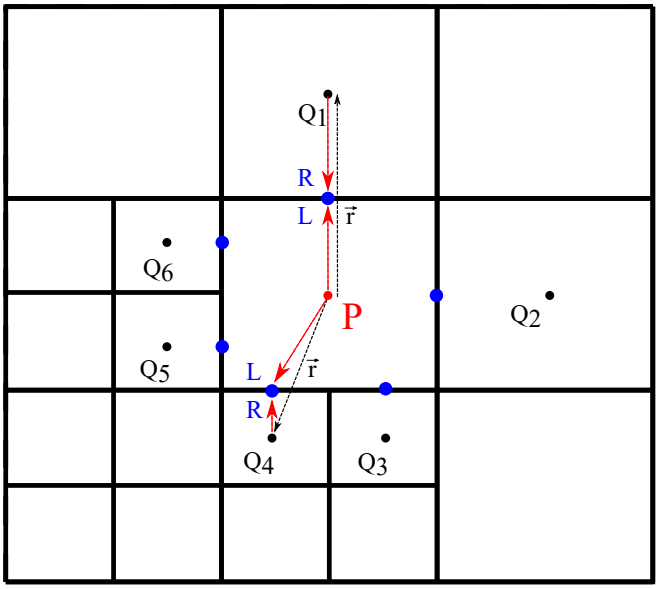


Figure 3.1: Example of the neighborhood of a Cartesian cell in $2D$. Numerical flux vectors $\tilde{\boldsymbol{f}}$ are computed between each neighbor via the Roe solver using reconstructed states, extrapolated from the neighboring finite volumes. Extrapolation may use either the same (P, Q_1), or different (P, Q_4) level neighbors.

Let P and Q denote two neighboring finite volumes (Figure 3.1). The (intercell) numerical flux vector $\tilde{\boldsymbol{f}}$ is computed via the Roe solver [209] by exactly solving an approximate Riemann problem

$$\tilde{f}_i \coloneqq \frac{1}{2}\left(f_{ij}^{I\,L} + f_{ij}^{I\,R}\right) n_j - \frac{1}{2}\tilde{D}_{in}\left(U_n^R - U_n^L\right) \tag{3.8}$$

where $n_j = \hat{n}_j \Delta S^{PQ,B}$ is the normal vector with magnitude equal to the area of their interface. The dissipation matrix $\tilde{\boldsymbol{D}}$ is a matrix that shares the same eigenvectors with the Jacobian matrix $\boldsymbol{A}_j = \frac{\partial \boldsymbol{f}_j^I}{\partial \boldsymbol{U}}$ (given in Appendix A.1), and the absolute values of its eigenvalues. This is computed using Roe–averaged quantities $\tilde{\boldsymbol{U}}$ as

$$\tilde{D}_{in} = \tilde{\Gamma}_{im} |\tilde{A}^{\Gamma}_{mnj} n_j| \tag{3.9}$$

$$|\tilde{A}^{\Gamma}_{mnj} n_j| = \tilde{\Gamma}^{-1}_{ml} \tilde{M}_{lq} \left|\tilde{\Lambda}_{qr}\right| \tilde{M}^{-1}_{rn} \tag{3.10}$$

with $\tilde{\boldsymbol{\Lambda}} = \mathrm{diag}\left(\tilde{u}_n,\ \tilde{u}_n,\ \tilde{u}_n + \tilde{c},\ \tilde{u}_n - \tilde{c}\right)$ being a diagonal matrix containing the eigenvalues of $\boldsymbol{A}$ and $c = \sqrt{u_n^2 + \beta^2 n_j n_j}$ the artificial speed of sound. Matrices $\tilde{\boldsymbol{M}}$, $\tilde{\boldsymbol{M}}^{-1}$ hold the linearly

independent right and left eigenvectors, as columns and rows respectively, and are provided in Appendix A.1.

The inviscid flux vectors at the finite volume faces in Eq. (3.8) are computed using reconstructed states stemming from the adjacent cell flow variables (see Figure 3.1), i.e.

$$f_{ij}^{I\,L/R} = f_{ij}^{I}\left(U_i^{L/R}\right) \tag{3.11}$$

$$U_i^{L/R} = U_i^{P/Q} + \phi \frac{\partial U_i}{\partial x_k}^{P/Q} \Delta x_k \tag{3.12}$$

where ϕ is the limiter value and $\frac{\partial U_i}{\partial x_k}^{P/Q}$ is the flow variable spatial derivatives tensor computed at each cell center using a linear Weighted Least Squares (WLSQ) method.

The computation of the numerical flux $\tilde{\boldsymbol{f}}$ differs on boundary faces that lie on solid walls S_W, than the remaining type of boundary faces, i.e. those lying at the inlets S_I, outlets S_O, or farfield boundaries S_∞. For faces that lie on S_W, $\tilde{\boldsymbol{f}}$ is computed by extrapolating the flow variables from the CC cell center and applying solid wall boundary conditions, i.e

$$\tilde{f}_i^{S_W} = f_{ij}^{I}\left(U_i^{S_W}\right) n_j \tag{3.13}$$

where $\boldsymbol{U}^{S_W}$ denotes the extrapolated flow variable values after the boundary condition is applied. For the remaining type of boundary faces, Eq. (3.8) is used to compute $\tilde{\boldsymbol{f}}$, where $\boldsymbol{U}^R$ is obtained using the specified boundary conditions. Details about the computation of $\boldsymbol{U}^{S_W}$ and $\boldsymbol{U}^R$ at the boundaries are discussed in Section 3.1.3.

3.1.2.2 Implementation of Limiters

In flows that exhibit strong flow variable spatial gradients, limiters are used to suppress spurious oscillations and maintain monotonicity by locally introducing additional numerical dissipation to the scheme. They are applied when reconstructing the L/R states of the finite volumes to ensure its boundness by truncating part of the computed flow variable spatial derivatives, Eq. (3.12). When non–uniform meshes are used, they can also have an impact on the solution accuracy even in smooth solutions [3, 33]. The CC solver employs two different limiting functions, namely the Barth–Jespersen and the Venkatakrishnan limiters [26, 263].

In the Barth–Jespersen scheme for unstructured meshes [26], the limiter value, used to compute the reconstructed states, corresponds to the minimum value computed at all faces, i.e.

$\phi = \min(\phi_f)$. Face limiter values are computed as

$$\phi_f = \begin{cases} \mathscr{F}\left(\frac{U_i^{max}-U_i}{U_i^*-U_i}\right) & U_i^* > U_i \\ \mathscr{F}\left(\frac{U_i-U_i^{min}}{U_i^*-U_i}\right) & U_i^* < U_i \\ 1 & U_i^* = U_i \end{cases} \tag{3.14}$$

with U^* being the non–limited reconstructed state at each face and

$$\mathscr{F} = \min(x, 1) \tag{3.15}$$

The Venkatakrishnan limiter [263] is a modified version of the Barth–Jespersen one [26], in which $\mathscr{F}$ is replaced by the differentiable limiting function

$$\mathscr{F} = \frac{x^2 + 2x + \varepsilon^2}{x^2 + x + 2 + \varepsilon^2} \tag{3.16}$$

to avoid convergence stall often seen when the Barth–Jespersen limiter is used [3, 33]. $\varepsilon = (Kh)^3$ is a constant value based on the finite volume edge size $h = \sqrt{\Delta x_j \Delta x_j}$, Eq. (2.4), and the parameter K (usually $K = 0.3$, [263]) determines the activation/deactivation threshold.

3.1.2.3 Flow Variable Spatial Derivatives Computation

The computation of the reconstructed states requires $\frac{\partial \boldsymbol{U}}{\partial \mathbf{x}}$ at the cell center of each finite volume, Eq. (3.12). These are computed via the WLSQ method, favoring its higher order of accuracy at the expense of increased computational cost, as compared to the Green–Gauss formulation [155]. The spatial derivatives of an arbitrary flow quantity ϕ are computed by solving a linear system of equations arising from the sum of squares of the differences between the $l = 1, L$ direct neighbors values and the ones extrapolated using a Taylor expansion from the finite volume cell center (Figure 3.1). By defining the extrapolation error E

$$E^2 \coloneqq \sum_l^L W_{lm}^2 \left(\Delta\phi^m - \frac{\partial \phi}{\partial x_j}\Delta x_j^m\right)^2 = \left(W_{lm}\Delta\phi^m - H_{lj}\frac{\partial \phi}{\partial x_j}\right)\left(W_{ln}\Delta\phi^n - H_{lj}\frac{\partial \phi}{\partial x_j}\right) \tag{3.17}$$

with

$$H_{lj} = W_{lm}\Delta x_j^m \tag{3.18}$$

where $\boldsymbol{W} = \mathrm{diag}(\boldsymbol{w})$, $w_l = \frac{1}{\mathrm{d}(l,\ P)}$ are the weight factor and $\mathrm{d}(l,\ P) = \sqrt{\left(x_j^l - x_j^P\right)\left(x_j^l - x_j^P\right)}$ the distance between the cell center of P and its l^{th} neighbor, the system of equations is

formulated by minimizing the error function w.r.t. the computed spatial derivatives values, i.e.

$$\frac{\partial E^2}{\partial\left(\frac{\partial\phi}{\partial x_k}\right)} = 2\left(W_{lm}\Delta\phi^m - H_{lj}\frac{\partial\phi}{\partial x_j}\right)\frac{\partial}{\partial\left(\frac{\partial\phi}{\partial x_k}\right)}\left(-H_{ln}\frac{\partial\phi}{\partial x_n}\right) \tag{3.19}$$

$$\Leftrightarrow \left(W_{lm}\Delta\phi^m - H_{lj}\frac{\partial\phi}{\partial x_j}\right)H_{ln}\delta_n^k = 0 \tag{3.20}$$

which leads to

$$H_{kl}^T H_{lj}\frac{\partial\phi}{\partial x_j} = H_{kl}^T W_{lm}\Delta\phi^m \tag{3.21}$$

to be solved using the Gauss–Jordan method.

A more convenient way of computing the spatial derivatives in Eq. (3.21) is by separating the geometric and flow quantities to pre–compute the geometric quantities and avoid their computation during the flow solution process. Rearranging Eq. (3.21) in the form of $\mathbf{Ax} = \mathbf{b}$ where all geometric quantities (and weights) are included in the $L \times 3$ matrix $\mathbf{A}$, $\mathbf{x} = \frac{\partial\phi}{\partial x_k}$ and $\mathbf{b} = \Delta\phi^m$, a solution is obtained by computing the pseudo–inverse matrix $\mathbf{A}^\dagger$, of size $3 \times L$, that also contains only geometric quantities, i.e.

$$\mathbf{x} = \mathbf{A}^\dagger\mathbf{b} \tag{3.22}$$

The pseudo–inverse matrix is computed by solving the following linear system

$$H_{km}^T H_{mj} A_{jl}^\dagger = W_{ml}H_{mk} \tag{3.23}$$

and stored to subsequently compute the spatial derivatives in each (pseudo) time step by updating flow variables difference vector $\mathbf{b}$ using Eq. (3.22).

Cells located at the computational domain boundaries have a reduced number of direct neighbors. For them, the WLSQ stencil is enriched with halo nodes, positioned outside the computational domain, to avoid an ill–conditioned WLSQ system (Eq. (3.23)) and maintain a second–order accurate gradient reconstruction [25, 91, 155].

Halo node coordinates x_j^D are computed using an averaged solid face comprising all the CC solid faces, as shown in Figure 3.2a, or using the boundary face normal vector via

$$x_j^D = x_j + 2\hat{n}_j r_k \hat{n}_k \tag{3.24}$$

where x_j are the coordinates of the boundary cell barycenter, r_k is the vector connecting the

face and cell barycenters, and $\hat{n}_k$ is the normal unit vector of either the boundary face or the solid faces averaged one. The flow variable differences are then computed based on the imposed boundary conditions. For Dirichlet conditionsn the halo nodal values are $\phi^D = 2\phi^W - \phi^B$ with ϕ^W being the specified flow variable value at the boundary and ϕ^B the boundary cell value, while for zero Neumann conditions, $\phi^D = \phi^B$.

3.1.2.4 Discretization of the Viscous Flux Vector

Discretizing the viscous flux vector in Eq. (3.6) one obtains

$$\int\limits_{S(\Omega^P)} f_{ij}^V \hat{n}_j dS \approx \sum_{Q}^{Nei(P)} \left(\overline{f}_i^V\right)^{PQ} + \sum_{B}^{\mathscr{B}(P)} \left(\overline{f}_i^V\right)^{B} \tag{3.25}$$

where $\overline{f}_i^{\,V} = f_{ij}^V n_j$ and, thus, requires the computation of the shear stress tensor on the finite volumes' faces, Eq. (3.5). The spatial derivatives of an arbitrary variable ϕ between two neighboring finite volumes P and Q is distance–weighted from the corresponding face barycenter LR as

$$\overline{\frac{\partial \phi}{\partial x_k}} = w \frac{\partial \phi}{\partial x_k}^{P} + (1-w) \frac{\partial \phi}{\partial x_k}^{Q} \tag{3.26}$$

with

$$w = \frac{\mathrm{d}\left(Q,\ LR\right)}{\mathrm{d}\left(Q,\ LR\right) + \mathrm{d}\left(P,\ LR\right)} \tag{3.27}$$

Term $\frac{\partial \phi}{\partial x_k}^{P/Q}$ refers to the spatial derivatives computed at each cell center using the WLSQ method as described in Section 3.1.2.3. The viscous surface fluxes are computed by performing a spatial derivative correction in the direction of the connecting cell centers, i.e.

$$\frac{\partial \phi}{\partial x_k}^{LR} = \overline{\frac{\partial \phi}{\partial x_k}} - \left[\overline{\frac{\partial \phi}{\partial x_j}} \hat{r}_j - \frac{\phi^Q - \phi^P}{d\left(P,\ Q\right)}\right] \hat{r}_k \tag{3.28}$$

where $\boldsymbol{r}$ is the vector connecting PQ (Figure 3.1) and $\hat{\boldsymbol{r}} = \frac{\boldsymbol{r}}{\mathrm{d}(P,Q)}$ the corresponding unit vector. In the uniform areas of the CC mesh, $\hat{\boldsymbol{r}}$ becomes a single element vector and, thus, Eq. (3.28) replaces the averaged WLSQ derivative with centered differences between the P, Q cells along that direction [31]. Overall, the resulting discretization is equivalent to a second–order central differencing scheme and incorporates all immediate neighbors [276].

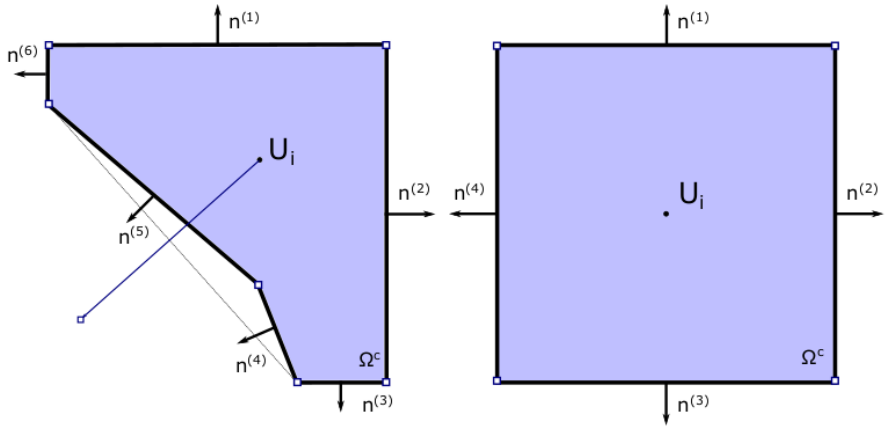


Figure 3.2: Illustration of $2D$ finite volumes in the CC method - (a) The presence of the body surface results in the creation finite volume with 6 boundary surfaces dS. The additional halo (dummy) nodes added outside the computation domain that partakes in the WLSQ stencil is also shown and (b) typical (non–cut) finite volume. The normal vectors $\mathbf{n}$ along each face are also illustrated.

3.1.3 Boundary Conditions

In the CC method, boundary conditions are imposed along faces aligned with the domain boundaries and, thus, a similar approach to conventional body–fitted solvers is followed. The boundary flow variables $\boldsymbol{U}^S$ are computed based on the imposed conditions. Along the solid walls, these are used to directly compute the surface fluxes while, at the remaining boundaries, they provide the reconstructed states $\boldsymbol{U}^R$ of the halo nodes.

3.1.3.1 Wall Boundary Conditions – S_W

In case of a slip wall, a no–penetration condition is applied, i.e. the normal velocity component of the boundary velocity vector is set to zero, i.e. $u_j^W = u_j - u_k\hat{n}_k\hat{n}_j$. For a no–slip wall, the velocity vector at the wall is zeroed, $u_j^W = 0$ and $\breve{p}$ is extrapolated from the boundary cell.

3.1.3.2 Inlet Boundary Conditions – S_I

Inlet oundary conditions are imposed by either specifying the velocity magnitude $\sqrt{u_j u_j}$ or the total pressure $p_t = p + \frac{1}{2}\varrho u_j u_j$. In either case, two angles that define the inlet flow velocity vector must be specified too. Then, inlet conditions are imposed by extrapolating the kinematic

pressure from the boundary cell, and computing the velocity vector at the boundary face as

$$u_1^S = \sqrt{u_j u_j}\sin\theta_{ZX}\cos\theta_{XY} \tag{3.29}$$

$$u_2^S = \sqrt{u_j u_j}\sin\theta_{ZX}\sin\theta_{XY} \tag{3.30}$$

$$u_3^S = \sqrt{u_j u_j}\cos\theta_{ZX} \tag{3.31}$$

with θ_{ZX} and θ_{XY} being the angle of the velocity vector from the Cartesian plane denoted by its subscript. If the total pressure is specified, the velocity magnitude is computed at each pseudo–time step using the kinematic pressure extrapolated from the interior of the domain.

3.1.3.3 Outlet Boundary Conditions – S_O

At the outlet boundaries, the static pressure is specified and is used to compute the kinematic pressure, while the remaining variables are extrapolated from the interior domain.

3.1.3.4 Farfield Boundary Conditions – S_∞

In external aerodynamics, the flow variables at the farfield boundary faces are computed from the imposed reference flow variables $\boldsymbol{U}_\infty$.

3.1.4 Numerical Solution of the Navier–Stokes Equations

The governing equations Eq. (3.6), expressed for finite volume P, can be written in semi–discrete form at each pseudo–time step as

$$\Omega^P \Gamma_{in}^P \left(\frac{\partial U_n}{\partial \tau}\right)^P = -\mathcal{R}_i^P \tag{3.32}$$

with

$$\mathcal{R}_i^P = \sum_Q^{Nei(P)} \left(\tilde{f}_i - \overline{f}_i^V\right)^{PQ} + \sum_B^{\mathscr{B}(P)} \left(\tilde{f}_i - \overline{f}_i^V\right)^B \tag{3.33}$$

Applying Taylor expansion on the right–hand–side of Eq. (3.32) and discretizing the pseudo–temporal term, the discretized system of equations, in Δ–form, becomes

$$\left[\Gamma_{in}^P \left(\frac{\Omega^P}{\Delta\tau^P}\right) + \frac{\partial \mathcal{R}_i^P}{\partial U_n}\right] \Delta U_n^P = -\mathcal{R}_i^P \tag{3.34}$$

where $\Delta\tau$ is the local pseudo–time step and $\Delta {U_n}^{\tau+1} = U_n^{\tau+1} - U_n^\tau$ the flow variables correction. The local pseudo–time step is defined based on a combination of the inviscid, the spectral radii

of $\boldsymbol{\Gamma}^{-1}\boldsymbol{A}$, and viscous terms, i.e.

$$\Delta\tau^P = \mathrm{CFL}\min\left(\Delta\tau_j^I + \Delta\tau_j^V\right) = \mathrm{CFL}\min\left(\frac{\Delta x_j^P}{|u_j^P| + c^P} + \frac{{\Delta x_j^2}^P}{2\frac{\mu}{\varrho}}\right) \tag{3.35}$$

where CFL is the Courant number [59]; typically, CFL = 5 is used.

Eq. (3.34) is iteratively solved by means of a point implicit $n \times n$ block symmetric Gauss–Seidel method by decomposing the left–hand–side into diagonal and off–diagonal matrix terms and rearranging, i.e.

$$\mathcal{D}_{in}^P\ \Delta U_n^{P,\tau+1} = -\left[\mathcal{R}_i^{P,\tau} + \sum_{Q\in\mathbb{L}}^{Nei(P)} \mathcal{Z}_{in}^{PQ}\Delta U_n^{Q,\tau+1} + \sum_{Q\in\mathbb{U}}^{Nei(P)} \mathcal{Z}_{in}^{PQ}\Delta U_n^{Q,\tau}\right] \tag{3.36}$$

$$\mathcal{D}_{in} = \Gamma_{in}^P\frac{\Omega^P}{\Delta\tau^P} + \frac{\partial\mathcal{R}_i^P}{\partial U_n^P} \qquad\qquad \mathcal{Z}_{in} = \frac{\partial\mathcal{R}_i^P}{\partial U_n^Q} \tag{3.37}$$

In Eq. (3.36), $\mathbb{L}$ refers to the set of neighbors whose flow variables correction has already been updated, while $\mathbb{U}$ the opposite. In each pseudo–time step, five sweeps of the symmetric Gauss–Seidel method are performed before updating the finite volume flow variables and recomputing $\boldsymbol{\mathcal{R}}$, $\boldsymbol{\mathcal{D}}$, and $\boldsymbol{\mathcal{Z}}$.

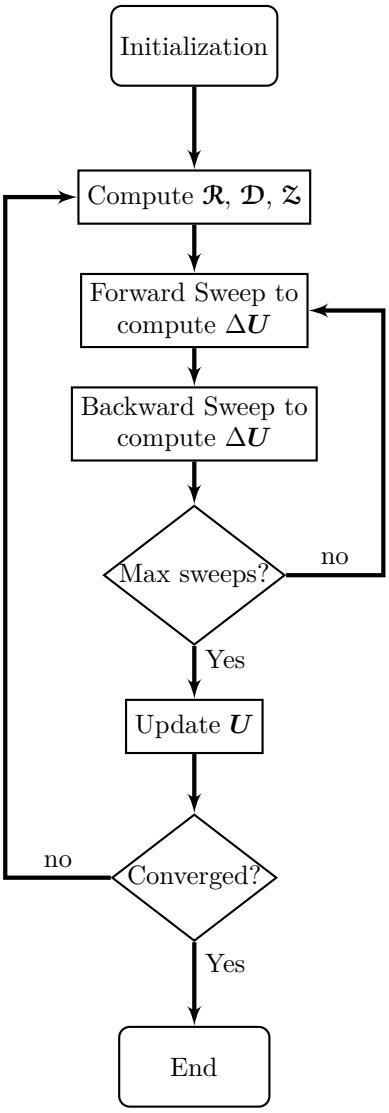


Figure 3.3: Flowchart of the numerical process - Solution of the Navier–Stokes equations using time–marching techniques. The symmetric Gauss–Seidel method consists of a forward and a backward sweep.

3.2 Simulating Turbulent Flows using the Cut–Cell Method

Up to this point, the precursor CC solver has been presented for which minor additions and modifications were attempted. In this section, the approach that facilitates single–phase turbulent flow simulations is described. The overviewed numerical solution process and discretization scheme are also utilized for the numerical solution of the turbulence model transport equations. Challenges arise due to the irregular finite volumes near the body surface and the inability of the CC method, and IBMs in general, to create high aspect finite volumes along the wall–normal direction [32, 46, 90, 250] effectively depriving wall–resolved simulations. In the following, the implications, and reasoning, that necessitate a wall model are discussed in more detail and a method to simulate flows governed by the RANS equations using the CC method is presented.

3.2.1 Preliminaries

The simulation of turbulent flows is closely associated with the generation of inflation layers near the solid walls, or in other words, the existence of very high aspect ratio finite volumes at these areas. These inflation layers rapidly increase the mesh resolution requirements to accurately resolve the thin boundary layer present in high Reynolds number flows. However, in the CC method, and the IBMs in general, the isotropically refined Cartesian meshes demand an unjustifiably large number of cells to accurately resolve the boundary layer and, thus, become impractical for wall–resolved turbulent simulations. The most obvious way to overcome this difficulty is to combine the background Cartesian meshes with inflation layers next to the wall, namely strand meshes [115, 274, 275]. However, this aberrates the cornerstone of the IBMs, i.e. simple and automatic mesh generation for complex geometries. For pure Cartesian and CC meshes, the use of a wall model can provide wall closure and alleviate the near-wall mesh resolution requirements. Thus, developments of such models have seen an increased research interest in both Cartesian–based and conventional body–fitted solvers [23, 52, 202, 210].

In light of the aforementioned, the extension of the CC method to turbulent flows is pursued using a *high–Reynolds* number turbulence model that inherently models the boundary layer next to the walls, namely the standard $k-\varepsilon$ model [143]. An additional reason for this choice lies in its extensive use when two–phase flows are considered [138, 222, 234, 280]. Near–wall modeling is achieved through analytic wall functions due to their maturity, wide application and well–understood limitations. Building on the last remark, one of their main limitations lies in their sensitivity to the first cell distance that calls for special treatment considering the irregular CC

finite volumes next to the walls [171].

3.2.2 The Standard k–ε Turbulence Model

The two-equation k-ε turbulence model was developed by Launder & Spalding [143] to provide turbulence closure to the RANS equations. Throughout the years, various additions [53, 193, 230, 278] have led to the refinement of the k-ε turbulence model and enabled its applicability in flows of increased complexity. Due to its maturity and relatively simple implementation, it is one of the most commonly used turbulence models, applicable to a variety of turbulent flows. In the following, the turbulence model equations are given in their compressible form to facilitate the extension towards two–phase turbulent flows that exhibit density variations. In addition, the implementation of wall functions specifically in the CC method is presented.

3.2.2.1 Transport Equations

The two-equation turbulence model solves the transport equations (PDEs) for the turbulent kinetic energy k and turbulent dissipation rate ε

$$\begin{aligned} \frac{\partial\left(\varrho k\ u_j\right)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] &= \mathscr{P} - \varrho\varepsilon \\ \frac{\partial\left(\varrho\varepsilon\ u_j\right)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] &= \left(C_{1\varepsilon}\mathscr{P} - C_{2\varepsilon}\varrho\varepsilon\right)\frac{\varepsilon}{k} \end{aligned} \qquad j = 1, 2\,(, 3) \tag{3.38}$$

in order to compute the eddy viscosity

$$\mu_t = C_\mu \varrho k \frac{k}{\varepsilon} \tag{3.39}$$

with model constants $C_\mu = 0.09$, $\sigma_k = 1.0$, $\sigma_\varepsilon = 1.3$, and $C_{1\varepsilon} = 1.44$, $C_{2\varepsilon} = 1.92$ [143]. The production of turbulent kinetic energy is defined as

$$\mathscr{P} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_j^i\right)\frac{\partial u_i}{\partial x_j} \tag{3.40}$$

3.2.2.2 Solid Wall Boundary Conditions – Wall Functions

Wall functions are based on analytic solutions that describe the flow of a zero pressure gradient flat plate to avoid resolving the boundary layer [237]. Contrary to the outer flow, where inertial forces dominate, the near-wall flow is governed by viscous stresses and, therefore, can be

described using the $1D$ boundary layer equation [126, 272] using only the diffusive terms, i.e.

$$\frac{\partial}{\partial y}\left((\mu+\mu_t)\,\frac{\partial u_t}{\partial y}\right)=0 \tag{3.41}$$

where here u_t denotes the streamwise velocity and y the wall–normal direction. Integrating Eq. (3.41) and scaling with the friction velocity u_τ and the kinematic viscosity $\nu=\frac{\mu}{\varrho}$ leads to a universal expression, valid for attached flows between the outer flow and the solid wall,

$$\left(1+\nu_t^+\right)\frac{d\mathrm{u}^+}{d\mathrm{y}^+}=1 \tag{3.42}$$

with $\mathrm{u}^+=\frac{u_t}{\mathrm{u}_\tau}$, $\mathrm{y}^+=\frac{y\mathrm{u}_\tau}{\nu}$, $\nu_t^+=\frac{\mu_t}{\varrho\nu}$.

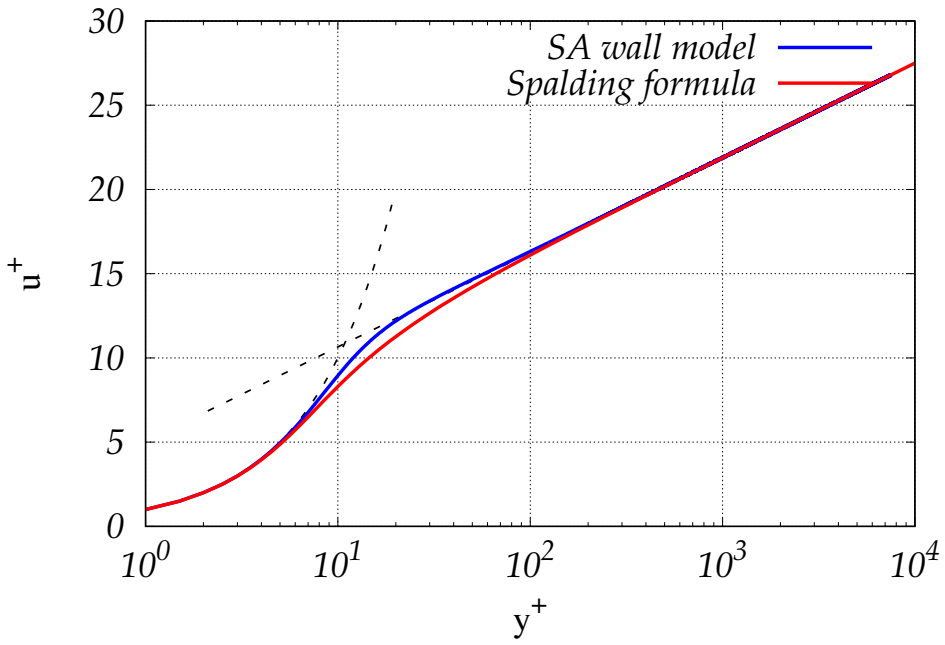


Figure 3.4: Universal wall function profile - Comparison of the SA wall model and the Spalding formula Eqs. (3.44) and (3.43). The linear and logarithmic are depicted with dashed lines in the region of their validity.

The viscous sublayer, $\nu_t^+ << 1$, is described by the linear correlation $\mathrm{u}^+=\mathrm{y}^+$, while the logarithmic layer, $\nu_t^+ >> 1$, is described by $\mathrm{u}^+=\frac{1}{\kappa}\log(\mathrm{y}^+)+B$ and $\nu_t^+=\kappa\mathrm{y}^+$, with κ being the von Kármán constant and $B\approx 5.033$. Universal formulas that correlate u^+ with y^+ in both

layers exist, such as the Spalding's formula [239], given by

$$\mathrm{y}^+ = \mathrm{u}^+ + e^{-\kappa B}\left(e^{\kappa \mathrm{u}^+} - \sum_{n=0}^{N=3} \frac{1}{n!}\left(\kappa \mathrm{u}^+\right)^n\right) \tag{3.43}$$

and the Spalart–Allmaras (SA) wall model

$$\begin{aligned}\mathrm{u}^+\left(\mathrm{y}^+\right) = B + c_1 \log\left(\left(\mathrm{y}^+ + a_1\right)^2 + b_1^2\right) - c_2 \log\left(\left(\mathrm{y}^+ + a_2\right)^2 + b_2^2\right) \\ - c_3 \arctan\left(\frac{b_1}{\mathrm{y}^+ + a_1}\right) - c_4 \arctan\left(\frac{b_2}{\mathrm{y}^+ + a_2}\right)\end{aligned} \tag{3.44}$$

with its many constants being, as presented in Allmaras & Johnson [8],

$$\begin{aligned} a_1 &= 8.148221580024245 & a_2 &= -6.9287093849022945 & c_3 &= 3.599459109332379 \\ b_1 &= 7.4600876082527945 & b_2 &= 7.468145790401841 & c_4 &= 3.6397531868684494 \\ c_1 &= 2.5496773539754747 & c_2 &= 1.3301651588535228 & & \end{aligned} \tag{3.45}$$

The two formulas are shown in Figure 3.4 along with the linear and logarithmic correlations showing their discrepancy at the buffer region.

However, the aforementioned cannot be directly implemented in the CC method since the first cell distances exhibit are highly irregular. Thus, a regularization of the considered wall–normal distances is first required to obtain an accurate description of the turbulent boundary layer. Following [32, 46, 210, 250], this is done using *linelets*, wherein *forcing points* $\mathbf{x}^F$ (Figure 3.5) are introduced normal to the solid walls inside the fluid domain, at a constant distance $d^F = \sqrt{d} \times \min(\Delta\mathbf{x})$, i.e.

$$\mathbf{x}^F = \mathbf{x}^f - d^F \hat{\mathbf{n}}^f \tag{3.46}$$

with $\mathbf{x}^f, \hat{\mathbf{n}}^f$ are the originating solid face barycenter and outwards facing unit normal vector, $d = 2$ in $2D$ and $d = 3$ in $3D$, and $\Delta\mathbf{x}$ is the Cartesian cell edge size, Eq. (2.4), to ensure the forcing point resides outside the CC. The linelets act as a two–layer subgrid governed by the $1D$ boundary layer equation Eq. (3.41) and are coupled with the outer RANS solution to obtain the friction velocity u_τ and, subsequently, the wall shear stress.

Tangential velocities u_t at the forcing points are reconstructed from the background RANS solution using the already computed velocity spatial derivatives of the finite volume that encompasses them. Then, they are used to solve either Eq. (3.43) or Eq. (3.44) to obtain u_τ. In this thesis, the SA wall model formula, viz. Eq. (3.44), is iteratively solved using a Newton

method, scaling $\left(u_t, d^F\right)$ to $(\mathrm{u}^+, \mathrm{y}^+)$. The iterative method used is described by

$$\begin{aligned} \mathrm{u}_\tau^{n+1} &= \mathrm{u}_\tau^n - \frac{f(\mathrm{u}_\tau)}{f'(\mathrm{u}_\tau)} \\ f(\mathrm{u}_\tau) &= B + c_1 \log\left(\left(\mathrm{y}^+ + a_1\right)^2 + b_1^2\right) - c_2 \log\left(\left(\mathrm{y}^+ + a_2\right)^2 + b_2^2\right) \\ &\quad - c_3 \arctan\left(\frac{b_1}{\mathrm{y}^+ + a_1}\right) - c_4 \arctan\left(\frac{b_2}{\mathrm{y}^+ + a_2}\right) - \mathrm{u}^+ = 0 \end{aligned} \tag{3.47}$$

$$f'(\mathrm{u}_\tau) = \frac{df}{d\mathrm{u}_\tau}(\mathrm{u}_\tau) = \frac{\mathrm{y}^+}{\mathrm{u}_\tau}\left[\frac{c_3 b_1 + 2c_1\left(\mathrm{y}^+ + a_1\right)}{\left(\mathrm{y}^+ + a_1\right)^2 + b_1^2} + \frac{c_4 b_2 - 2c_2\left(\mathrm{y}^+ + a_2\right)}{\left(\mathrm{y}^+ + a_2\right)^2 + b_2^2}\right] + \frac{\mathrm{u}^+}{\mathrm{u}_\tau}$$

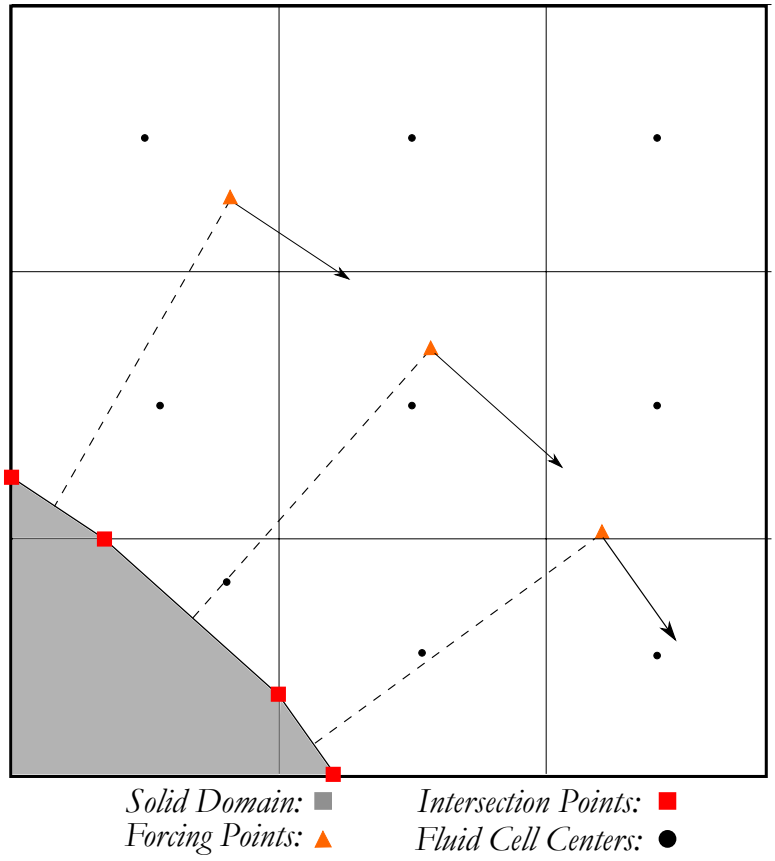


Figure 3.5: Generation of linelets - Illustration of the forcing points positioned at a constant distance d^F from the solid wall inside the fluid domain to create the linelets. The velocity vector is extrapolated from the cell centers of the finite volumes the forcing points reside in and projected along the tangential direction.

The viscous flux at the solid faces is computed through $\tau_w = \varrho \mathrm{u}_\tau^2$, and the appropriate boundary conditions for the turbulent variables are imposed on the CCs based on their barycen-

ter normal distance to the wall y.

If the said point resides in the log layer, $\mathrm{y}^+ > y^+_{cr}$, $\mathscr{P} \approx \varrho\varepsilon$ is assumed leading to

$$k = \frac{\mathrm{u}_\tau^2}{\sqrt{C_\mu}}, \qquad \varepsilon = \frac{\mathrm{u}_\tau^3}{\kappa y} \tag{3.48}$$

In contrast, if it resides in the viscous sublayer, $\mathrm{y}^+ \leq y^+_{cr}$,

$$k = \frac{\mathrm{u}_\tau^2}{\sqrt{C_\mu}} \left(\frac{\mathrm{y}^+}{y^+_{cr}} \right)^2, \qquad \varepsilon = k \frac{\sqrt{k} y + 5.3\nu}{\kappa C_\mu^{-\frac{3}{4}} y^2} \tag{3.49}$$

where $\mathrm{y}^+_{cr} = \frac{1}{\kappa} \log(\mathrm{y}^+_{cr}) + B$.

Finally, the velocity spatial derivatives of the CCs, previously computed by the WLSQ method, are replaced by the velocity spatial derivatives of the wall function profile using [249]

$$\frac{d\mathrm{u}^+}{d\mathrm{y}^+} = \frac{\nu}{\mathrm{u}_\tau^2} \frac{du_t}{dy} \tag{3.50}$$

The implemented method requires additional connectivity information; forcing points can reside in Cartesian cells that are not immediate neighbors of the CCs. In parallel computations, these might as well not reside in the overlapping domain. As such, an auxiliary data structure is created to provide the necessary data. These comprise the solid faces, the Cartesian cells that encompass the forcing points and the coordinates of the forcing points, used for subsequent velocity reconstruction. For this reason, the forcing points are created on the master processor during the CC mesh generation process using multiple threads via OpenMP directives, and the associated data structure is then computed and provided to the slave processors along with their mesh subdomain for the subsequent flow solutions. The additional inter–processor communications occur after each flow variable and spatial derivatives update to provide the new extrapolated velocity vector. It is more convenient to reconstruct the velocity vectors at the forcing point in the processor they reside in and provide it to the CC so that the tangential velocity can natively be computed using the already available unit normal vector of the CC. On a further note regarding the implementation of the forcing points , initially, a linelet was created for every solid face residing in the CC. The finite volume u_τ was computed by averaging the u_τ of each solid face. However, the accuracy gain was negligible since u_τ differences within the finite volume were minute. Currently, each CC creates an average solid face, similarly to Figure 3.2a, and the forcing point coordinates arise using Eq. (3.46) and the averaged solid face unit normal vector to compute a single u_τ. As such, the additional memory requirements

and computations that are associated with storing forcing points and solving Eq. (3.47) are significantly reduced.

3.2.2.3 Inlet/Farfield Boundary Conditions – S_I/S_∞

Explicitly computing/guessing realistic turbulent variables values at S_I/S_∞ can be challenging and requires delicate handling. A common alternative is to estimate some turbulence–related flow properties that are used to compute the required boundary turbulent variables values and are closer to human intuition. Herein, the turbulent intensity I is used to compute k, which is then used to compute ε based on a provided eddy viscosity ratio $\frac{\mu_t}{\mu}$, i.e.

$$k^S = \frac{3}{2} I^2 u_j^S u_j^S \tag{3.51}$$

$$\varepsilon^S = C_\mu \varrho \frac{\left(k^S\right)^2}{\mu} \left(\frac{\mu_t}{\mu}\right)^{-1} \tag{3.52}$$

where the typical turbulent intensity inside a pipe corresponds to $1\% < I < 5\%$, while for external flows it can be even lower. The eddy viscosity ratio directly relates to how strong the eddy viscosity is and it is usually set between $1 < \frac{\mu_t}{\mu} < 30$.

3.2.2.4 Outlet Boundary Conditions – S_O

At the outlet boundaries, zero Neumann boundary conditions are imposed for both turbulence variables and, thus, the field turbulence variables are extrapolated from the interior domain.

3.2.3 The RANS Equations

The RANS equations are solved in a coupled manner with the unknown flow variables being $\boldsymbol{U} = \left[p\ u_1\ u_2\ u_3\ \varrho k\ \varrho\varepsilon\right]^T$ for $3D$ problems [67]. Even though there are different opinions in the literature and some observations could be attributed to case dependency, the coupled formulation was opted partly due to its potential to facilitate better stability and convergence by being more mathematically consistent [141, 142, 147, 235]. An additional factor that also led to the coupled formulation was that it was more convenient to implement it in the precursor software; a similar to the Navier–Stokes solution procedure can be followed (Figure 3.3) by extending the preconditioner matrix and the inviscid and viscous flux vectors of Eq. (3.5) to incorporate the two additional transport equations. For the RANS equations, these become

$$\mathscr{R}_i := \Gamma_{in} \frac{\partial U_n}{\partial \tau} + \frac{\partial f_{ij}^I}{\partial x_j} - \frac{\partial f_{ij}^V}{\partial x_j} - S_i = 0 \tag{3.53}$$

with

$$\boldsymbol{\Gamma} = \begin{bmatrix} \left[\boldsymbol{\Gamma}\right]^{NS} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix}, \boldsymbol{f}_j^I = \begin{bmatrix} \boldsymbol{f}_j^{I,NS} \\ \varrho k u_j \\ \varrho \varepsilon u_j \end{bmatrix}, \boldsymbol{f}_j^V = \begin{bmatrix} \boldsymbol{f}_j^{V,NS} \\ \left(\mu + \frac{\mu_t}{\sigma_k}\right) \frac{\partial k}{\partial x_j} \\ \left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right) \frac{\partial \varepsilon}{\partial x_j} \end{bmatrix}, \boldsymbol{S} = \begin{bmatrix} \mathbf{0} \\ \mathscr{P} - \varrho\varepsilon \\ \left(C_{1\varepsilon}\mathscr{P} - C_{2\varepsilon}\varrho\varepsilon\right) \frac{\varepsilon}{k} \end{bmatrix} \tag{3.54}$$

where the superscript NS denotes the expression used in the Navier–Stokes equations. The shear stress is now computed as $\breve{\tau}_{kj} = \frac{(\mu+\mu_t)}{\varrho}\left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k}\right)$, following Boussinesq hypothesis, and $\boldsymbol{S}$ refers to the source term vector introduced due to the turbulence model.

The inviscid Jacobian matrix is formed as follows

$$\boldsymbol{A}_j = \begin{bmatrix} & & \left[\boldsymbol{A}_j\right]^{NS} & & \mathbf{0} & \mathbf{0} \\ 0 & \varrho k\ \delta_j^1 & \varrho k\ \delta_j^2 & \varrho k\ \delta_j^3 & u_j & 0 \\ 0 & \varrho \varepsilon\ \delta_j^1 & \varrho \varepsilon\ \delta_j^2 & \varrho \varepsilon\ \delta_j^3 & 0 & u_j \end{bmatrix} \tag{3.55}$$

where $\left[\boldsymbol{A_j}\right]^{NS}$ is given in Appendix A.1. This results in two additional eigenvalues, $\tilde{\boldsymbol{\Lambda}} = \mathrm{diag}\left(\tilde{u}_n,\ \tilde{u}_n,\ \tilde{u}_n + \tilde{c},\ \tilde{u}_n - \tilde{c},\ \tilde{u}_n,\ \tilde{u}_n\right)$, and an eigensystem of the form

$$\boldsymbol{M} = \begin{bmatrix} \left[\boldsymbol{M}\right]^{NS} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{M}^{-1} = \begin{bmatrix} \left[\boldsymbol{M}^{-1}\right]^{NS} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix} \tag{3.56}$$

The turbulence model inviscid and viscous fluxes are computed to second–order accuracy using the approach presented in Sections 3.1.2.1 and 3.1.2.1.

3.2.3.1 Discretization of the Turbulence Model Source Terms

The turbulence model introduces source terms to be discretized. Their values are considered to be constant within each finite volume, therefore

$$\int_{\Omega^P} S_i d\Omega \approx S_i \Omega^P \tag{3.57}$$

and are included in the discretized residual vector (Section 3.1.4)

$$\mathcal{R}_i^P = \mathcal{R}_i^{NS,P} - S_i^P \Omega^P \tag{3.58}$$

The source terms split into source *(Production)* and sink *(Destruction)* components, to incorporate them in the Δ-form and increase diagonal dominance [67, 261]. Specifically, the negative

component of the source terms is treated implicitly, i.e. added to the diagonal term in Eq. (3.37), while the positive explicitly. As such,

$$\boldsymbol{S} = \boldsymbol{S}^{+} + \boldsymbol{S}^{-} \tag{3.59}$$

$$\frac{\partial \boldsymbol{S}^{-}}{\partial \boldsymbol{U}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -2C_{\mu}\frac{\varrho k}{\mu_t} & 0 \\ \mathbf{0} & C_{2\varepsilon}\left(\frac{\varrho\varepsilon}{\varrho k}\right)^2 & -2C_{2\varepsilon}\frac{\varrho\varepsilon}{\varrho k} \end{bmatrix} \tag{3.60}$$

where the expression $\varrho\varepsilon = C_{\mu}\frac{(\varrho k)^2}{\mu_t}$ is used for the destruction term for the k-equation.

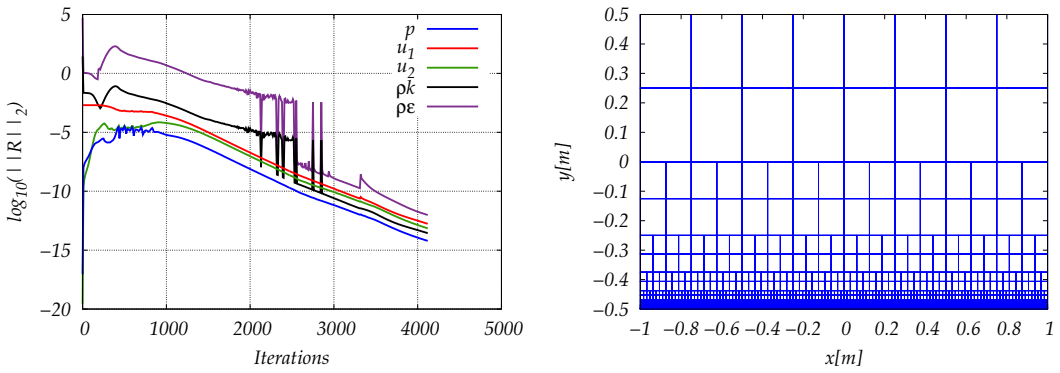
3.2.4 Cut–Cell based Single–phase Turbulent Flow Simulations

Single–phase turbulent flow simulations are limited to the following two $2D$ examples since additional validation/verification cases are also performed for two–phase turbulent flows in Section 3.5. The purpose of the simulated examples is to assess the implemented standard $k-\varepsilon$ turbulence model along with the described wall function technique in the CC solver. The first example considers a flat plate and is used to validate the CC turbulent solver, while in the second, a 90° curved channel is simulated to assess the forcing point technique in the presence of curved surfaces.

3.2.5 Flat Plate $Re_L = 5 \times 10^6$

The first case involves the turbulent flow over a flat plate, at $Re_L = 5 \times 10^6$, and comparisons with numerical [256] and experimental [277] data, to assess the accuracy of the implemented approach. For the simulation, the velocity magnitude ($5\frac{m}{s}$) and direction ($\theta_{XY} = 0°$), turbulent intensity (1%), and eddy viscosity ratio ($\frac{\mu_t}{\mu_{ref}} = 20$) are specified at the inlet, while for the outlet only the static pressure ($0Pa$) is prescribed. For the comparison, two Cartesian meshes, refined near the solid wall, are generated consisting of $25K$ cells (coarse) and $50K$ cells (fine). The simulation is terminated when the maximum flow equation residuals are below $||\mathcal{R}_i||_{\infty} < 10^{-12}$. The artificial compressibility parameter is set to $\beta = 10$.

In Figure 3.6a, the residual history of the simulation for the fine mesh is shown, wherein a reduction of approximately 12 orders is observed after 4000 iterations. This translates to about 20 minutes on 48 AMD EPYC 7401 (2.0 Ghz) processors and includes the generation of the CC mesh, shown in Figure 3.6b, forcing points and associated data structure, as well as its simulation. As seen in Figure 3.6b, all Cartesian cells have the same aspect ratio (1 : 1), excluding the CCs, that are cut by the flat plate at the southern border.



(a) (b)

Figure 3.6: Turbulent flow over a flat plate, $Re_L = 5 \times 10^6$ - (a) Flow equations residual history where a convergence of more than 10 orders of magnitude is observed and (b) CC mesh used for the simulation consisting of approximately $50K$ cells. The cell refinement levels are highest near the flat plate and progressively coarsen to reduce the total number of cells.

Figure 3.7 presents the non–dimensionalized eddy viscosity iso–areas as in the Turbulence Modeling Resource [256], i.e. the y–axis is expanded to show the area of interest, to allow immediate comparisons. A very good agreement is observed overall, with minor differences at the edge of the boundary layer, which were expected because of the different mesh sizes. The observed curved iso–areas at the edge of the boundary layer become more pronounced due to the expanded y–axis and the cell refinement level. It should be noted, however, that in the reference figure, a much finer mesh of 545×385 is used.

Figure 3.8 compares the computed skin friction coefficient distribution along the flat plate with numerical and experimental data. Also here, an excellent agreement is observed, with just minor discrepancies. For both simulations, the computed y^+ distribution shown in Figure 3.9 lies well inside the recommended bounds, $30 < y^+ < 300$.

Based on the obtained results, the forcing point technique allows for the simulation of turbulent flows with substantially relaxed mesh resolution requirements and a good representation of the turbulent boundary layer.

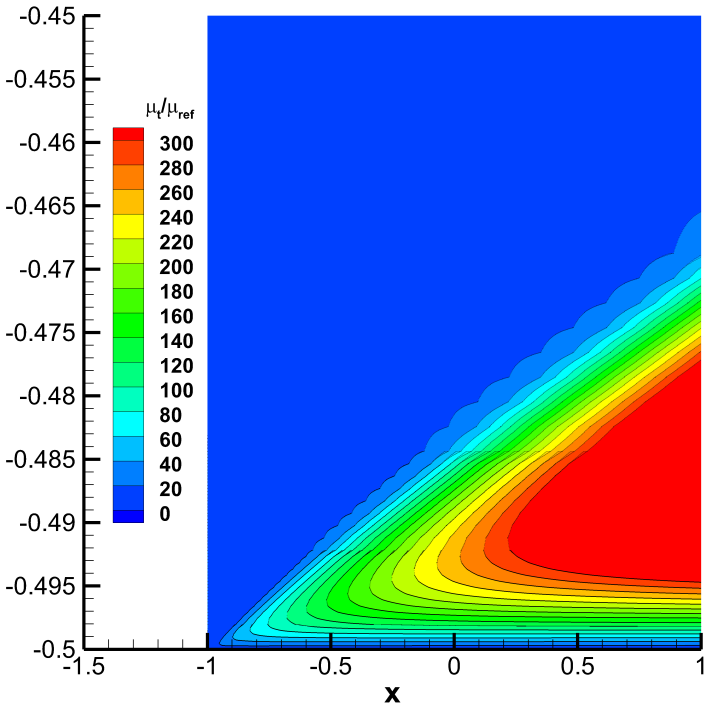
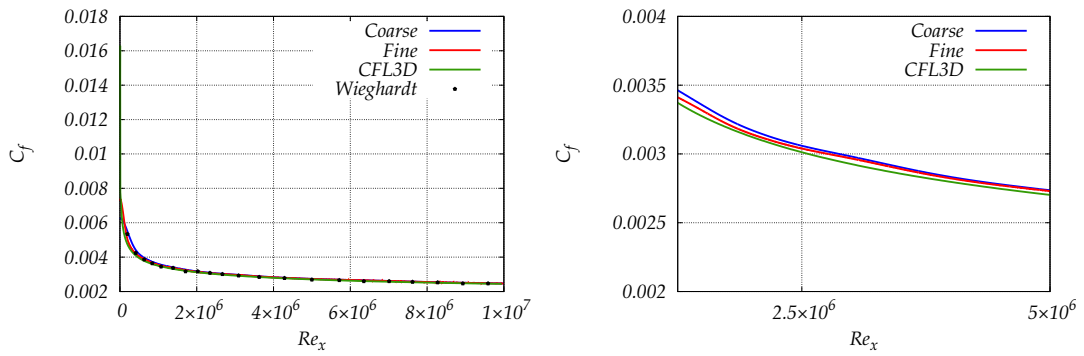


Figure 3.7: Turbulent flow over a flat plate, $Re_L = 5 \times 10^6$ - Non–dimensionalized eddy viscosity iso–areas. The y–axis is expanded based on the same iso–areas given in [256]. The curved iso–areas observed are intensified due to the expanded y–axis.



(a) **(b)**

Figure 3.8: Turbulent flow over a flat plate, $Re_L = 5 \times 10^6$ - (a) Comparison of the obtained skin friction coefficient with numerical [256] and experimental [277] results along the flat plate. (b) Close–up view of the computed skin friction coefficient for a coarse and fine CC mesh. Note that the reference values are computed using a 4× larger mesh than the fine CC mesh of this study.

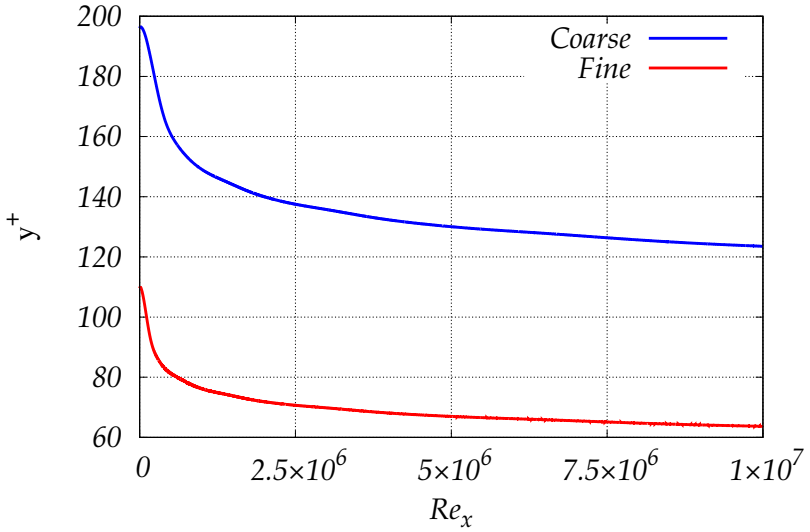


Figure 3.9: Turbulent flow over a flat plate, $Re_L = 5 \times 10^6$ - y^+ distribution for the coarse and fine CC meshes. Simulations using the wall function technique should ideally place the first point inside the logarithmic region.

3.2.6 90° Curved Channel $Re_W = 1 \times 10^5$

In the second case, a turbulent flow inside a curved channel is considered to assess the wall function technique over curved surfaces. For the simulation, the total pressure ($0.5Pa$), velocity direction ($\theta_{XY} = 0°$), turbulent intensity (2%) and eddy viscosity ratio ($\frac{\mu_t}{\mu_{ref}} = 2$) are specified at the inlet. At the outlet, the static pressure is zero. The Reynolds number $Re_W = 1 \times 10^5$ is based on the inlet width. The artificial compressibility parameter, in this case, is set to $\beta = 2$ due to the smaller velocities inside the channel. A Cartesian mesh consisting of approximately $75K$ cells is generated with finer refinement close to the solid walls and is shown in Figure 3.10. The close–up view of the CC mesh in Figure 3.10 near the curved body surface also depicts the positioning of the forcing points used for the wall function technique previously described. It can be seen that the forcing points are positioned at constant distances from the solid walls lying in the immediate (or diagonal) neighbors of the CCs. The computed y^+ distribution along the solid walls averaged at a value of 60.

Figure 3.11a shows the residual history of the simulation of the curved channel. Good convergence is observed as the flow equations residuals are reduced by more than 12 orders of magnitude after 14000 iterations. Due to the higher number of iterations, the total simulation time amounted to approximately 95 minutes on 48 AMD EPYC 7401 (2.0 Ghz) processors that

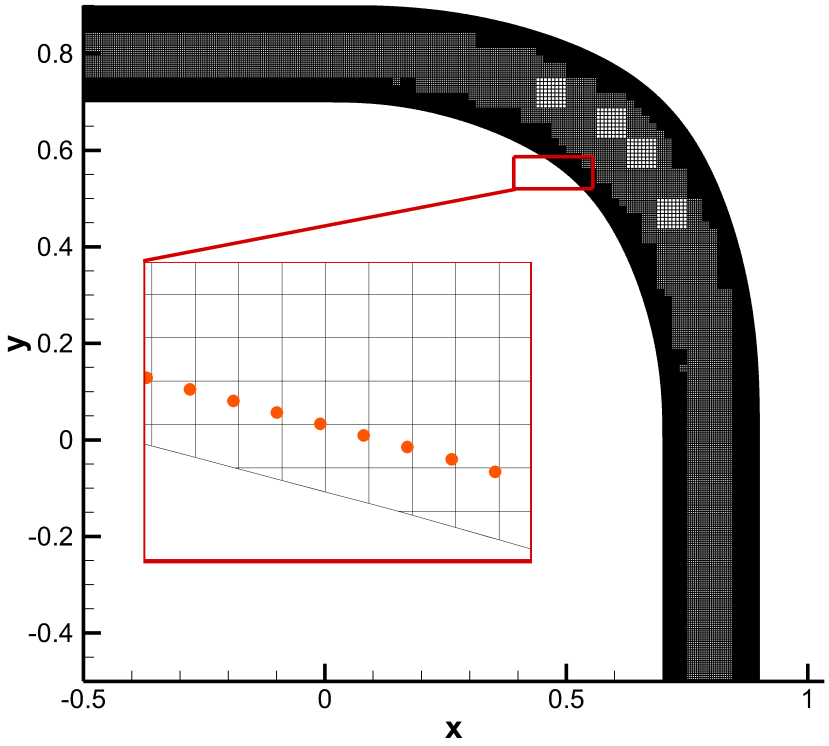
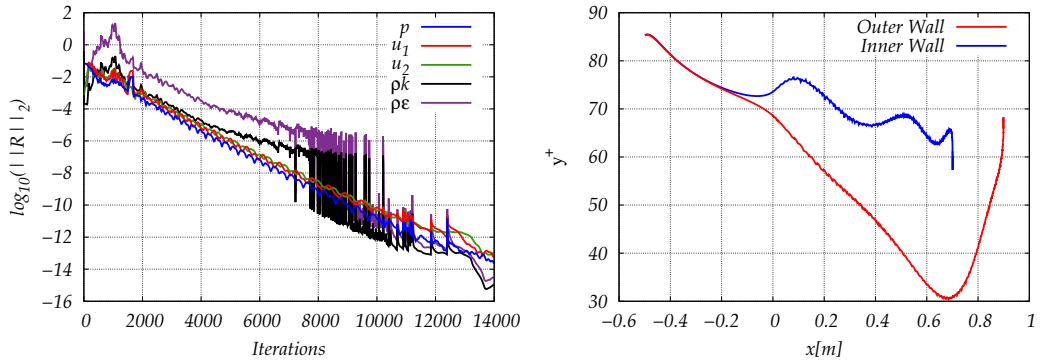


Figure 3.10: Turbulent flow over a 90° curved channel, $Re_W = 1 \times 10^5$ - The resulting CC mesh consisted of approximately $75K$ cells with additional refinement near the solid walls. A close–up view of the CC mesh, where the position of the forcing points along the solid walls is depicted, is also provided.

again includes the CC mesh generation process. In Figure 3.11b, the y^+ distributions for the inner and outer walls is presented. A smooth distribution is observed due to the forcing point technique that remedies the severely fluctuating y^+ values reported in the literature when no treatment is performed [31, 171].

In Figure 3.12, the non–dimensionalized eddy viscosity iso–areas in the curved channel case are depicted. An increase is observed at the outer wall of the curved section. Figures 3.13a and 3.13b show the distribution of the turbulent variables along the cross–section of the curved channel (dashed line in Figure 3.12). Near the solid walls the turbulent variable increases in magnitude, while at the bulk flow, these are minimal. At the outer wall, a reduced turbulent dissipation rate ε leads to the largest eddy viscosity values observed.



(a) (b)

Figure 3.11: Turbulent flow over a 90° curved channel, $Re_W = 1 \times 10^5$ - (a) Flow equations residual history, where a convergence of more than 12 orders of magnitude is observed after 14000 iterations. (b) y^+ distribution for the inner and outer walls of the curved channel. The forcing point technique implemented recovers a smooth y^+ distribution.

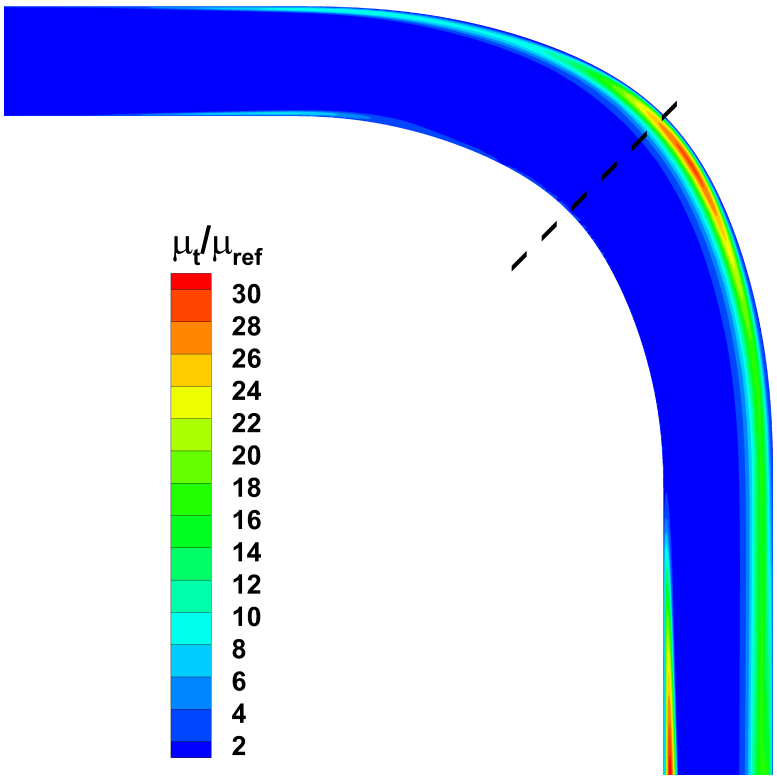
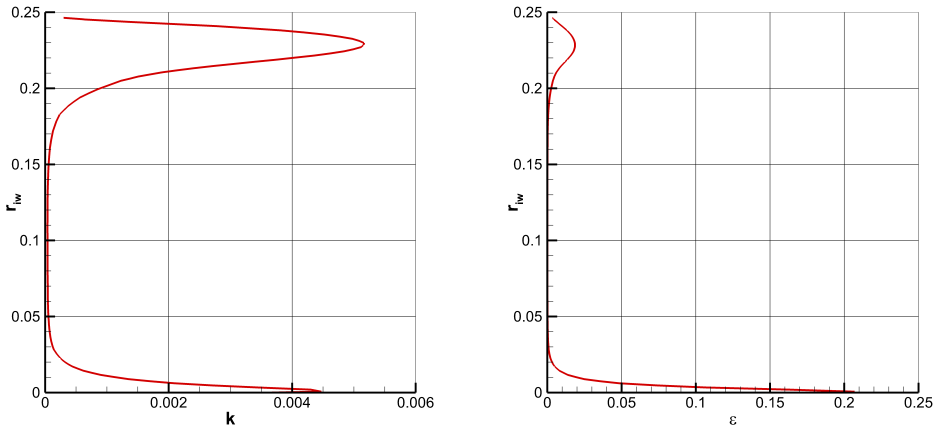


Figure 3.12: Turbulent flow over a 90° curved channel, $Re_W = 1 \times 10^5$ - Non–dimensionalized eddy viscosity iso–areas. The curved solid walls of the channel add an additional challenge to the wall function technique.



(a) **(b)**

Figure 3.13: Turbulent flow over a 90° curved channel, $Re_W = 1 \times 10^5$ - Distribution of the $k - \varepsilon$ flow variables along the cross–section shown in Figure 3.12. r_{iw} denotes the distance from the inner wall.

3.3 Simulation of Two–Phase Cavitating Flows

The CC solver was extended to facilitate two–phase cavitating flows by implementing a homogeneous TEM method. This method, commonly employed in these flows due to its reduced computational cost [6, 20, 138, 280], assumes a homogeneous mixture of liquid and vapour, i.e. a pseudo–fluid with averaged properties. It is based on the local kinematic and thermodynamic equilibrium between the two species, i.e. the constituents share common velocities, pressure and temperature. Even though the constituent phases are (usually) assumed to be incompressible, the mixture fluid is treated as a compressible fluid with a considerably varying density [129].

The governing equations express the conservation of momentum for the mixture fluid and the conservation of volume for each constituent, through the use of transport equations, to track them. Numerically, their interface is handled using a shock–capturing approach (pseudo–VoF, [98]), opting its higher flexibility (relaxed Courant number constraints, innate mass transfer handling) and simpler implementation, as compared to a (true) VoF method that can introduce additional difficulties when resolving *foamy* parts of the developed cavity [123]. Extension to cavitating flows is accomplished by additional source terms, derived from a cavitation model that quantifies the mass transfer between each constituent. Overall, the homogeneity assumption remains valid for a large range of cavitating flows, avoiding the need for slip velocities, as both

phases tend to remain relatively separate [123]. This makes the homogeneous–mixture method a very popular choice when simulating cavitating flows since it allows for the prediction of important cavity features such as large scale re–entrance jets, jet–cavity interactions [219], as well as the generation of baroclinic vorticity at the cavity closure region [220].

3.3.1 The Two–phase Navier–Stokes Equations, with Cavitation Modeling

In the homogeneous TEM method, the mixture has its properties altered throughout the computational domain based on the species concentration at each finite volume, namely

$$1 = a_l + a_v \tag{3.61}$$

$$\rho_m = a_l \varrho_l + (1 - a_l)\varrho_v \tag{3.62}$$

$$\mu_m = a_l \mu_l + (1 - a_l)\mu_v \tag{3.63}$$

where a denotes the volume fraction and subscripts m, l, v refer to the mixture, liquid and vapour phase, respectively. The equations of motion in the $2D(3D)$ Cartesian space x_j, $j = 1, 2\,(, 3)$ consisting of each constituent volume conservation equation and the mixture momentum conservation equations are

$$\begin{aligned}
\frac{\partial\left(\alpha_v u_j\right)}{\partial x_j} &= -\frac{1}{\varrho_v}\dot{m} \\
\frac{\partial\left(\alpha_l u_j\right)}{\partial x_j} &= \frac{1}{\varrho_l}\dot{m} \\
\frac{\partial\left(\rho_m u_k u_j\right)}{\partial x_j} + \frac{\partial p}{\partial x_k} &= \frac{\partial}{\partial x_j}\left(\mu_m\left[\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3}\frac{\partial u_\ell}{\partial x_\ell}\delta_k^j\right]\right), \qquad k = 1, 2\,(, 3)
\end{aligned} \tag{3.64}$$

where $\dot{m}$ represents the interphase mass transfer rate due to cavitation. Similarly to the single–phase formulation, the system of equation Eq. (3.64) is preconditioned to render it hyperbolic and allow for time–marching techniques. Preconditioning (presented in Kunz *et al.* [138]) is specifically derived to yield an eigensystem that is independent of the constituent phases density ratio $\frac{\varrho_l}{\varrho_v}$. Thus, the artificial speed of sound $c = \sqrt{u_n^2 + \beta^2 n_j n_j}$ still stands and leads to a similar numerical behavior for a large range of density ratios [138]. The preconditioned governing

equations become

$$\left(\frac{\alpha_v}{\rho_m \beta^2}\right)\frac{\partial p}{\partial \tau} + \frac{\partial \alpha_v}{\partial \tau} + \frac{\partial\left(\alpha_v u_j\right)}{\partial x_j} = -\frac{1}{\varrho_v}\dot{m} \tag{3.65}$$

$$\left(\frac{\alpha_l}{\rho_m \beta^2}\right)\frac{\partial p}{\partial \tau} + \frac{\partial \alpha_l}{\partial \tau} + \frac{\partial\left(\alpha_l u_j\right)}{\partial x_j} = \frac{1}{\varrho_l}\dot{m} \tag{3.66}$$

$$\frac{\partial\left(\rho_m u_k\right)}{\partial \tau} + \frac{\partial\left(\rho_m u_k u_j\right)}{\partial x_j} + \frac{\partial p}{\partial x_k} = \frac{\partial}{\partial x_j}\left(\mu_m\left[\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3}\frac{\partial u_\ell}{\partial x_\ell}\delta_k^j\right]\right) \tag{3.67}$$

The governing equations are re–arranged by adding the species transport equations to obtain the volume conservation equation for the whole mixture to bring the system of equations closer to the single–phase structure. Combining Eqs. (3.65) and (3.66) and considering Eq. (3.61) yields,

$$\left(\frac{[\alpha_v + \alpha_l]}{\rho_m \beta^2}\right)\frac{\partial p}{\partial \tau} + \frac{\partial\left([\alpha_v + \alpha_l]\right)}{\partial \tau} + \frac{\partial\left([\alpha_v + \alpha_l]\, u_j\right)}{\partial x_j} = \left(\frac{1}{\varrho_l} - \frac{1}{\varrho_v}\right)\dot{m} \tag{3.68}$$

$$\Leftrightarrow \left(\frac{1}{\rho_m \beta^2}\right)\frac{\partial p}{\partial \tau} + \frac{\partial u_j}{\partial x_j} = \left(\frac{1}{\varrho_l} - \frac{1}{\varrho_v}\right)\dot{m} \tag{3.69}$$

which closely resembles the mass conservation equation of single–phase flows (Eq. (3.3)), with the only difference being the addition of the interphase mass transfer rates and the mixture density. The governing two–phase equations of motion in $3D$ can, thus, be written as

$$\mathscr{R}_i := \Gamma_{in}\frac{\partial U_n}{\partial \tau} + \frac{\partial f_{ij}^I}{\partial x_j} - \frac{\partial f_{ij}^V}{\partial x_j} - S_i = 0 \tag{3.70}$$

where

$$\boldsymbol{\Gamma} = \begin{bmatrix} \left(\frac{1}{\varrho_m \beta^2}\right) & 0 & 0 & 0 & 0 \\ 0 & \rho_m & 0 & 0 & u_1 \Delta\varrho \\ 0 & 0 & \rho_m & 0 & u_2 \Delta\varrho \\ 0 & 0 & 0 & \rho_m & u_3 \Delta\varrho \\ \left(\frac{a_l}{\varrho_m \beta^2}\right) & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.71}$$

and

$$\boldsymbol{f}_j^I = \begin{bmatrix} u_j \\ \rho_m u_1\, u_j + \delta_j^1\, p \\ \rho_m u_2\, u_j + \delta_j^2\, p \\ \rho_m u_3\, u_j + \delta_j^3\, p \\ a_l u_j \end{bmatrix}, \qquad \boldsymbol{f}_j^V = \begin{bmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ 0 \end{bmatrix}, \qquad \boldsymbol{S} = \dot{m}\begin{bmatrix} \left(\frac{1}{\varrho_l} - \frac{1}{\varrho_v}\right) \\ 0 \\ 0 \\ 0 \\ \frac{1}{\varrho_l} \end{bmatrix} \tag{3.72}$$

with the vector of unknown flow variables being $\boldsymbol{U} = \left[p\ u_1\ u_2\ u_3\ a_l\right]^T$. Note that, in the simulation of two–phase flows, mixture density ρ_m (and viscosity μ_m) can vary based on the

species concentration (a_l) and, thus, the unknown flow variables include the static pressure p instead of the kinematic pressure $\breve{p}$. Accordingly, $\tau_{kj} = \mu_m \left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3} \frac{\partial u_\ell}{\partial x_\ell} \delta_k^j \right)$.

The introduction of the liquid volume fraction transport equation modifies the inviscid eigensystem by introducing one additional eigenvalue, i.e. $\tilde{\mathbf{\Lambda}} = \mathrm{diag}\,(\tilde{u}_n,\ \tilde{u}_n,\ \tilde{u}_n + \tilde{c},\ \tilde{u}_n - \tilde{c},\ \tilde{u}_n)$, and the corresponding eigenvector matrix (Appendix A.2).

3.3.1.1 Boundary Conditions

The introduced unknown variable a_l requires additional conditions to be specified at each domain boundary. Faces along the inlet and farfield boundaries impose a Dirichlet condition to the value of the liquid volume fraction, i.e. the mixture concentration is specified. Along the outlet wall boundaries, zero Neumann conditions are imposed by copying the interior domain variables to the boundary faces.

The boundary conditions imposed for the remaining unknown variables are identical to those of single–phase flows, see Section 3.1.3.

3.3.2 Cavitation Modeling

Several cavitation models can be used to quantify mass transfer due to cavitation Kunz *et al.* [138], Merkle [159], Senocak & Shyy [223], Singhal *et al.* [234], Zwart *et al.* [281]; a description of these models and their differences is given in Chapter 1. In this thesis, interphase mass transfer between liquid and vapour is modeled via an empirical finite rate cavitation model [137] wherein evaporation manifests when the local static pressure drops below the vapour pressure p_v. Condensation is based on a power series following the Ginzburg–Landau potential (Hohenberg & Halperin [99]). In more detail, the mass transfer rate is computed as

$$\dot{m} = \underbrace{\frac{C_{dest}\varrho_v}{\frac{1}{2}\varrho_l U_\infty^2 t_\infty}\ a_l\ \min(0, p - p_v)}_{evaporation\ (\dot{m}^-)} + \underbrace{\frac{C_{prod}\varrho_v}{t_\infty}\ {a_l}^2\,(1 - a_l)}_{condensation\ (\dot{m}^+)} \tag{3.73}$$

The empirical time rate constants C_{dest}, C_{prod} are non–dimensionalized with respect to the mean-flow time scale, $t_\infty = \frac{L}{U_\infty}$, L being the characteristic length and U_∞ a reference velocity.

To track the onset of cavitation, the non–dimensional cavitation number σ is used that expresses the potential of the flow to cavitate. It is defined as the non–dimensionalized difference of the local and vapour pressure, i.e.

$$\sigma = \frac{p_\infty - p_v}{\frac{1}{2}\varrho_l U_\infty^2} \tag{3.74}$$

where p_∞ is the reference static pressure.

3.3.3 Discretization of the Cavitation Model Source Terms

To solve the two–phase governing equation Eq. (3.70) a similar approach, used to extend the laminar solver to turbulent flows, presented in Section 3.2.3, is followed. The preconditioner matrix, inviscid, and viscous flux vectors are extended accordingly to account for the additional species transport equation. Furthermore, the cavitation model source terms are also included in the numerical solution algorithm 3.34, as per the turbulence model source terms. Following Vankateswaran *et al.* [262], the source term is split to enforce diagonal dominance. Its contributions to the diagonal matrix terms only exist if $p - p_v < 0$, i.e. evaporation is occurs and are

$$\frac{\partial \boldsymbol{S}^-}{\partial \boldsymbol{U}} = \frac{C_{dest}\varrho_v}{\frac{1}{2}\varrho_l U_\infty^2 t_\infty} \begin{bmatrix} \left(\frac{1}{\varrho_l} - \frac{1}{\varrho_v}\right) \\ \mathbf{0} \\ \frac{1}{\varrho_l} \end{bmatrix} \begin{bmatrix} a_l \\ \mathbf{0} \\ p - p_v \end{bmatrix}^T \tag{3.75}$$

The explicit source term $\boldsymbol{S}^+$ can cause instabilities to the numerical solution process at early iterations and is, therefore, under–relaxed [138].

3.4 Two–phase RANS Equations using the Cut–Cell Method

Following the methods presented in Sections 3.2 and 3.3, two–phase cavitating flows are simulated by coupling the standard $k - \varepsilon$ model with the governing equations, i.e. Eq. (3.70). Densities and viscosities existing in the turbulence model now refer to the mixture. Similarly, the mixture eddy viscosity reads [138, 222]

$$\mu_{m,t} = C_\mu \rho_m k \frac{k}{\varepsilon} \tag{3.76}$$

When $3D$ two–phase turbulent flows are simulated, the vector of unknown flow variables becomes $\boldsymbol{U} = \left[p\ u_1\ u_2\ u_3\ a_l,\ \rho_m k,\ \rho_m \varepsilon\right]^T$. Discretization aspects of the specific eigensystem, preconditioning matrix, inviscid and viscous flux vectors, and source vector follows that presented in Sections 3.2.3 and 3.3.1 and will be overviewed very briefly.

The $3D$ equations of motion governing turbulent, two–phase flows, that include cavitating phenomena, can be expressed as

$$\mathscr{R}_i := \Gamma_{in} \frac{\partial U_n}{\partial \tau} + \frac{\partial f_{ij}^I}{\partial x_j} - \frac{\partial f_{ij}^V}{\partial x_j} - S_i = 0 \tag{3.77}$$

with

$$\boldsymbol{\Gamma} = \begin{bmatrix} \left[\boldsymbol{\Gamma}\right]^{2\phi} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix}, \boldsymbol{f}_j^I = \begin{bmatrix} \boldsymbol{f}_j^{I,2\phi} \\ \varrho_m k u_j \\ \varrho_m \varepsilon u_j \end{bmatrix}, \boldsymbol{f}_j^V = \begin{bmatrix} \boldsymbol{f}_j^{V,2\phi} \\ \left(\mu_m + \frac{\mu_{m,t}}{\sigma_k}\right) \frac{\partial k}{\partial x_j} \\ \left(\mu_m + \frac{\mu_{m,t}}{\sigma_\varepsilon}\right) \frac{\partial \varepsilon}{\partial x_j} \end{bmatrix}, \boldsymbol{S} = \begin{bmatrix} \boldsymbol{S}^{2\phi} \\ \mathscr{P} - \varrho_m \varepsilon \\ \left(C_{1\varepsilon}\mathscr{P} - C_{2\varepsilon}\varrho_m \varepsilon\right) \frac{\varepsilon}{k} \end{bmatrix} \tag{3.78}$$

where $\boldsymbol{S}^{2\phi}$ includes the cavitation model source terms. Similarly, the mixture shear stress tensor is computed based on the mixture properties, viz. $\tau_{kj} = (\mu_m + \mu_{m,t}) \left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3} \frac{\partial u_\ell}{\partial x_\ell} \right)$.

As per the inviscid eigensystem, two additional eigenvalues are introduced, compared to the NS equations, leading to $\tilde{\boldsymbol{\Lambda}} = \mathrm{diag}\,(\tilde{u}_n,\ \tilde{u}_n,\ \tilde{u}_n + \tilde{c},\ \tilde{u}_n - \tilde{c},\ \tilde{u}_n,\ \tilde{u}_n,\ \tilde{u}_n)$. The linearly independent eigenvectors are constructed as in the single–phase formulation, i.e.

$$\boldsymbol{M} = \begin{bmatrix} \left[\boldsymbol{M}\right]^{2\phi} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{M}^{-1} = \begin{bmatrix} \left[\boldsymbol{M}^{-1}\right]^{2\phi} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix} \tag{3.79}$$

where $\left[\boldsymbol{M}\right]^{2\phi}$ is the corresponding two–phase eigenvector matrix given in Appendix A.2.

3.5 Cut-cell based Turbulent Flow Two–phase Simulations

In the following examples, the two–phase turbulent CC method is assessed in both $2D$ and $3D$ cases. This also allows the assessment of the forcing point technique in $3D$, as well as the accuracy of the implemented cavitation model, the effect of different cavitation numbers on the resulting flow, and the implemented turbulence model in the event of two–phase flows simulated using a CC–based solver.

3.5.1 NACA 66(MOD) hydrofoil $Re_c = 2 \times 10^6$

The first case considers the turbulent flow around the NACA 66(MOD) hydrofoil [40] which has a camber ratio of 0.02, a mean line of 0.8, a thickness ratio of 0.09, and a chord length of $0.1524m$. The hydrofoil was previously investigated both experimentally [228] and numerically [6, 167]. For example, in Morgut & Nobile [167] a stochastic optimization strategy is used to fine–tune the mass transfer rates under cavitating conditions. Herein, the hydrofoil is studied for three cavitation numbers $\sigma = \infty,\ 0.91,\ 0.84$ at $Re_c = 2 \times 10^6$, based on the hydrofoil chord length, and an angle of attack (AoA) of $\alpha_\infty = 4°$. The density ratio between the two phases is $\frac{\varrho_l}{\varrho_v} = 1000$. A Cartesian mesh, refined near the hydrofoil contour, consists of approximately

$80K$ cells, as shown in Figure 3.14 along with a close–up view of the generated forcing points, with an average $y^+ \approx 50$.

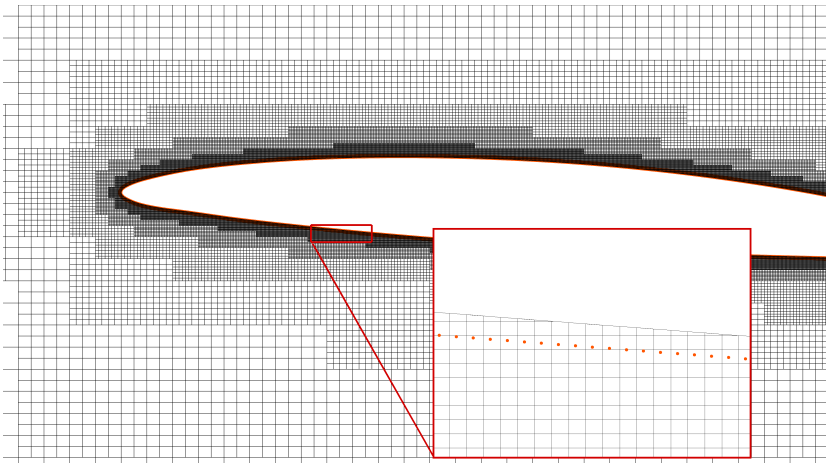
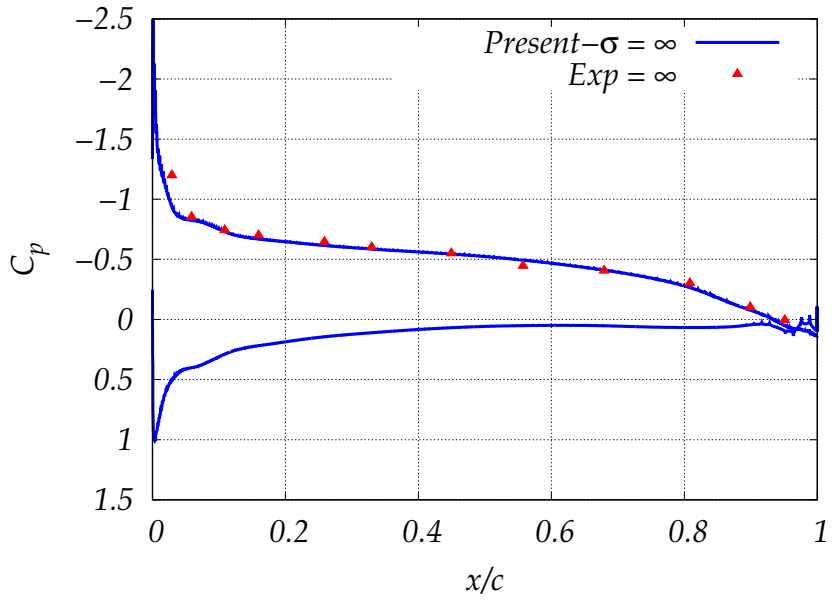


Figure 3.14: Turbulent, cavitating flow over the NACA 66(MOD) hydrofoil, $Re_c = 2 \times 10^6$ - Close–up view of the generated CC mesh that also the forcing points (orange dots) near the hydrofoil walls. The total number of cells is approximately $80K$.

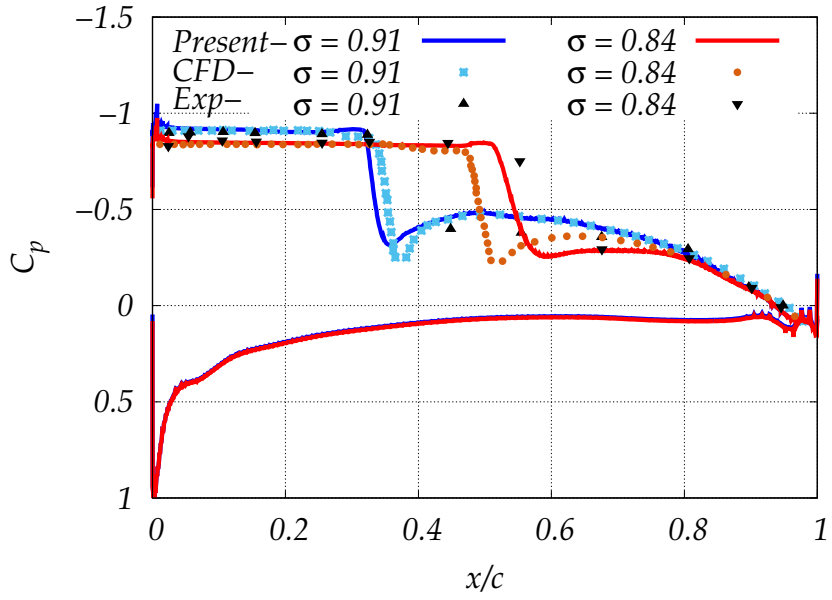
The best value–set of the mass transfer rates are found to be $C_{dest} = 1.5 \times 10^3$ and $C_{prod} = 135$ after several simulations [265]. To perform the said simulations, free–stream boundary conditions are imposed at the domain boundaries: a velocity magnitude ($10\,\frac{m}{s}$) and its direction ($\theta_{XY} = 4°$), the static pressure ($10^5 Pa$), a fully liquid phase ($a_l = 1$), the turbulent intensity (1%) and the mixture eddy viscosity ratio ($\frac{\mu_{m,t}}{\mu_l} = 20$). The simulations were performed in parallel on 48 AMD EPYC 7401 (2.0 Ghz) processors that required about 110 minutes to create the CC mesh with the forcing point substructure and solve the governing equations.

In Figure 3.15a, the computed surface pressure distribution is compared with the experimental data of Shen & Dimotakis [228] for non–cavitating canditions $\sigma = \infty$. A good agreement is observed verifying the adequacy of the generated CC mesh and the accuracy of the single–phase CC solver. Under cavitating conditions $\sigma = 0.84$ and 0.91, the computed surface pressure distributions compare favorably with both experimental [228] and numerical data [167], Figure 3.15b. The observed surface pressure distributions flattening on the hydrofoil suction side, signifies the presence of a vapour bubble.

Figure 3.16a shows the liquid phase iso–areas, and thus, the generated cavity for the $\sigma = 0.91$ case, while in Figure 3.16b the corresponding iso–bar areas are shown. Accordingly, in Figures 3.17a and 3.17b the same iso–areas are shown for $\sigma = 0.84$. Reducing the cavitation number increases the cavitation intensity, which in turn results in larger cavities around the hydrofoil



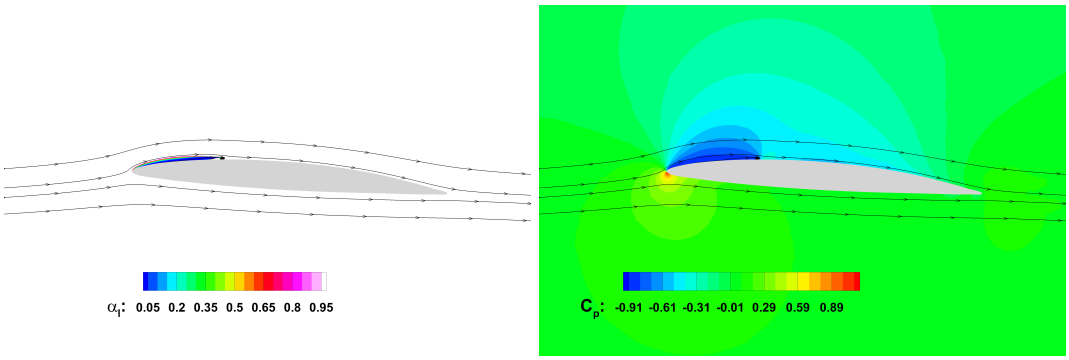
(a)



(b)

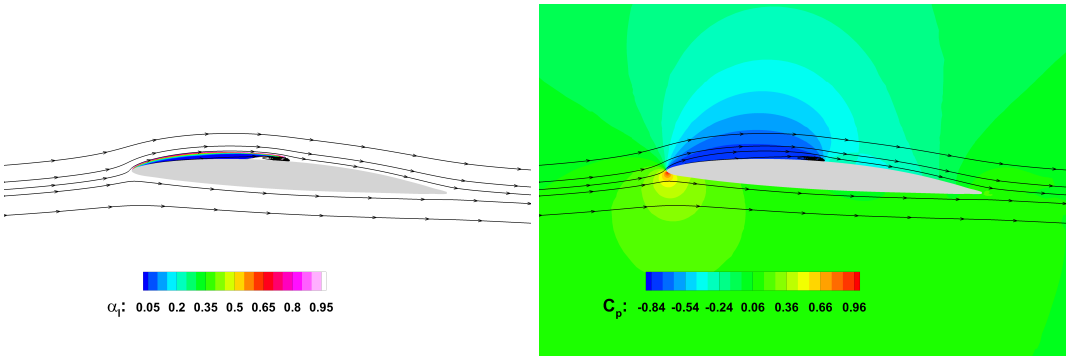
Figure 3.15: Turbulent flow over the NACA 66(MOD) hydrofoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 4°$, $\sigma = (\infty, 0.91, 0.84)$ - Computed surface pressure distributions compared with (a) experimental data [228] for the non–cavitating ($\sigma = \infty$) case and (a) both experimental (black points) [228] and numerical (light blue and orange points) [167] for two cavitating cases (with $\sigma = 0.91$ and 0.84).

that manifest on a greater extent.



(a) **(b)**

Figure 3.16: Turbulent, cavitating flow over the NACA 66(MOD) hydrofoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 4°$, $\sigma = 0.91$ - Computed iso–areas of (a) the liquid volume fraction and (a) pressure.



(a) **(b)**

Figure 3.17: Turbulent, cavitating flow over the NACA 66(MOD) hydrofoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 4°$, $\sigma = 0.84$ - Computed iso–areas of (a) the liquid volume fraction and (a) pressure.

3.5.2 Hemispherical Cavitator $Re_D = 136000$

To validate the CC two–phase solver in $3D$ cavitating flows, a hemispherical cavitator is considered. For the study, a $Re_D = 136000$, based on the diameter of the cavitator, is defined to be able to compare with the experimental data of Rouse & McNown [211], where different cavitator shapes are studied at various flow conditions. The inlet boundary conditions consist of the velocity magnitude ($10\frac{m}{s}$), its direction, a fully liquid phase ($a_l = 1$), the turbulent intensity (2%), and the eddy viscosity ratio of the mixture ($\frac{\mu_{m,t}}{\mu_l} = 20$) at the inlet, while at the outlet the static pressure ($10^5 Pa$) is defined. The liquid–vapour density ratio is $\frac{\varrho_l}{\varrho_v} = 100$. After

several evaluations the mass transfer rates required to properly capture the incepted cavity are found to be $C_{dest} = 9 \times 10^3$ and $C_{prod} = 80$.

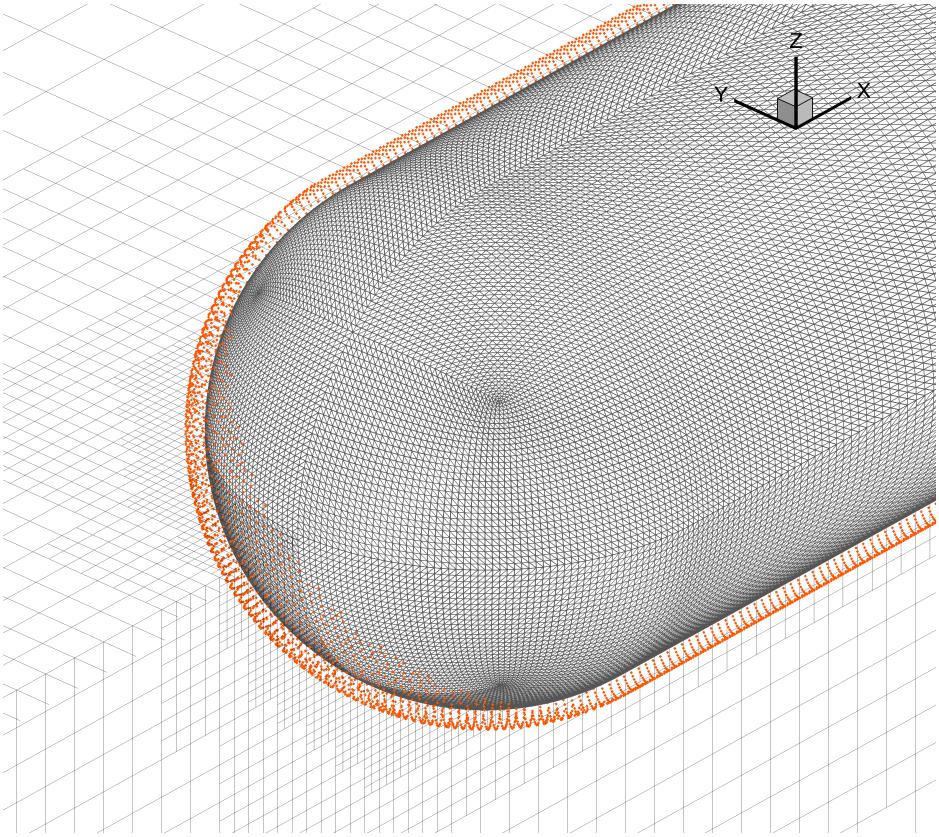


Figure 3.18: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$ - Close–up view of the hemispherical cavitator depicting two slices of the generated CC mesh, the discretized cavitator and the generated forcing points (orange dots). The total number of cells is approximately $450K$.

The generated CC mesh, progressively refined closer to the cavitator, that provided mesh independent solutions resulted in a computational mesh of approximately $450K$ cells and is comparable to the structured mesh used in Kunz *et al.* [137] to simulate the same cavitating flow. In Figure 3.18, a close–up view of the discretized cavitator, the CC mesh, and the generated forcing points, used for the wall function technique, are shown. Note that, properly capturing the curved surface of the cavitator is inherently challenging for the CC method and, thus, required a sufficiently dense discretized geometry. However, it can be seen that the CC method captures the curved surfaces, indicated by the generated forcing points that require the normal vectors along the solid walls. Figure 3.19 depicts an xy–slice of the generated

mesh. Additional refinement is introduced close to the cavitator head using the window–based refinement (Figure 2.5) to accurately resolve the inception of the cavity.

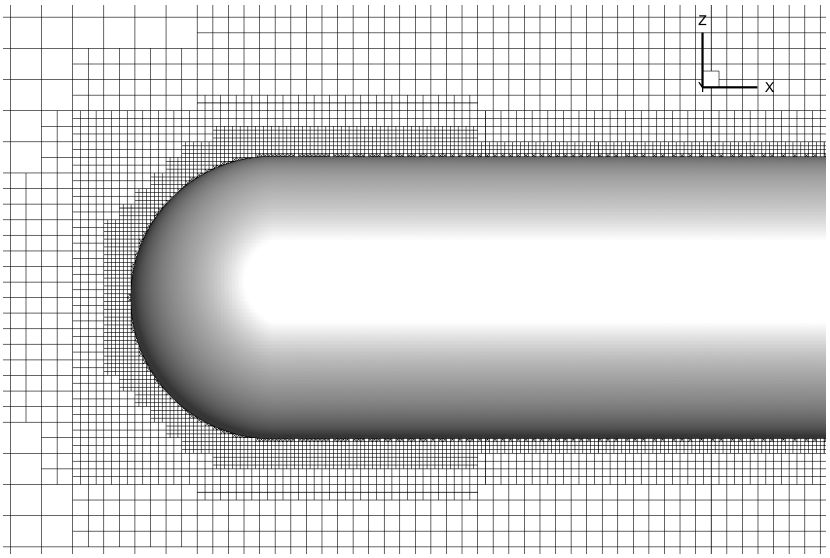


Figure 3.19: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$ - Close–up view of the mesh along the xy–plane showing the additional local refinement at the head of the cavitator, where large flow gradients are expected to appear.

3.5.2.1 0° Angle of Attack

Three different cavitating flows with $\sigma = \infty,\ 0.4,\ 0.3$ are simulated with an AoA of $\alpha_\infty = 0°$ for the oncoming liquid flow, i.e. $\theta_{ZX} = \theta_{XY} = 0°$. In Figure 3.20a, the residual history of the governing equations is shown for $\sigma = 0.4$, where a convergence is obtained after about 4000 iterations, while Figure 3.20b these are shown for the non–cavitating case $\sigma = \infty$. Comparison of the two figures reveals that the cavitating case requires more iterations to converge due to the appearance of a recirculation zone and the stiff mass transfer source terms. The simulations were performed on 48 AMD EPYC 7401 (2.0 Ghz) processors and required approximately 6 hours to complete. This includes the generation of the CC mesh and the forcing points, and the satisfaction of the governing equations. Figure 3.21 shows the computed surface pressure distribution over a meridional plane and is compared with the experimental data of [211]. The same trend can be seen, namely with lower cavitation numbers, larger cavities manifest and thus, the flattening of the surface pressure has a greater extent. At the cavity closure, a pressure overshoot is observed due to the local stagnation at the bubble closure, creating a recirculation zone as seen in Figure 3.22 where the iso–bar areas are shown along the same plane with

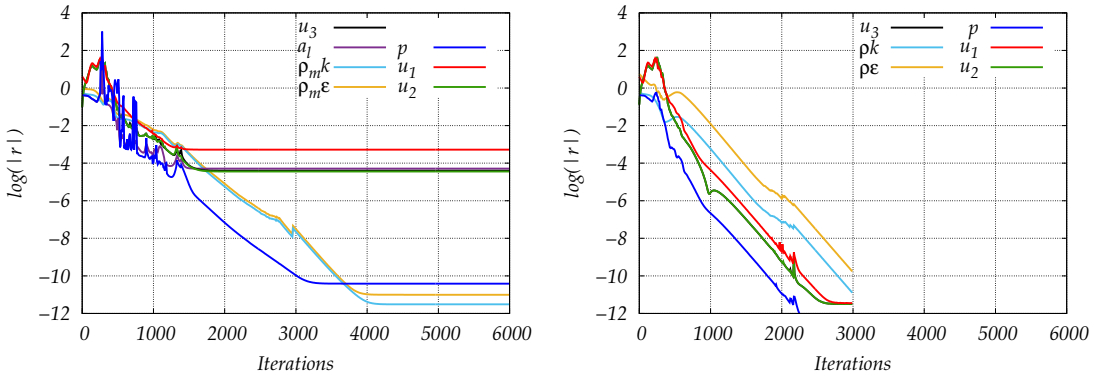


Figure 3.20: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$, $\alpha_\infty = 0°$, $\sigma = 0.4$ - Flow variable equations residual history for the cavitating $\sigma = 0.4$ and non–cavitating $\sigma = \infty$case.

streamlines.

In Figure 3.23, two different computed skin friction coefficient $C_f^s = \frac{\rho_s u_\tau^2}{\frac{1}{2}\varrho_l U_\infty^2}$, $s = \{m, l\}$ are depicted over a meridional plane. Even though only the mixture skin friction coefficient C_f^m is true to the performed simulation, remarks w.r.t. the mixture density distribution along the geometry surface and its effect of the presence of vapour can be made, by quantitatively comparing the two surface distributions. It is evident that the presence of vapour significantly reduces the skin friction at the cavity location due to its smaller, compared to liquid, density, while away from the cavity, where pure liquid exists, the same skin friction is recovered, since the mixture density equals the liquid density. At the cavity closure region, the flow is stagnated locally, resulting in an almost zero skin friction coefficient for both cases. Note that a smooth skin friction distribution is obtained in the presence of curved surfaces, which is challenging when performing simulations using the CC method in high Reynolds numbers [31, 32, 249], due to the approach presented in Section 3.2.

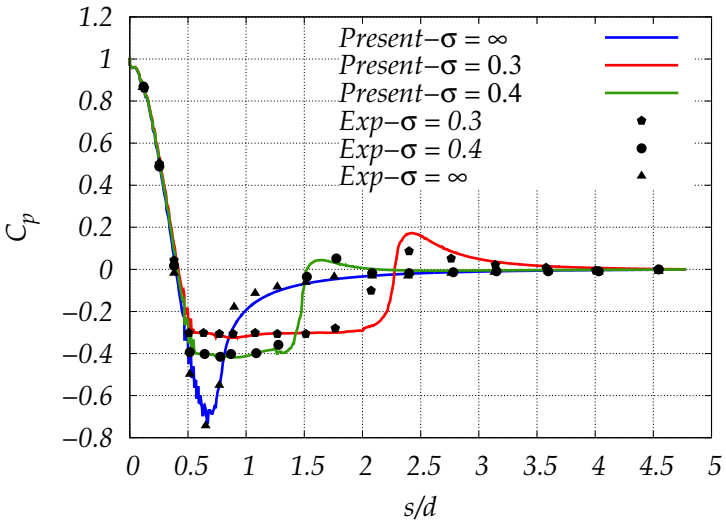


Figure 3.21: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$, $\alpha_\infty = 0°$, $\sigma = 0.4$ - Comparison of the computed surface pressure distribution over a meridional plane plane of the cavitator with the experimental data of Rouse & McNown [211].

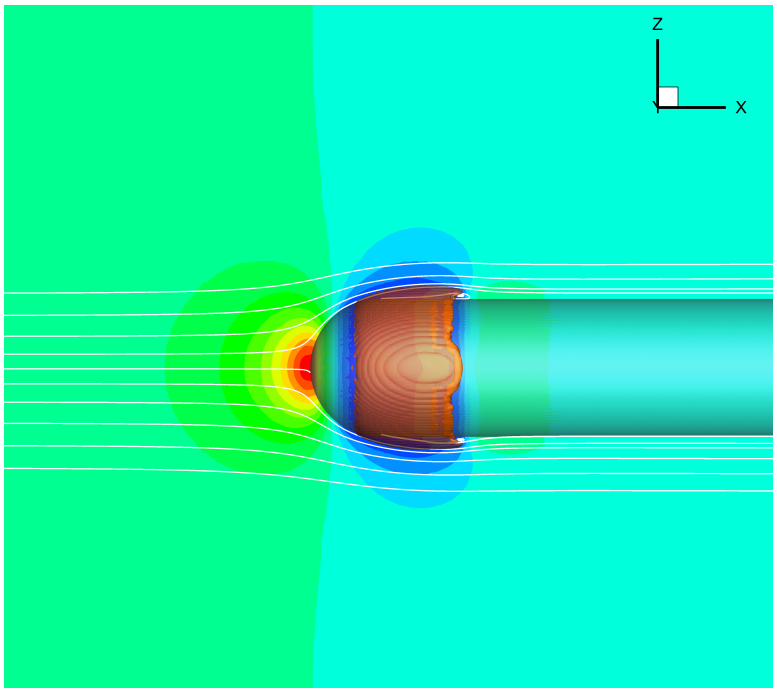


Figure 3.22: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$, $\alpha_\infty = 0°$, $\sigma = 0.4$ - Iso–bar areas along the $x = 0$ plane with the $a_l = 0.9$ iso–surface. Included streamlines show the recirculation areas aft the cavity.

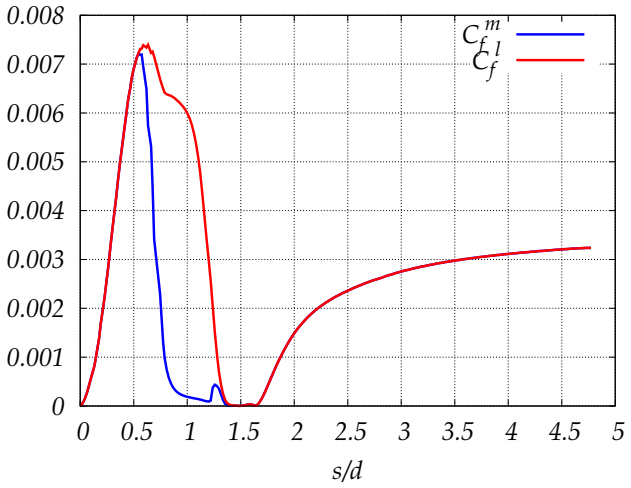


Figure 3.23: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$, $\alpha_\infty = 0°$, $\sigma = 0.4$ - Computed skin friction coefficient over a meridional plane. A sudden reduction is observed at the cavity location $0.6 < s/d < 1.2$ indicating the presence of vapour, followed by an almost zero skin friction coefficient due to the recirculation aft the cavity. C_f^m refers to the computed skin friction coefficient, while C_f^l assumes a pure liquid phase to quantitatively illustrate the effect of the presence of vapour on the skin friction.

3.5.2.2 10° Angle of Attack

The same hemispherical cavitator is simulated for an AoA of $\alpha_\infty = 10°$ ($\theta_{XY} = 10°$) at cavitation number $\sigma = 0.30$. In this case, the asymmetry of the resulting flow solution required a larger computational domain, which is elongated along the meridional direction. This leads to an approximately 15% increase in the total number of cells. In Figure 3.24, the computed surface pressure distribution is depicted along the $z = 0$ plane, revealing the existence of a small cavity at the bottom side and a significantly larger one at the top side of the cavitator.

Figure 3.25 shows the resulting cavity with selected streamlines, colored by the flow velocity magnitude, that pinpoint the recirculation zone at the top side. As noted in Kunz *et al.* [137], the flow is highly $3D$, and the cavity is largest off the $z = 0$ plane. At the top side, the large recirculation zone weakens the associated pressure recovery resulting in the more gradual surface pressure distribution observed in Figure 3.24. The said recirculation also causes the collapse of the cavity at the top side. The recirculation zone weakens off the $z = 0$ plane, allowing the cavity to extend longer, which leads to the depicted cavity shape. At the bottom side, the generated vapour is convected toward the top side due to the non–zero AoA, resulting in a much smaller cavity.

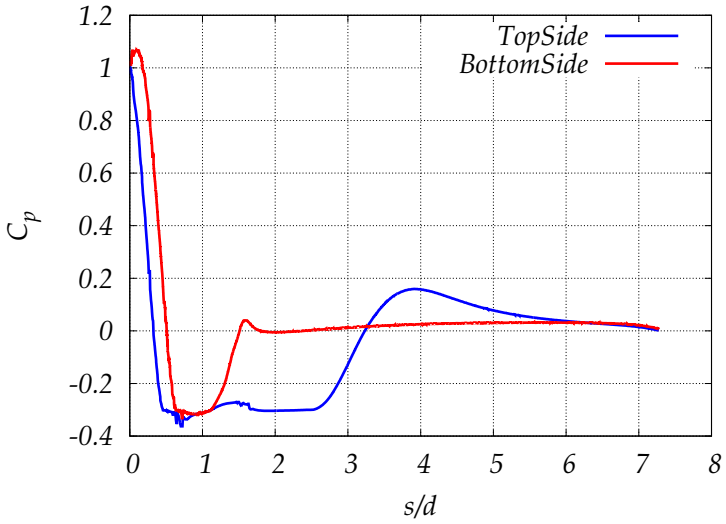


Figure 3.24: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$, $\alpha_\infty = 10°$, $\sigma = 0.3$ - Computed surface pressure distribution along the $z = 0$ plane. The non–zero AoA result in the generation of an asymmetric cavity occurs which is elongated along the top side.

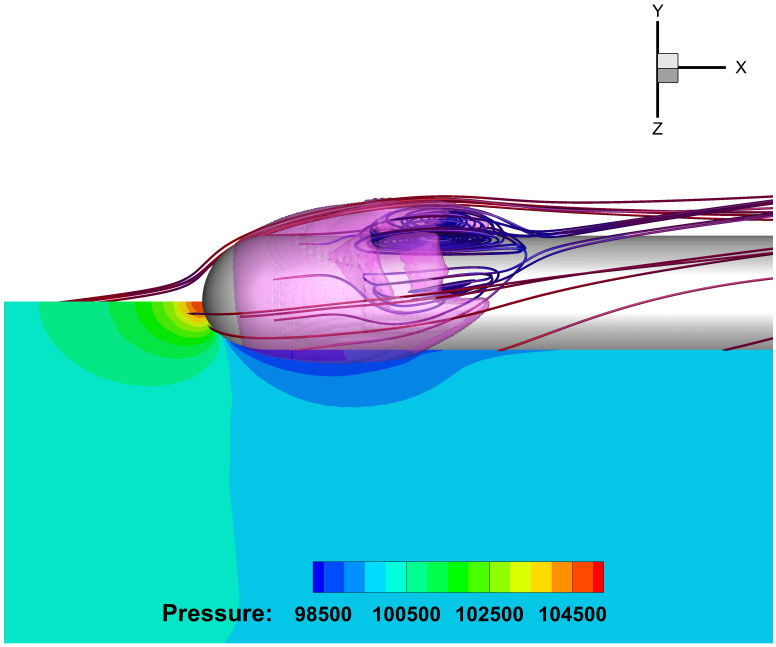


Figure 3.25: Turbulent, cavitating flow over a hemispherical cavitator, $Re_D = 136000$, $\alpha_\infty = 10°$, $\sigma = 0.3$ - The isosurface of $a_l = 0.9$ reveals the developed cavity location. The selected streamlines show the occurring recirculation zone aft the cavity at the top side of the cavitator and are colored by the velocity magnitude. The extracted slice along the yz–plane shows the iso–bar areas. At the bottom side, a much smaller cavity occurs.

3.6 Concluding Remarks

In this chapter, flow models extending the CC–based solver to turbulent and cavitating two–phase flows were presented. These include the implementation of the standard $k-\varepsilon$ model, the CC–specific use of linelets (forcing points) that enable the use of the wall functions technique. In addition, two–phase cavitating flows are simulated via the homogeneous mixture assumption and a cavitation model. Its extension to the RANS equations is performed by expressing the turbulence model w.r.t. the mixture. The above–mentioned are integrated into the precursor software following Object–Oriented Programming (OOP) and can now flexibly alternate between different flow models.

The single–phase RANS solver is demonstrated in two examples; a flat plate in which the computed skin friction coefficient showed excellent agreement with numerical and experimental data, and a curved channel verifying that the forcing point technique can compute smoothly varying friction velocities on curved surfaces. Furthermore, the implemented linelet technique, cavitation model and two–phase RANS solver are assessed using $2D$ and $3D$ test cases from the literature. In the examples presented, good agreement is observed with affordable mesh requirements. As per the implemented cavitation model, relative sensitivity and case–dependency to the empirical terms is observed that requires calibration [265].

Regarding the computational resources footprint of each extension: the ability to simulate turbulent flows using the CC method requires the storage of additional connectivity information and iteratively solving algebraic equations at the solid walls. The former is a unique CC requirement to allow modeling the turbulent boundary layer with relative accuracy and significantly alleviating mesh resolution requirements; as an example, in [31], the $2D$ flat–plate turbulent simulations presented without wall functions and a $y^+ \approx 10$ (which is still high for *Low–Reynolds* turbulence models) requires $O(10^6)$ cells. The cost of solving the algebraic equations is presumably similar to that of conventional body–fitted solvers. The simulation of two–phase cavitating flows in the $3D$ case requires approximately $2\times$ more iterations to reach a converged solution, compared to the single–phase simulation $\sigma = \infty$ that needs about 1600 more iterations using the same mesh, Figure 3.20. This increase can be attributed to the stiff cavitation–related source terms, the lower CFL requirements and the introduced cavitation dynamics that influence both the numerical solution process and the resulting flow features.

Chapter 4

The Continuous Adjoint Method in Single– and Two–phase flows

This chapter presents the development of the adjoint counterparts of the flow models presented in Chapter 3. The adjoint method can be formulated following either the continuous or discrete approach [103]. In the former [186, 191, 255, 282], the adjoint equations are derived by differentiating the PDEs governing the flow and are then discretized and numerically solved. In the latter [13, 172, 174], the adjoint equations emerge by differentiating the discretized form of the governing PDEs and directly result in an adjoint problem in discrete form.

Herein, the continuous adjoint problem is formulated based on the SI approach [187], benefiting from the use of the CC that exhibits zero internal volume mesh variations and aims at the gradient–based optimization of aero/hydrodynamic applications that incorporate turbulence and two–phase effects. Concerning the inclusion of turbulence effects, the implemented turbulence model is differentiated to obtain the exact objective function gradient expression, avoiding the *frozen turbulence* assumption. This choice follows related literature indicating that this assumption can potentially lead to significantly inaccurate objective function gradient, likely affecting the optimization path [71, 189, 191, 282–284]. Concerning two–phase adjoint formulations, relatively scarce literature exists, and some applications are found for free–surface flows [134, 183, 184, 242]; regarding two–phase flows featuring cavitating, or mass transfer in general, even fewer exist [17, 35]. Hitherto, the commonly followed approach is to perform optimization studies in flows featuring cavitation via a single–phase flow and an objective function that quantify areas below the vapour pressure.

In Boger [35], the continuous adjoint method was derived for the barotropic model, wherein

a barotropic state law is algebraically connecting density with pressure and, thus, avoids the introduction of species transport equations and mass transfer source terms. This approach introduces several limitations, for instance, no baroclinic vorticity can be generated [86] due to the introduction of an artificial equation of state that coupled density with pressure. Instead, herein, a TEM approach is followed, wherein species transport equations and mass transfer source terms are introduced and require additional differentiation when formulating the adjoint problem. Thus, the mathematical development of the continuous adjoint problem for cavitating two–phase flows using a TEM approach in conjunction with the CC method [266] has several novel aspects and is pursued throughout this thesis.

Initially, a brief introduction to the adjoint–based optimization is given, followed by the derivation of the adjoint problem when considering turbulent cavitating two–phase flows. The differentiation of $k-\varepsilon$ turbulence models has been presented for single–phase incompressible flows (with constant density) in [189, 283] by PCOpt. However, in two–phase flows, the mixture density varies greatly and, thus, is incorporated and differentiated to derive the adjoint counterpart. This results in additional terms appearing in the adjoint problem arising from mixture density and viscosity. Concerning the two–phase mean–flow equations, the same remarks apply, with the additional differentiation of the cavitation model source terms, presented in [266].

This chapter focuses on the mathematical formulation of the continuous adjoint problem. In addition, the objective functions used in this thesis to perform gradient–based optimization are presented and differentiated.

4.1 Gradient–based Optimization

In gradient–based optimization, the gradient of the objective function J w.r.t. a set of design variables $b_i,\ i=1,...,N$ must be computed. In the most general case, the objective function J may include an integral quantity that is the entire or part of the boundary of the domain (S^J) and/or a field integral over the entire computational domain (Ω), i.e.

$$J=\int_{S^J} j_S dS+\int_{\Omega} j_\Omega d\Omega \tag{4.1}$$

Differentiating the objective function w.r.t. the design variables and applying the Leibniz theorem for integral variations yields the sought objective function gradient, also referred to as SDs

[116, 186, 191],

$$\begin{aligned}\frac{\delta J}{\delta b_i} = &\int_\Omega \frac{\partial j_\Omega}{\partial U_n}\frac{\partial U_n}{\partial b_i}d\Omega + \int_S j_\Omega \hat{n}_k \frac{\delta x_k}{\delta b_i}dS + \int_{S^J}\frac{\partial j_S}{\partial U_n}\frac{\delta U_n}{\delta b_i}dS \\ &+ \int_{S^J}\frac{\partial j_S}{\partial\left(\tau_{kj}\hat{n}_k\hat{n}_j\right)}\frac{\delta\left(\tau_{kj}\hat{n}_k\hat{n}_j\right)}{\delta b_i}dS + \int_{S^J}\frac{\partial j_S}{\partial\left(\tau_{kj}\hat{n}_k\hat{t}_j\right)}\frac{\delta\left(\tau_{kj}\hat{n}_k\hat{t}_j\right)}{\delta b_i}dS \\ &+ \int_{S^J}\frac{\partial j_S}{\partial \hat{n}_m}\frac{\delta(\hat{n}_m dS)}{\delta b_i} + \int_{S^J}\frac{\partial j_S}{\partial x_k}\frac{\delta x_k}{\delta b_i}dS\end{aligned} \tag{4.2}$$

where $\frac{\delta()}{\delta b_i}$, $\frac{\partial()}{\partial b_i}$ refer to the total and partial derivatives, and $x_k,\ k = 1, 2(, 3)$ is the position in space. The total derivative accounts for the effect caused by the position change of a mesh node (in the discrete sense), whereas the partial derivative accounts only for flow variable variations. The two are correlated as follows

$$\frac{\delta()}{\delta b_i} = \frac{\partial()}{\partial b_i} + \frac{\partial()}{\partial x_k}\frac{\delta x_k}{\delta b_i} \tag{4.3}$$

In Eq. (4.2) derivatives of flow quantities w.r.t. the design variables $\frac{\delta U_n}{\delta b_i}$, $\frac{\delta \tau_{kj}}{\delta b_i}$ are associated with high computational cost. For example, if the SDs are to be approximated using FDs to avoid additional implementation aspects regarding the computation of these terms, say using central differences for second–order accuracy, the total cost corresponds to $2N$ EFS. The reason for this is that each design variable has to be modified by an infinitesimally small increment in both directions to compute the corresponding J values and approximate its derivative w.r.t. the corresponding design variable, through subtraction and division by twice the increment used. Alternatively, the adjoint problem circumvents the computation of derivatives ($\frac{\delta U_n}{\delta b_i}$, $\frac{\delta \tau_{kj}}{\delta b_i}$). This makes the cost of solving the adjoint problem independent to the number of design variables and the total cost to 2 EFS, since solving the adjoint problem costs approximately 1 EFS.

4.2 Formulation of the Continuous Adjoint Method

In the continuous adjoint method, the objective function is augmented by the field (over Ω) integral of the product of Lagrangian multipliers $\boldsymbol{\psi}$ (adjoint variables) and the residuals $\mathscr{R}$ of the governing equations; this gives rise to the so–called Lagrangian

$$L = J + \int_\Omega \psi_n \mathscr{R}_n d\Omega \tag{4.4}$$

where an adjoint variable field is introduced for each governing equation. Upon convergence of the governing equations $\boldsymbol{\mathscr{R}} = \mathbf{0}$, $L = J$, $\frac{\delta L}{\delta b_i} = \frac{\delta J}{\delta b_i}$ stand and, thus, the objective function

gradient can be computed by differentiating Eq. (4.4).

The continuous adjoint method is formulated for $3D$ turbulent, cavitating two–phase flows that are solved similarly to the governing equations Eq. (3.77), i.e. a coupled adjoint system of PDEs is derived and simultaneously solved. For convenience, the vector of unknown flow variables is reiterated $\boldsymbol{U} = \left[p\ u_1\ u_2\ u_3\ a_l,\ \rho_m k,\ \rho_m \varepsilon\right]^T$. Thus, the emerging vector of unknown adjoint variables $\boldsymbol{\psi}$ consists of the adjoint pressure ψ_1, (adjoint) velocity components $\psi_j,\ j = 2, 3(, 4)$, (adjoint) liquid fraction ψ_5, and (adjoint) turbulent kinetic energy ψ_6 and dissipation rate ψ_7, both multiplied by the fluid density. The adjoint formulation for flows that are a subset of the presented one, i.e. other combinations of $2D/3D$ inviscid/laminar/turbulent single–/two–phase flows, arise by omitting the appropriate terms in the following subsections. For example, for $2D$ inviscid, cavitating two–phase flows, the viscous vector and terms involving turbulence variables are zero, the cavitation model source terms remain, $\boldsymbol{U} = \left[p\ u_1\ u_2\ a_l\right]^T$ and, thus, $\boldsymbol{\psi} = \left[\psi_1\ \psi_2\ \psi_3\ \psi_5\right]^T$.

The adjoint problem to incompressible flows, solved via the artificial compressibility method, can be formulated following two different (though equivalent) approaches. The first approach derives the *preconditioned adjoint equations* using the non–preconditioned governing equations as a point of reference. The arising adjoint problem requires preconditioning to facilitate time–marching techniques [255]. In the second one, the governing equations are multiplied by the inverse preconditioning matrix and are subsequently differentiated. This results in the *adjoint to the preconditioned equations* problem that inherently has the same eigensystem (but with opposite eigenvalues) of the governing equations. This approach is most convenient when the governing equations are solved in a similar fashion [22].

Their equivalence can briefly be shown for either single– or two–phase flows as follows.

Let $\tilde{\boldsymbol{\mathscr{R}}} := \boldsymbol{\Gamma}^{-1}\boldsymbol{\mathscr{R}}$ be the preconditioned residual vector, $\boldsymbol{\Gamma}$ is the preconditioner matrix and $\boldsymbol{\mathscr{R}}$ is the non–preconditioned residual vector, (see Eq. (3.71)). In the first approach, the differentiation of the field integral in Eq. (4.4) results in

$$\frac{\delta}{\delta b_i}\int_\Omega \psi_n \mathscr{R}_n d\Omega = \int_\Omega \psi_n \frac{\partial \mathscr{R}_n}{\partial b_i} d\Omega + \int_S \psi_n \mathscr{R}_n \hat{n}_k \frac{\delta x_k}{\delta b_i} dS \tag{4.5}$$

whereas in the second approach

$$\frac{\delta}{\delta b_i}\int_\Omega \psi_n \tilde{\mathscr{R}}_n d\Omega = \int_\Omega \psi_n \frac{\partial \tilde{\mathscr{R}}_n}{\partial b_i} d\Omega + \int_S \psi_n \tilde{\mathscr{R}}_n \hat{n}_k \frac{\delta x_k}{\delta b_i} dS$$

$$
\begin{aligned}
&= \int_\Omega \psi_n \frac{\partial \left(\Gamma^{-1}_{nm}\mathscr{R}_m\right)}{\partial b_i} d\Omega + \underbrace{\int_\Omega \psi_n \frac{\partial \Gamma^{-1}_{nm}}{\partial b_i}\mathscr{R}_m d\Omega}_{=0} + \int_S \psi_n \Gamma^{-1}_{nm}\mathscr{R}_m \hat{n}_k \frac{\delta x_k}{\delta b_i} dS \\
&= \int_\Omega \psi_n \Gamma^{-1}_{nm} \frac{\partial \mathscr{R}_m}{\partial b_i} d\Omega + \int_S \psi_n \Gamma^{-1}_{nm}\mathscr{R}_m \hat{n}_k \frac{\delta x_k}{\delta b_i} dS \\
&= \int_\Omega \tilde{\psi}_m \frac{\partial \mathscr{R}_m}{\partial b_i} d\Omega + \int_S \tilde{\psi}_m \mathscr{R}_m \hat{n}_k \frac{\delta x_k}{\delta b_i} dS
\end{aligned} \tag{4.6}
$$

where $\tilde{\boldsymbol{\psi}} = \boldsymbol{\Gamma}^{-T}\boldsymbol{\psi}$ was used. Therefore, in Eq. (4.6) the adjoint equations are derived and solved for the preconditioned adjoint variables. On the contrary, in Eq. (4.5) the non–preconditioned adjoined equations are derived and require to be preconditioned. A more thorough discussion and analysis, showing the equivalence of the two formulations, in terms of SD computation accuracy, can be found in Asouti *et al.* [21], Asouti [22]. Herein, *preconditioned adjoint equations* are derived, ergo using Eq. (4.5), and require preconditioning to obtain an eigensystem with opposite eigenvalues to the governing equations. Based on the aforementioned, differentiation of the Lagrangian results in

$$
\begin{aligned}
\frac{\delta L}{\delta b_i} &= \frac{\delta J}{\delta b_i} + \int_\Omega \psi_n \frac{\partial \mathscr{R}_n}{\partial b_i} d\Omega + \int_S \psi_n \mathscr{R}_n \hat{n}_k \frac{\delta x_k}{\delta b_i} dS \\
&= \frac{\delta J}{\delta b_i} + \int_\Omega \psi_n \frac{\partial}{\partial b_i} \left[\underbrace{\frac{\partial f^I_{nj}}{\partial x_j}}_{\mathcal{T}^I} \underbrace{- \frac{\partial f^V_{nj}}{\partial x_j}}_{\mathcal{T}^D} \underbrace{- S_n}_{\mathcal{T}^S} \right] d\Omega + \int_S \psi_n \mathscr{R}_n \hat{n}_k \frac{\delta x_k}{\delta b_i} dS
\end{aligned} \tag{4.7}
$$

In the following sections, the convective $\mathcal{T}^I$ (Section 4.2.1), diffusive $\mathcal{T}^D$ (Section 4.2.2) and source $\mathcal{T}^S$ (Section 4.2.3) terms in Eq. (4.7) are developed separately to identify terms that contribute to the Field Adjoint Equations (FAEs), Adjoint Boundary Conditions (ABCs) and the expression computing the SDs.

The FAEs define a new system of PDEs, obtained by circumventing the computationally intensive task of computing the derivatives of flow variables w.r.t. the design variables $\frac{\partial \boldsymbol{U}}{\partial b_i}$ thought Ω. To do so, all volume integrals containing the said derivatives are collected and have their multiplier set to zero, i.e. $\int_\Omega \underbrace{(FAE)_n}_{=\mathbf{0}} \frac{\partial U_n}{\partial b_i} d\Omega$. Boundary integrals are treated using a similar approach. In this case, partial derivatives are first transformed to total derivatives using Eq. (4.3), to account for geometric changes as well. Boundary terms multiplied by the total derivatives of flow variables $\frac{\delta \boldsymbol{U}}{\delta b_i}$ are eliminated, and this gives rise to the ABCs, while the remaining terms, boundary integrals containing geometric derivatives, contribute to the SDs expression. The computation of geometric derivatives at the boundaries is a relatively low–cost

process and is discussed in Section 5.1, especially for the CC method.

4.2.1 Differentiation of the Inviscid Terms

Term $\mathcal{T}^I$ in Eq. (4.7), containing the inviscid terms multiplied by the adjoint variables, is expanded as follows

$$\begin{aligned}
\mathcal{T}^I &:= \int_\Omega \psi_n \frac{\partial}{\partial b_i}\left(\frac{\partial f^I_{nj}}{\partial x_j}\right) d\Omega = \int_\Omega \psi_n \frac{\partial}{\partial x_j}\left(\frac{\partial f^I_{nj}}{\partial b_i}\right) d\Omega \\
&= \int_\Omega \frac{\partial}{\partial x_j}\left(\psi_n \frac{\partial f^I_{nj}}{\partial b_i}\right) d\Omega - \int_\Omega \frac{\partial \psi_n}{\partial x_j}\frac{\partial f^I_{nj}}{\partial b_i} d\Omega \\
&= \int_\Omega \frac{\partial}{\partial x_j}\left(\psi_n \frac{\partial f^I_{nj}}{\partial b_i}\right) d\Omega - \int_\Omega \frac{\partial \psi_n}{\partial x_j}\frac{\partial f^I_{nj}}{\partial U_m}\frac{\partial U_m}{\partial b_i} d\Omega \\
&= \int_S \psi_n \hat{n}_j \frac{\partial f^I_{nj}}{\partial b_i} dS - \int_\Omega \frac{\partial \psi_n}{\partial x_j} A_{nmj} \frac{\partial U_m}{\partial b_i} d\Omega
\end{aligned} \tag{4.8}$$

since partial derivatives can interchange. The resulting volume integral will be included in the expression of FAEs, while the boundary integral will further be expanded to isolate ABCs and SD contributions.

4.2.2 Differentiation of the Diffusive Terms

The diffusive terms in the momentum and turbulence model equations arise due to both the molecular and eddy viscosity. Following the same procedure, these can be expanded as

$$\begin{aligned}
\mathcal{T}^D &:= -\int_\Omega \psi_n \frac{\partial}{\partial b_i}\left(\frac{\partial f^V_{nj}}{\partial x_j}\right) d\Omega = -\int_S \psi_n \hat{n}_j \frac{\partial f^V_{nj}}{\partial b_i} dS + \int_\Omega \frac{\partial \psi_n}{\partial x_j}\frac{\partial f^V_{nj}}{\partial b_i} d\Omega \\
&= -\int_S \psi_n \hat{n}_j \frac{\partial f^V_{nj}}{\partial b_i} dS + \int_\Omega \underbrace{\frac{\partial \psi_{k+1}}{\partial x_j}\frac{\partial \tau_{kj}}{\partial b_i}}_{\mathcal{T}^{SS}} d\Omega \\
&+ \int_\Omega \underbrace{\frac{\partial \psi_6}{\partial x_j}\frac{\partial}{\partial b_i}\left(\tilde{\mu}^{(k)}\frac{\partial k}{\partial x_j}\right) + \frac{\partial \psi_7}{\partial x_j}\frac{\partial}{\partial b_i}\left(\tilde{\mu}^{(\varepsilon)}\frac{\partial \varepsilon}{\partial x_j}\right)}_{\mathcal{T}^{TD}} d\Omega
\end{aligned} \tag{4.9}$$

where $\tilde{\mu}^{(k)} = \left(\mu_m + \frac{\mu_{m,t}}{\sigma_k}\right)$ and $\tilde{\mu}^{(\varepsilon)} = \left(\mu_m + \frac{\mu_{m,t}}{\sigma_\varepsilon}\right)$.

Term $\mathcal{T}^{SS}$ is developed by expanding the stress tensor and applying the divergence theorem once more as

$$\mathcal{T}^{SS} := \int_\Omega \frac{\partial \psi_{k+1}}{\partial x_j}\frac{\partial \tau_{kj}}{\partial b_i} d\Omega = \int_\Omega \frac{\partial \psi_{k+1}}{\partial x_j}\frac{\partial}{\partial b_i}\left[(\mu_m + \mu_{m,t})\left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3}\frac{\partial u_\ell}{\partial x_\ell}\delta^j_k\right)\right] d\Omega$$

$$
\begin{aligned}
&= \int_\Omega \frac{\partial \psi_{k+1}}{\partial x_j} \left(\mu_m + \mu_{m,t}\right) \frac{\partial}{\partial b_i} \left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3} \frac{\partial u_\ell}{\partial x_\ell} \delta_k^j \right) d\Omega \\
&+ \int_\Omega \frac{\partial \psi_{k+1}}{\partial x_j} \left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3} \frac{\partial u_\ell}{\partial x_\ell} \delta_k^j \right) \frac{\partial}{\partial b_i} \left(\mu_m + \mu_{m,t}\right) d\Omega \\
&= \int_S \tau_{kj}^{\psi} \hat{n}_k \frac{\partial u_j}{\partial b_i} dS - \int_\Omega \frac{\partial \tau_{kj}^{\psi}}{\partial x_k} \frac{\partial u_j}{\partial b_i} d\Omega \\
&+ \int_\Omega \frac{\partial \psi_{k+1}}{\partial x_j} \frac{\tau_{kj}}{\mu_m + \mu_{m,t}} \left[\frac{\partial \mu_m}{\partial U_p} \frac{\partial U_p}{\partial b_i} + \frac{\partial \mu_{m,t}}{\partial U_p} \frac{\partial U_p}{\partial b_i} \right] d\Omega
\end{aligned} \tag{4.10}
$$

with $\tau_{kj}^{\psi} = \left(\mu_m + \mu_{m,t}\right) \left(\frac{\partial \psi_{k+1}}{\partial x_j} + \frac{\partial \psi_{j+1}}{\partial x_k} - \frac{2}{3} \frac{\partial \psi_{\ell+1}}{\partial x_\ell} \delta_k^j \right)$. The last integral in Eq. (4.10) includes terms that are introduced due to variations in the mixture molecular (Eq. (3.63)) and eddy (Eq. (3.76)) viscosity and contribute to the FAEs for the adjoint liquid volume fraction and turbulence variables. The turbulence model diffusion terms are similarly expanded resulting in

$$
\begin{aligned}
\mathcal{T}^{TD} &\coloneqq \int_\Omega \frac{\partial \psi_6}{\partial x_j} \frac{\partial}{\partial b_i} \left(\tilde{\mu}^{(k)} \frac{\partial k}{\partial x_j} \right) + \frac{\partial \psi_7}{\partial x_j} \frac{\partial}{\partial b_i} \left(\tilde{\mu}^{(\varepsilon)} \frac{\partial \varepsilon}{\partial x_j} \right) d\Omega \\
&= \int_\Omega \frac{\partial \psi_6}{\partial x_j} \left(\frac{\partial \tilde{\mu}^{(k)}}{\partial b_i} \frac{\partial k}{\partial x_j} + \tilde{\mu}^{(k)} \frac{\partial}{\partial b_i} \left(\frac{\partial k}{\partial x_j} \right) \right) d\Omega \\
&+ \int_\Omega \frac{\partial \psi_7}{\partial x_j} \left(\frac{\partial \tilde{\mu}^{(\varepsilon)}}{\partial b_i} \frac{\partial \varepsilon}{\partial x_j} + \tilde{\mu}^{(\varepsilon)} \frac{\partial}{\partial b_i} \left(\frac{\partial \varepsilon}{\partial x_j} \right) \right) d\Omega \\
&= \int_\Omega \frac{\partial \psi_6}{\partial x_j} \tilde{\mu}^{(k)} \frac{\partial}{\partial b_i} \left(\frac{\partial k}{\partial x_j} \right) + \frac{\partial \psi_7}{\partial x_j} \tilde{\mu}^{(\varepsilon)} \frac{\partial}{\partial b_i} \left(\frac{\partial \varepsilon}{\partial x_j} \right) d\Omega \\
&+ \int_\Omega \frac{\partial \psi_6}{\partial x_j} \frac{\partial k}{\partial x_j} \frac{\partial \tilde{\mu}^{(k)}}{\partial b_i} + \frac{\partial \psi_7}{\partial x_j} \frac{\partial \varepsilon}{\partial x_j} \frac{\partial \tilde{\mu}^{(\varepsilon)}}{\partial b_i} d\Omega \\
&= \int_S \left[\tilde{\mu}^{(k)} \frac{\partial \psi_6}{\partial x_j} \hat{n}_j \frac{\partial k}{\partial U_p} \frac{\partial U_p}{\partial b_i} + \tilde{\mu}^{(\varepsilon)} \frac{\partial \psi_7}{\partial x_j} \hat{n}_j \frac{\partial \varepsilon}{\partial U_p} \frac{\partial U_p}{\partial b_i} \right] dS \\
&- \int_\Omega \frac{\partial}{\partial x_j} \left(\tilde{\mu}^{(k)} \frac{\partial \psi_6}{\partial x_j} \right) \frac{\partial k}{\partial U_p} \frac{\partial U_p}{\partial b_i} + \frac{\partial}{\partial x_j} \left(\tilde{\mu}^{(\varepsilon)} \frac{\partial \psi_7}{\partial x_j} \right) \frac{\partial \varepsilon}{\partial U_p} \frac{\partial U_p}{\partial b_i} d\Omega \\
&+ \int_\Omega \left(\frac{\partial \psi_6}{\partial x_j} \frac{\partial k}{\partial x_j} + \frac{\partial \psi_7}{\partial x_j} \frac{\partial \varepsilon}{\partial x_j} \right) \frac{\partial \mu_m}{\partial U_p} \frac{\partial U_p}{\partial b_i} d\Omega \\
&+ \int_\Omega \left(\frac{1}{\sigma_k} \frac{\partial \psi_6}{\partial x_j} \frac{\partial k}{\partial x_j} + \frac{1}{\sigma_\varepsilon} \frac{\partial \psi_7}{\partial x_j} \frac{\partial \varepsilon}{\partial x_j} \right) \frac{\partial \mu_{m,t}}{\partial U_p} \frac{\partial U_p}{\partial b_i} d\Omega
\end{aligned} \tag{4.11}
$$

4.2.3 Differentiation of Source Terms

The source terms $\mathcal{T}^S$ is split into contributions associated with the interphase mass transfer due to cavitation (Eq. (3.73)) and the turbulence model source terms (Eq. (3.54)), namely

$$
\mathcal{T}^S \coloneqq - \int_\Omega \psi_n \frac{\partial S_n}{\partial b_i} d\Omega = - \int_\Omega \psi_n \left(\frac{\partial S_n^{CAV}}{\partial b_i} + \frac{\partial S_n^{KE}}{\partial b_i} \right) d\Omega \tag{4.12}
$$

Further expanding the cavitation source term leads to

$$-\int_\Omega \psi_n \frac{\partial S_n^{CAV}}{\partial b_i} d\Omega = -\int_\Omega \mathcal{B}_1 \frac{\partial \dot{m}}{\partial U_p}\frac{\partial U_p}{\partial b_i} d\Omega \tag{4.13}$$

where $\mathcal{B}_1 = \frac{\psi_1}{\varrho_l - \varrho_v} + \frac{\psi_5}{\varrho_l}$ was used. The differentiation of the interphase mass transfer term $\dot{m}$ w.r.t. the flow variables yields

$$\frac{\partial \dot{m}}{\partial \boldsymbol{U}} = \begin{bmatrix} \mathcal{H} & 0 & 0 & 0 & \mathcal{G} & 0 & 0 \end{bmatrix} \tag{4.14}$$

with

$$\mathcal{H} := \frac{\partial \dot{m}}{\partial p} = \begin{cases} \frac{C_{dest}\varrho_v}{\frac{1}{2}\varrho_l U_\infty^2 t_\infty} a_l & , \quad p - p_v < 0 \\ 0, & otherwise \end{cases} \tag{4.15}$$

$$\mathcal{G} := \frac{\partial \dot{m}}{\partial a_l} = \begin{cases} \frac{C_{prod}\varrho_v}{t_\infty}\, a_l\,(2 - 3a_l) + \frac{C_{dest}\varrho_v}{\frac{1}{2}\varrho_l U_\infty^2 t_\infty}\,(p - p_v)\,, & p - p_v < 0 \\ \frac{C_{prod}\varrho_v}{t_\infty}\, a_l\,(2 - 3a_l)\,, & otherwise \end{cases} \tag{4.16}$$

The differentiation of the turbulence model source term w.r.t. the design variables takes place by first splitting it into source and sink components, i.e.

$$\begin{aligned} -\int_\Omega \psi_n \frac{\partial S_n^{KE}}{\partial b_i} d\Omega &= -\int_\Omega \boldsymbol{\psi}^T \frac{\partial}{\partial b_i}\left(\begin{bmatrix} \mathbf{0} \\ 1 \\ C_{1\varepsilon}\frac{\rho_m \varepsilon}{\rho_m k} \end{bmatrix} \mathscr{P} - \begin{bmatrix} \mathbf{0} \\ \rho_m \varepsilon \\ C_{2\varepsilon}\frac{(\rho_m \varepsilon)^2}{\rho_m k} \end{bmatrix} \right) d\Omega \\ &= -\int_\Omega \boldsymbol{\psi}^T \left(\frac{\partial}{\partial b_i}\begin{bmatrix} \mathbf{0} \\ 1 \\ C_{1\varepsilon}\frac{\rho_m \varepsilon}{\rho_m k} \end{bmatrix} \mathscr{P} + \begin{bmatrix} \mathbf{0} \\ 1 \\ C_{1\varepsilon}\frac{\rho_m \varepsilon}{\rho_m k} \end{bmatrix} \frac{\partial \mathscr{P}}{\partial b_i} - \frac{\partial}{\partial b_i}\begin{bmatrix} \mathbf{0} \\ 1 \\ C_{2\varepsilon}\frac{(\rho_m \varepsilon)^2}{\rho_m k} \end{bmatrix} \right) d\Omega \end{aligned} \tag{4.17}$$

The differentiation of each term appearing in Eq. (4.17) w.r.t. the design variable is presented separately. Starting from the turbulence model production of turbulent kinetic energy (Eq. (3.40)), its differentiation results in

$$\begin{aligned} \frac{\partial \mathscr{P}}{\partial b_i} &= \frac{\partial \mu_{m,t}}{\partial b_i}\frac{\mathscr{P}}{\mu_{m,t}} + \mu_{m,t}\frac{\partial}{\partial b_i}\left(\frac{\mathscr{P}}{\mu_{m,t}}\right) \\ &= \frac{\mathscr{P}}{\mu_{m,t}}\frac{\partial \mu_{m,t}}{\partial U_p}\frac{\partial U_p}{\partial b_i} + \mu_{m,t}\left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k}\right)\left[\frac{\partial}{\partial x_j}\left(\frac{\partial u_k}{\partial b_i}\right) + \frac{\partial}{\partial x_k}\left(\frac{\partial u_j}{\partial b_i}\right)\right] \\ &\quad - \mu_{m,t}\frac{4}{3}\frac{\partial u_\ell}{\partial x \ell}\delta_j^k \frac{\partial}{\partial x_j}\left(\frac{\partial u_k}{\partial b_i}\right) \\ &= \frac{\mathscr{P}}{\mu_{m,t}}\frac{\partial \mu_{m,t}}{\partial U_p}\frac{\partial U_p}{\partial b_i} + 2\mu_{m,t}\left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3}\frac{\partial u_\ell}{\partial x \ell}\delta_j^k\right)\frac{\partial}{\partial x_j}\left(\frac{\partial u_k}{\partial b_i}\right) \end{aligned} \tag{4.18}$$

where $\left(\frac{\partial u_k}{\partial x_j}+\frac{\partial u_j}{\partial x_k}\right)\frac{\partial u_k}{\partial x_j}=\frac{1}{2}\left(\frac{\partial u_k}{\partial x_j}+\frac{\partial u_j}{\partial x_k}\right)^2$. For the remaining terms, these are

$$\frac{\partial}{\partial b_i}\left(C_{1\varepsilon}\frac{\rho_m\varepsilon}{\rho_m k}\right)=\frac{C_{1\varepsilon}}{\rho_m k}\left(\frac{\partial \rho_m\varepsilon}{\partial b_i}-\frac{\rho_m\varepsilon}{\rho_m k}\frac{\partial \rho_m k}{\partial b_i}\right) \tag{4.19}$$

$$\frac{\partial}{\partial b_i}\left(C_{2\varepsilon}\frac{(\rho_m\varepsilon)^2}{\rho_m k}\right)=C_{2\varepsilon}\frac{\rho_m\varepsilon}{\rho_m k}\left(2\frac{\partial \rho_m\varepsilon}{\partial b_i}-\frac{\rho_m\varepsilon}{\rho_m k}\frac{\partial \rho_m k}{\partial b_i}\right) \tag{4.20}$$

$$\frac{\partial \mu_{m,t}}{\partial b_i}=C_\mu\left(2\frac{\rho_m k}{\rho_m\varepsilon}\frac{\partial \rho_m k}{\partial b_i}-\left(\frac{\rho_m k}{\rho_m\varepsilon}\right)^2\frac{\partial \rho_m\varepsilon}{\partial b_i}\right)=2\frac{\mu_{m,t}}{\rho_m k}\frac{\partial \rho_m k}{\partial b_i}-\frac{\mu_{m,t}}{\rho_m\varepsilon}\frac{\partial \rho_m\varepsilon}{\partial b_i} \tag{4.21}$$

Additionally, an auxiliary variable is defined

$$\mathcal{B}_2 \coloneqq \boldsymbol{\psi}^T\begin{bmatrix}\mathbf{0}\\1\\C_{1\varepsilon}\frac{\rho_m\varepsilon}{\rho_m k}\end{bmatrix}=\psi_6+C_{1\varepsilon}\frac{\rho_m\varepsilon}{\rho_m k}\psi_7 \tag{4.22}$$

Taking into account Eqs. (4.13)-(4.22), the differentiation of the source term $\mathcal{T}^S$ w.r.t. the design variables results in

$$\begin{aligned}\mathcal{T}^S=&-\int_\Omega \mathcal{B}_1\mathcal{H}\frac{\partial p}{\partial b_i}+\mathcal{B}_1\mathcal{G}\frac{\partial a_l}{\partial b_i}d\Omega\\&\underbrace{-\int_\Omega\left[2\mu_{m,t}\mathcal{B}_2\left(\frac{\partial u_k}{\partial x_j}+\frac{\partial u_j}{\partial x_k}\right)-\frac{2}{3}\frac{\partial u_\ell}{\partial x\ell}\delta_j^k\right]\frac{\partial}{\partial x_j}\left(\frac{\partial u_k}{\partial b_i}\right)d\Omega}_{\mathcal{T}^P}\\&-\int_\Omega\frac{\mathscr{P}}{\mu_{m,t}}\left[\mathcal{B}_2\frac{\partial \mu_{m,t}}{\partial U_p}\frac{\partial U_p}{\partial b_i}+\mu_{m,t}\psi_7\frac{C_{1\varepsilon}}{\rho_m k}\left(\frac{\partial \rho_m\varepsilon}{\partial b_i}-\frac{\rho_m\varepsilon}{\rho_m k}\frac{\partial \rho_m k}{\partial b_i}\right)\right]d\Omega\\&+\int_\Omega\psi_6\frac{\partial \rho_m\varepsilon}{\partial b_i}+\psi_7 C_{2\varepsilon}\frac{\rho_m\varepsilon}{\rho_m k}\left(2\frac{\partial \rho_m\varepsilon}{\partial b_i}-\frac{\rho_m\varepsilon}{\rho_m k}\frac{\partial \rho_m k}{\partial b_i}\right)d\Omega\end{aligned} \tag{4.23}$$

What remains is to further expand term $\mathcal{T}^P$ that contains $\frac{\partial}{\partial x_j}\left(\frac{\partial u_k}{\partial b_i}\right)$ by applying the divergence theorem as

$$\begin{aligned}\mathcal{T}^P=&-\int_S 2\mu_{m,t}\mathcal{B}_2\left(\frac{\partial u_k}{\partial x_j}+\frac{\partial u_j}{\partial x_k}-\frac{2}{3}\frac{\partial u_\ell}{\partial x\ell}\delta_j^k\right)\hat{n}_j\frac{\partial u_k}{\partial b_i}dS\\&+\int_\Omega\frac{\partial}{\partial x_j}\left[2\mu_{m,t}\mathcal{B}_2\left(\frac{\partial u_k}{\partial x_j}+\frac{\partial u_j}{\partial x_k}-\frac{2}{3}\frac{\partial u_\ell}{\partial x\ell}\delta_j^k\right)\right]\frac{\partial u_k}{\partial b_i}d\Omega\end{aligned} \tag{4.24}$$

4.2.4 Field Adjoint Equations

The FAEs are derived by collecting all volume integrals containing derivatives w.r.t. the flow variables, i.e. $\frac{\partial \boldsymbol{U}}{\partial b_i}$ and eliminating their multipliers. The new system of PDEs should have similar properties to the governing equations and are solved following a similar method to the one presented in Section 3.1.4. Since the preconditioned adjoint equations are derived,

appropriate pseudo–time derivatives are introduced to obtain an eigensystem with the opposite eigenvalues of the governing equations. This is achieved using the transposed preconditioner of the governing equations.

The FAEs, obtained by collecting terms from Eqs. (4.2), (4.8), (4.10), (4.11) and (4.23), read

$$\Gamma^{T}_{ni}\frac{\partial\psi_i}{\partial\tau}-\frac{\partial\psi_n}{\partial x_j}A_{nmj}-\frac{\partial h^{\psi}_{nj}}{\partial x_j}-S^{\psi}_{n}-Z^{\psi}_{n}+\frac{\partial j_{\Omega}}{\partial U_n}=0 \tag{4.25}$$

with

$$\boldsymbol{h}^{\psi}_{j}=\begin{bmatrix}0\\ \tau^{\psi}_{1j}-2\mu_{m,t}\mathcal{B}_2\left(\frac{\partial u_1}{\partial x_j}+\frac{\partial u_j}{\partial x_1}-\frac{2}{3}\frac{\partial u_\ell}{\partial x\ell}\delta^1_j\right)\\ \tau^{\psi}_{2j}-2\mu_{m,t}\mathcal{B}_2\left(\frac{\partial u_2}{\partial x_j}+\frac{\partial u_j}{\partial x_2}-\frac{2}{3}\frac{\partial u_\ell}{\partial x\ell}\delta^2_j\right)\\ \tau^{\psi}_{3j}-2\mu_{m,t}\mathcal{B}_2\left(\frac{\partial u_3}{\partial x_j}+\frac{\partial u_j}{\partial x_3}-\frac{2}{3}\frac{\partial u_\ell}{\partial x\ell}\delta^3_j\right)\\ -\frac{\Delta\varrho}{\rho_m}\left(k\tilde{\mu}^{(k)}\frac{\partial\psi_6}{\partial x_j}+\varepsilon\tilde{\mu}^{(\varepsilon)}\frac{\partial\psi_7}{\partial x_j}\right)\\ \frac{\tilde{\mu}^{(k)}}{\varrho_m}\frac{\partial\psi_6}{\partial x_j}\\ \frac{\tilde{\mu}^{(\varepsilon)}}{\varrho_m}\frac{\partial\psi_7}{\partial x_j}\end{bmatrix},\boldsymbol{S}^{\psi}=\begin{bmatrix}\mathcal{B}_1\mathcal{H}\\ 0\\ 0\\ 0\\ \mathcal{B}_1\mathcal{G}\\ \frac{2}{\rho_m k}\mathscr{P}\psi_6+\frac{\rho_m\varepsilon}{(\rho_m k)^2}\left(C_{1\varepsilon}\mathscr{P}+C_{2\varepsilon}\rho_m\varepsilon\right)\psi_7\\ -\frac{1}{\rho_m\varepsilon}\left(\mathscr{P}+\rho_m\varepsilon\right)\psi_6-2C_{2\varepsilon}\frac{\rho_m\varepsilon}{\rho_m k}\psi_7\end{bmatrix}$$

$$\boldsymbol{Z}^{\psi}=\begin{bmatrix}0\\ 0\\ 0\\ 0\\ \Delta\mu\left(\frac{\tau_{kj}}{\mu_m+\mu_{m,t}}\frac{\partial\psi_{k+1}}{\partial x_j}+\frac{\partial\psi_6}{\partial x_j}\frac{\partial k}{\partial x_j}+\frac{\partial\psi_7}{\partial x_j}\frac{\partial\varepsilon}{\partial x_j}\right)\\ 2\frac{\mu_{m,t}}{\rho_m k}\left(\frac{\tau_{kj}}{\mu_m+\mu_{m,t}}\frac{\partial\psi_{k+1}}{\partial x_j}+\frac{1}{\sigma_k}\frac{\partial\psi_6}{\partial x_j}\frac{\partial k}{\partial x_j}+\frac{1}{\sigma_\varepsilon}\frac{\partial\psi_7}{\partial x_j}\frac{\partial\varepsilon}{\partial x_j}\right)\\ -\frac{\mu_{m,t}}{\rho_m\varepsilon}\left(\frac{\tau_{kj}}{\mu_m+\mu_{m,t}}\frac{\partial\psi_{k+1}}{\partial x_j}+\frac{1}{\sigma_k}\frac{\partial\psi_6}{\partial x_j}\frac{\partial k}{\partial x_j}+\frac{1}{\sigma_\varepsilon}\frac{\partial\psi_7}{\partial x_j}\frac{\partial\varepsilon}{\partial x_j}\right)\end{bmatrix} \tag{4.26}$$

where vector $\boldsymbol{h}^{\psi}$ contains the adjoint viscous flux vector terms originating from the differentiation of the diffusive terms and turbulence model production, vector $\boldsymbol{S}^{\psi}=\boldsymbol{S}^{\psi}(\boldsymbol{\psi})$ contains source terms due to the cavitation and turbulence model source vector, while vector $\boldsymbol{Z}^{\psi}=\boldsymbol{Z}^{\psi}\left(\frac{\partial\boldsymbol{\psi}}{\partial\mathbf{x}}\right)$ contains source terms resulting from the differentiation of the mixture molecular and eddy viscosity.

4.2.5 Adjoint Boundary Conditions

The remaining terms in the objective function gradient expression are boundary integrals. Aiming at obtaining a SDs expression that does not require the computation of flow vari-

ables' derivatives along the boundaries, terms containing $\frac{\partial \boldsymbol{U}}{\partial b_i}$, and terms that can be expressed w.r.t. the said derivative, are further expanded. To do so, partial derivatives of the flow variables are transformed to total derivatives of the flow variables and geometric quantities. Total derivatives of the flow variables are first collected and, then, have their multiplier set to zero, by also taking into account the boundary condition of the problem; this leads to the ABCs. The remaining surface integrals (SI) are summarized as

$$\begin{aligned}
\frac{\delta L}{\delta b_i} = & \int_S \left[\psi_n \hat{n}_j \frac{\partial f^I_{nj}}{\partial b_i} - \psi_n \hat{n}_j \frac{\partial f^V_{nj}}{\partial b_i} + \tau^{\psi}_{kj} \hat{n}_k \frac{\partial u_j}{\partial b_i} \right] dS \\
& - \int_S 2\mu_{m,t} \mathcal{B}_2 \left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3} \frac{\partial u_\ell}{\partial x \ell} \delta^k_j \right) \hat{n}_j \frac{\partial u_k}{\partial b_i} dS \\
& + \int_S \left[\tilde{\mu}^{(k)} \frac{\partial \psi_6}{\partial x_j} \hat{n}_j \frac{\partial k}{\partial U_p} \frac{\partial U_p}{\partial b_i} + \tilde{\mu}^{(\varepsilon)} \frac{\partial \psi_7}{\partial x_j} \hat{n}_j \frac{\partial \varepsilon}{\partial U_p} \frac{\partial U_p}{\partial b_i} \right] dS \\
& + \int_S \psi_n \mathscr{R}_n \hat{n}_k \frac{\delta x_k}{\delta b_i} dS + \frac{\delta J}{\delta b_i}
\end{aligned} \tag{4.27}$$

Substitution of the viscous flux vector expression and some re–arrangement results in

$$\begin{aligned}
\frac{\delta L}{\delta b_i} = & \int_S \underbrace{\psi_n \hat{n}_j \frac{\partial f^I_{nj}}{\partial b_i}}_{SI_1} \underbrace{- \psi_{k+1} \hat{n}_j \frac{\partial \tau_{jk}}{\partial b_i}}_{SI_2} + \underbrace{\left(\tau^{\psi}_{kj} - 2\mu_{m,t} \mathcal{B}_2 \left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3} \frac{\partial u_\ell}{\partial x \ell} \delta^k_j \right) \right) \hat{n}_k \frac{\partial u_j}{\partial b_i}}_{SI_3} dS \\
& - \int_S \underbrace{\psi_6 \hat{n}_j \left(\frac{\partial \tilde{\mu}^{(k)}}{\partial b_i} \frac{\partial k}{\partial x_j} + \tilde{\mu}^{(k)} \frac{\partial}{\partial b_i} \left(\frac{\partial k}{\partial x_j} \right) \right)}_{SI_4} + \underbrace{\psi_7 \hat{n}_j \left(\frac{\partial \tilde{\mu}^{(\varepsilon)}}{\partial b_i} \frac{\partial \varepsilon}{\partial x_j} + \tilde{\mu}^{(\varepsilon)} \frac{\partial}{\partial b_i} \left(\frac{\partial \varepsilon}{\partial x_j} \right) \right)}_{SI_5} dS \\
& + \int_S \underbrace{\frac{\tilde{\mu}^{(k)}}{\varrho_m} \frac{\partial \psi_6}{\partial x_j} \hat{n}_j \frac{\partial \rho_m k}{\partial b_i}}_{SI_6} dS + \int_S \underbrace{\frac{\tilde{\mu}^{(\varepsilon)}}{\varrho_m} \frac{\partial \psi_7}{\partial x_j} \hat{n}_j \frac{\partial \rho_m \varepsilon}{\partial b_i}}_{SI_7} dS \\
& - \int_S \underbrace{\Delta \varrho \left[k \tilde{\mu}^{(k)} \frac{\partial \psi_6}{\partial x_j} + \varepsilon \tilde{\mu}^{(\varepsilon)} \frac{\partial \psi_7}{\partial x_j} \right] \hat{n}_j \frac{\partial a_l}{\partial b_i}}_{SI_8} dS + \int_S \psi_n \mathscr{R}_n \hat{n}_k \frac{\delta x_k}{\delta b_i} dS + \frac{\delta J}{\delta b_i}
\end{aligned} \tag{4.28}$$

A term–by–term analysis to extract the contributions to the ABCs and the SDs from the surface integrals follows.

The inviscid boundary term SI_1 is re–written as

$$\begin{aligned}
SI_1 \coloneqq & \int_S \psi_n \hat{n}_j \frac{\partial f^I_{nj}}{\partial b_i} dS = \int_S \psi_n \hat{n}_j \left(\frac{\delta f^I_{nj}}{\delta b_i} - \frac{\partial f^I_{nj}}{\partial x_l} \frac{\delta x_l}{\delta b_i} \right) dS \\
= & \int_S \psi_n \left(\frac{\delta \left(f^I_{nj} \hat{n}_j \right)}{\delta b_i} - f^I_{nj} \frac{\delta \hat{n}_j}{\delta b_i} - \hat{n}_j \frac{\partial f^I_{nj}}{\partial x_l} \frac{\delta x_l}{\delta b_i} \right) dS
\end{aligned}$$

$$= \underbrace{\int_S \psi_n \frac{\delta\left(f^I_{nj}\hat{n}_j\right)}{\delta b_i} dS}_{ABCs} \underbrace{- \int_S \psi_n f^I_{nj} \frac{\delta \hat{n}_j}{\delta b_i} dS}_{SDs} \underbrace{- \int_S \psi_n A_{npj} \hat{n}_j \frac{\partial U_p}{\partial x_l} \frac{\delta x_l}{\delta b_i} dS}_{SDs} \tag{4.29}$$

Similarly, the stress tensor term SI_2 is expanded as follows

$$\begin{aligned} SI_2 &\coloneqq -\int_S \psi_{k+1} \hat{n}_j \frac{\partial \tau_{kj}}{\partial b_i} dS \\ &= \underbrace{-\int_S \psi_{k+1} \frac{\delta\left(\tau_{kj}\hat{n}_j\right)}{\delta b_i} dS}_{AEBC+SDs} \underbrace{+ \int_S \psi_{k+1} \tau_{kj} \frac{\delta \hat{n}_j}{\delta b_i} dS}_{SDs} \underbrace{+ \int_S \psi_{k+1} \hat{n}_j \frac{\partial \tau_{kj}}{\partial x_l} \frac{\delta x_l}{\delta b_i} dS}_{SDs} \end{aligned} \tag{4.30}$$

Term SI_3 is developed as follows

$$\begin{aligned} SI_3 &\coloneqq \int_S \left(\tau^{\psi}_{kj} - 2\mu_{m,t}\mathcal{B}_2\left(\frac{\partial u_k}{\partial x_j} + \frac{\partial u_j}{\partial x_k} - \frac{2}{3}\frac{\partial u_\ell}{\partial x \ell}\delta^k_j\right)\right) \hat{n}_k \frac{\partial u_j}{\partial b_i} dS = \int_S \mathcal{T}^{\psi SS}_j \frac{\partial u_j}{\partial b_i} dS \\ &= \underbrace{\int_S \mathcal{T}^{\psi SS}_j \frac{\delta u_j}{\delta b_i} dS}_{ABCs} \underbrace{- \int_S \mathcal{T}^{\psi SS}_j \frac{\partial u_j}{\partial x_l} \frac{\delta x_l}{\delta b_i} dS}_{SDs} \end{aligned} \tag{4.31}$$

The differentiation of the diffusion term in the k–equation, SI_4, yields

$$\begin{aligned} SI_4{}^* &\coloneqq -\int_S \psi_6 \hat{n}_j \left(\frac{\partial \tilde{\mu}^{(k)}}{\partial b_i}\frac{\partial k}{\partial x_j} + \tilde{\mu}^{(k)}\frac{\partial}{\partial b_i}\left(\frac{\partial k}{\partial x_j}\right)\right) dS \\ &= \underbrace{-\int_S \psi_6 \hat{n}_j \frac{\partial k}{\partial x_j}\frac{\delta \tilde{\mu}^{(k)}}{\delta b_i} dS}_{ABCs} \underbrace{+ \int_S \psi_6 \hat{n}_j \frac{\partial k}{\partial x_j}\frac{\partial \tilde{\mu}^{(k)}}{\partial x_l}\frac{\delta x_l}{\delta b_i} dS}_{SDs} \\ &\quad \underbrace{-\int_S \psi_6 \hat{n}_j \tilde{\mu}^{(k)} \frac{\delta}{\delta b_i}\left(\frac{\partial k}{\partial x_j}\right) dS}_{ABCs} \underbrace{+ \int_S \psi_6 \hat{n}_j \tilde{\mu}^{(k)} \frac{\partial}{\partial x_l}\left(\frac{\partial k}{\partial x_j}\right)\frac{\delta x_l}{\delta b_i} dS}_{SDs} \end{aligned} \tag{4.32}$$

Following the same procedure for the ε–equation, SI_5 becomes

$$\begin{aligned} SI_5{}^* &\coloneqq -\int_S \psi_7 \hat{n}_j \left(\frac{\partial \tilde{\mu}^{(\varepsilon)}}{\partial b_i}\frac{\partial \varepsilon}{\partial x_j} + \tilde{\mu}^{(\varepsilon)}\frac{\partial}{\partial b_i}\left(\frac{\partial \varepsilon}{\partial x_j}\right)\right) dS \\ &= \underbrace{-\int_S \psi_\varepsilon \hat{n}_j \frac{\partial k}{\partial x_j}\frac{\delta \tilde{\mu}^{(\varepsilon)}}{\delta b_i} dS}_{ABCs} \underbrace{+ \int_S \psi_7 \hat{n}_j \frac{\partial \varepsilon}{\partial x_j}\frac{\partial \tilde{\mu}^{(\varepsilon)}}{\partial x_l}\frac{\delta x_l}{\delta b_i} dS}_{SDs} \\ &\quad \underbrace{-\int_S \psi_7 \hat{n}_j \tilde{\mu}^{(\varepsilon)} \frac{\delta}{\delta b_i}\left(\frac{\partial \varepsilon}{\partial x_j}\right) dS}_{ABCs} \underbrace{+ \int_S \psi_7 \hat{n}_j \tilde{\mu}^{(\varepsilon)} \frac{\partial}{\partial x_l}\left(\frac{\partial \varepsilon}{\partial x_j}\right)\frac{\delta x_l}{\delta b_i} dS}_{SDs} \end{aligned} \tag{4.33}$$

Some more terms' expansion follow:

$$\begin{aligned} SI_6 &:= \int_S \frac{\tilde{\mu}^{(k)}}{\varrho_m}\frac{\partial \psi_6}{\partial x_j}\hat{n}_j\frac{\partial \rho_m k}{\partial b_i}dS \\ &= \underbrace{\int_S \frac{\tilde{\mu}^{(k)}}{\varrho_m}\frac{\partial \psi_6}{\partial x_j}\hat{n}_j\frac{\delta \rho_m k}{\delta b_i}dS}_{ABCs} \underbrace{- \int_S \frac{\tilde{\mu}^{(k)}}{\varrho_m}\frac{\partial \psi_6}{\partial x_j}\hat{n}_j\frac{\partial \rho_m k}{\partial x_l}\frac{\delta x_l}{\delta b_i}dS}_{SDs} \end{aligned} \tag{4.34}$$

$$\begin{aligned} SI_7 &:= \int_S \frac{\tilde{\mu}^{(\varepsilon)}}{\varrho_m}\frac{\partial \psi_7}{\partial x_j}\hat{n}_j\frac{\partial \rho_m \varepsilon}{\partial b_i}dS \\ &= \underbrace{\int_S \frac{\tilde{\mu}^{(\varepsilon)}}{\varrho_m}\frac{\partial \psi_7}{\partial x_j}\hat{n}_j\frac{\delta \rho_m \varepsilon}{\delta b_i}dS}_{ABCs} \underbrace{- \int_S \frac{\tilde{\mu}^{(\varepsilon)}}{\varrho_m}\frac{\partial \psi_7}{\partial x_j}\hat{n}_j\frac{\partial \rho_m \varepsilon}{\partial x_l}\frac{\delta x_l}{\delta b_i}dS}_{SDs} \end{aligned} \tag{4.35}$$

and, finally, the last term to further be developed gives

$$\begin{aligned} SI_8 &:= -\int_S \Delta\varrho\left[k\tilde{\mu}^{(k)}\frac{\partial \psi_6}{\partial x_j} + \varepsilon\tilde{\mu}^{(\varepsilon)}\frac{\partial \psi_7}{\partial x_j}\right]\hat{n}_j\frac{\partial a_l}{\partial b_i}dS \\ &= \underbrace{-\int_S \Delta\varrho\left[k\tilde{\mu}^{(k)}\frac{\partial \psi_6}{\partial x_j} + \varepsilon\tilde{\mu}^{(\varepsilon)}\frac{\partial \psi_7}{\partial x_j}\right]\hat{n}_j\frac{\delta a_l}{\delta b_i}dS}_{ABCs} \\ &\underbrace{+\int_S \Delta\varrho\left[k\tilde{\mu}^{(k)}\frac{\partial \psi_6}{\partial x_j} + \varepsilon\tilde{\mu}^{(\varepsilon)}\frac{\partial \psi_7}{\partial x_j}\right]\hat{n}_j\frac{\partial a_l}{\partial x_l}\frac{\delta x_l}{\delta b_i}dS}_{SDs} \end{aligned} \tag{4.36}$$

In order to derive the final expressions of ABCs at each boundary, the conditions imposed on the governing equations are considered. This is done in the following sections.

4.2.5.1 Wall Boundaries – S_W

Along the wall boundaries, the no–slip boundary ($u_j = 0$) condition is imposed. As such, the ABCs term in Eq. (4.29) can be re-written as

$$\int_{S_W}\psi_n\frac{\delta\left(f^I_{nj}\hat{n}_j\right)}{\delta b_i}dS = \underbrace{\int_{S_W}\psi_{k+1}\hat{n}_k\frac{\delta p}{\delta b_i}dS}_{ABCs} + \underbrace{\int_{S_W}\psi_{k+1}p\frac{\delta \hat{n}_k}{\delta b_i}dS}_{SDs} \tag{4.37}$$

*Terms $SI_{4,5}$ can further be developed by using $\delta\tilde{\mu}^{(k,\varepsilon)} = \Delta\varrho\ \delta a_l + \frac{2}{\sigma_{(k,\varepsilon)}}\frac{\mu_{m,t}}{\rho_m k}\ \delta\rho_m k - \frac{1}{\sigma_{(k,\varepsilon)}}\frac{\mu_{m,t}}{\rho_m \varepsilon}\ \delta\rho_m\varepsilon$, $\frac{\partial k}{\partial x_j} = \frac{1}{\varrho_m}\left(\frac{\partial \rho_m k}{\partial x_j} - k\Delta\varrho\frac{\partial a_l}{\partial x_j}\right)$ and $\frac{\partial \varepsilon}{\partial x_j} = \frac{1}{\varrho_m}\left(\frac{\partial \rho_m \varepsilon}{\partial x_j} - \varepsilon\Delta\varrho\frac{\partial a_l}{\partial x_j}\right)$. However, as shown in the following sections, these terms need to be eliminated through the adjoint boundary conditions to avoid the computation of $\frac{\delta}{\delta b_i}\left(\frac{\partial \rho_m k}{\partial x_j}\right)$ and $\frac{\delta}{\delta b_i}\left(\frac{\partial \rho_m \varepsilon}{\partial x_j}\right)$.

Considering that ($k = 0$, $\varepsilon = \epsilon$), the ABCs terms in Eqs. (4.34), (4.35), and (4.36) vanish automatically, since the flow boundary conditions lead to $\frac{\delta u_j}{\delta b_i} = 0$, $\frac{\delta \rho_m k}{\delta b_i} = 0$, and $\frac{\delta \rho_m \varepsilon}{\delta b_i} = 0$.

Since the wall function technique is employed, the wall shear stress is computed using the law of the wall and, thus, the ABCs term in Eq. (4.30) needs to further be developed by differentiating the entire procedure. The development builds upon the concept of *adjoint wall functions*, initially introduced in Zymaris *et al.* [283], but is adapted to two–phase flows and the introduced CC–specific linelets, overviewed in Section 3.2.2.2. A detailed derivation is given in Appendix B. The corresponding term now becomes

$$-\int_S \psi_{k+1} \frac{\delta\left(\tau_{kj}\hat{n}_j\right)}{\delta b_i} dS = \underbrace{-\int_{S_W} \Psi_n \frac{\delta\left(\hat{n}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} dS}_{ABCs} \underbrace{-\int_{S_W} \Psi_z \frac{\delta\left(\hat{z}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} dS}_{ABCs} \tag{4.38}$$

$$\underbrace{+\int_{S_W} \tau_{kj}\hat{n}_j \left(\Psi_n \frac{\delta \hat{n}_k}{\delta b_i} + \Psi_t \frac{\delta \hat{t}_k}{\delta b_i} + \Psi_z \frac{\delta \hat{z}_k}{\delta b_i}\right) dS}_{SDs} \tag{4.39}$$

$$\underbrace{+\int_{S_W} \mathrm{u}_\tau \Psi_t \left[\mathrm{u}_\tau \Delta\varrho + 2\mathrm{u}_\tau \frac{\partial \mathrm{u}_\tau}{\partial \nu_m}\left(\Delta\mu - \nu_m \Delta\varrho\right)\right] \frac{\partial a_l^F}{\partial b_i} dS}_{ST} \tag{4.40}$$

$$\underbrace{-\int_{S_W} \mathrm{u}_\tau \Psi_t \left[\mathrm{u}_\tau \Delta\varrho + 2\mathrm{u}_\tau \frac{\partial \mathrm{u}_\tau}{\partial \nu_m}\left(\Delta\mu - \nu_m \Delta\varrho\right)\right] \frac{\partial a_l^F}{\partial x_l} \frac{\delta x_l}{\delta b_i} dS}_{SDs} \tag{4.41}$$

$$\underbrace{+\int_{S_W} \mathcal{B}_{WF}\left(\hat{t}_k + u_i^F \mathbb{M}_{ij}\mathbb{P}_{jk}\right) \frac{\partial u_k^F}{\partial b_i} dS}_{ST} \tag{4.42}$$

$$\underbrace{-\int_{S_W} \mathcal{B}_{WF}\left(\hat{t}_k + u_i^F \mathbb{M}_{ij}\mathbb{P}_{jk}\right) \frac{\partial u_k^F}{\partial x_l} \frac{\delta x_l}{\delta b_i} dS}_{SDs} \tag{4.43}$$

$$\underbrace{-\int_{S_W} \mathcal{B}_{WF} u_i^F \mathbb{M}_{ij}\mathbb{Q}_{jk} \frac{\delta \hat{n}_k}{\delta b_i} dS}_{SDs} \tag{4.44}$$

whereas, terms denoted by *ST* are included in the FAEs and act upon the Cartesian cells that encompass the forcing points as source terms. Computing these source terms introduces additional implementation complexities, especially when considering parallel computations. In detail, the forcing points can reside on different subdomains and require additional communications to obtain the associated data. Interestingly, in the adjoint method, data are in reversed order compared to the flow solution, i.e. data from the solid wall need to be transferred to the Cartesian cell to compute the associated terms (see Section 3.2.2.2). On a final note regard-

ing the code implementation, an assumption is made regarding the emerging source terms, i.e. the reconstruction process between the cell center and the forcing points is neglected. In case this assumption is not made, Eq. (3.22) must be included in the differentiation, resulting in additional source terms in the neighboring to the cell encompassing the forcing point, cells.

The remaining ABCs terms over the solid walls in Eqs. (4.2), (4.32), (4.33), (4.37), and (4.38) are eliminated by imposing [191, 255]

$$\psi_{k+1}\hat{n}_k = -\frac{\partial j_S}{\partial p} \tag{4.45}$$

$$\psi_{k+1}\hat{t}_k = \frac{\partial j_S}{\partial\left(\tau_{ij}\hat{n}_i\hat{t}_j\right)} \tag{4.46}$$

$$\psi_{k+1}\hat{z}_k = \frac{\partial j_S}{\partial\left(\tau_{ij}\hat{n}_i\hat{z}_j\right)} \tag{4.47}$$

$$\psi_6 = 0 \tag{4.48}$$

$$\psi_7 = 0 \tag{4.49}$$

4.2.5.2 Inlet & Outlet Boundaries – S_I & S_O

The ABCs at the inlet and outlet boundaries are used to compute the adjoint boundary variables based on the following system of equations:

$$\psi_i\left(A_{imk}\hat{n}_k Q_{mn}\right) + (1-\kappa)\, h^{\psi}_{nk}\hat{n}_k + \frac{\partial j_S}{\partial U_n} = 0, \qquad n = 1, ..., 7 \tag{4.50}$$

where $\boldsymbol{Q} = \frac{\partial \boldsymbol{U}^{BC}}{\partial \boldsymbol{U}}$ is the matrix containing the partial derivatives of the imposed boundary conditions w.r.t. the flow variables, and its form is given separately for each imposed boundary condition. Two different cases are considered for the inlet boundaries depending on whether the total pressure or the velocity magnitude is specified. The latter, this results in $\frac{\delta u_j}{\delta b_i} = 0$ at the inlet, automatically eliminating the ABC terms containing the adjoint viscous flux vector terms. As such, in Eq. (4.50) $\kappa = 1$ when the velocity magnitude is imposed at the inlet, whereas $\kappa = 0$ when the total pressure is imposed.

It should also be noted that the inlet and outlet boundaries are assumed to be fixed/unparameterized, therefore, all geometric derivatives are zero there, i.e. $\frac{\delta x_I}{\delta b_i} = \frac{\delta \hat{n}_k}{\delta b_i} = 0$. This results in no additional SDs terms along the inlet and outlet.

Inlet Boundaries – Specified Velocity Magnitude In case the inlet velocity magnitude is specified, the $\boldsymbol{Q}$ matrix is defined as

$$\boldsymbol{Q} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{4.51}$$

since all unknown flow variables, excluding the pressure, are fixed due to the Dirichlet condition ($\frac{\delta U_n}{\delta b_i} = 0,\ n = 2, ..., 7$). Thus, the ABC terms in Eqs. (4.31), (4.34), (4.35), and (4.36) vanish automatically. An additional condition needs to be imposed to eliminate the remaining ABCs terms in Eqs. (4.32) and (4.33), namely

$$\psi_6 = 0 \tag{4.52}$$

$$\psi_7 = 0 \tag{4.53}$$

Inlet Boundaries – Specified Total Pressure As mentioned in Section 3.1.3.2, the inlet velocity magnitude is computed based on the extrapolated static pressure, when imposing the total pressure. This results in $\frac{\delta u_j}{\delta b_i} \neq 0$, and, thus the adjoint viscous flux is included in Eq. (4.50) ($\kappa = 0$). Furthermore, this leads to dependencies of the imposed flow variables w.r.t. the static pressure $\left(\frac{\partial \boldsymbol{U}^{BC}}{\partial p}\right)$; for instance the imposed velocity at the inlet results from the dynamic pressure there. These dependencies are expressed through the $\boldsymbol{Q}$ matrix with non–zero elements arising in its first column, which after some development using Eqs. (3.29)-(3.31) and (3.51), becomes

$$\boldsymbol{Q} = \frac{1}{u_j u_j} \begin{bmatrix} u_j u_j & -u_1^S & -u_2^S & -u_3^S & 0 & -2k^S & -4\varepsilon^S \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}^T \tag{4.54}$$

The remaining ABCs terms (Eqs. (4.32) and (4.33)) are eliminated again by additionally imposing Eqs. (4.52) and (4.53), as in the previous case.

Outlet Boundaries Zero Neuman conditions are imposed on all flow variables ($\kappa = 0$) but the pressure at the outlet boundaries. This results in the elimination of the ABC terms in Eqs. (4.32) and (4.33) since $\frac{\delta}{\delta b_i}\left(\frac{\partial \rho_m k}{\partial x_k}\hat{n}_k\right) = \frac{\delta}{\delta b_i}\left(\frac{\partial \rho_m \varepsilon}{\partial x_k}\hat{n}_k\right) = 0$ and, thus, requires no additional boundary conditions. The $\boldsymbol{Q}$ matrix becomes a diagonal matrix given as

$$\boldsymbol{Q} = \text{diag}\,(0, 1, 1, 1, 1, 1, 1) \tag{4.55}$$

4.2.5.3 Farfield Boundaries

Dirichlet boundary conditions are imposed on all flow variables such that $\frac{\delta U_n}{\delta b_i} = 0$ at the farfield boundaries. This also leads to $\boldsymbol{Q} = \begin{bmatrix}\mathbf{0}\end{bmatrix}$. Spatial derivatives in the direction normal to the boundary can be neglected by assuming a uniform flow at the farfield boundaries ($\kappa = 1$), which leads to the elimination of all ABCs terms. The corresponding adjoint boundary flux is computed using only quantities extrapolated from the interior of the flow domain.

4.2.6 Expression of Sensitivity Derivatives

After satisfying the FAEs and imposing the ABCs, the SD expression contains only boundary terms that are multiplied with total derivatives of geometric quantities w.r.t. the design variables, marked with SDs in the previous formulas. The SDs are computed at the solid walls S_W where $k = \mu_{m,t} = 0, \varepsilon = \epsilon$ resulting in

$$
\begin{aligned}
\frac{\delta J}{\delta b_i} = &\int_{S_W} \left[\psi_{k+1} p - \psi_n f^I_{nk} + \psi_{j+1}\tau_{jk}\right] \frac{\delta \hat{n}_k}{\delta b_i} dS \\
&+ \int_{S_W} \left[-\psi_n A_{npj}\hat{n}_j \frac{\partial U_p}{\partial x_l} + \psi_{k+1}\hat{n}_j \frac{\partial \tau_{kj}}{\partial x_l} - \mathcal{T}^{\psi SS}_j \frac{\partial u_j}{\partial x_l}\right] \frac{\delta x_l}{\delta b_i} dS \\
&- \int_{S_W} \frac{\mu_m}{\rho_m}\left(\frac{\partial \psi_6}{\partial x_j}\hat{n}_j \frac{\partial \rho_m k}{\partial x_l} + \frac{\partial \psi_7}{\partial x_j}\hat{n}_j \frac{\partial \rho_m \varepsilon}{\partial x_l}\right) \frac{\delta x_l}{\delta b_i} dS \\
&+ \int_{S_W} \tau_{kj}\hat{n}_j \left(\Psi_n \frac{\delta \hat{n}_k}{\delta b_i} + \Psi_t \frac{\delta \hat{t}_k}{\delta b_i} + \Psi_z \frac{\delta \hat{z}_k}{\delta b_i}\right) dS - \int_{S_W} \mathcal{B}_{WF} u^F_i \mathbb{M}_{ij} \mathbb{Q}_{jk} \frac{\delta \hat{n}_k}{\delta b_i} dS \\
&- \int_{S_W} \mathcal{B}_{WF}\left(\hat{t}_k + u^F_i \mathbb{M}_{ij}\mathbb{P}_{jk}\right) \frac{\partial u^F_k}{\partial x_l}\frac{\delta x_l}{\delta b_i} dS \\
&- \int_{S_W} \mathrm{u}_\tau \Psi_t \left[\mathrm{u}_\tau \Delta\varrho + 2\mathrm{u}_\tau \frac{\partial \mathrm{u}_\tau}{\partial \nu_m}\left(\Delta\mu - \nu_m \Delta\varrho\right)\right] \frac{\partial a^F_l}{\partial x_l}\frac{\delta x_l}{\delta b_i} dS \\
&+ \int_{S_W} \left(\psi_n \mathscr{R}_n \hat{n}_k + j_\Omega \hat{n}_k + \frac{\partial j_S}{\partial x_k}\right) \frac{\delta x_k}{\delta b_i} dS + \int_{S_W} \frac{\partial j_S}{\partial \hat{n}_m} \frac{\delta(\hat{n}_m dS)}{\delta b_i}
\end{aligned}
\tag{4.56}
$$

where the first two integrals are obtained from differentiating the mean flow equations, the third one due to the turbulence model, the next four arise from the differentiation of the law of the wall (Appendix B), and the last two are obtained from the differentiation of the objective function.

4.3 Objective Functions and their Differentiation

The continuous adjoint method developed in the previous sections is employed to perform gradient–based optimization for three different objective functions. In the following sections,

the objective functions expressions are provided along with their differentiation w.r.t. the design variables.

4.3.1 Volume–averaged Total Pressure Losses

The objective function used to quantify the volume–averaged total pressure losses takes the following form

$$J_{P_t} = \int_{S^J} j_S dS = -\int_{S_I} p_t u_k \hat{n}_k \, dS - \int_{S_O} p_t u_k \hat{n}_k \, dS \tag{4.57}$$

where the negative sign used at the inlet boundary arises due to the outward facing unit normal vector. This objective function is defined at the fixed inlet(s) and outlet(s) and therefore, $\frac{\delta n_j}{\delta b_i} = \frac{\delta x_k}{\delta b_i} = 0$. J_{P_t} is differentiated for when the total pressure or velocity magnitude is defined at the inlet and results in objective function contributions only to the ABCs, Eq. (4.50). More specifically, its differentiation yields

$$\frac{\delta J_{P_t}}{\delta b_i} = \int_{S_J} \frac{\partial j_S}{\partial \boldsymbol{U}} \frac{\delta \boldsymbol{U}}{\delta b_i} = -\int_{S_I} \varphi \frac{\delta p}{\delta b_i} dS - \int_{S_O} u_k \hat{n}_k \frac{\delta p}{\delta b_i} dS - \int_{S_O} \left(u_k \hat{n}_k u_j + p_t \hat{n}_j \right) \frac{\delta u_j}{\delta b_i} dS \tag{4.58}$$

where

$$\varphi = \begin{cases} u_k \hat{n}_k & \text{if velocity magnitude is specified} \\ \frac{u_k^S \hat{n}_k}{u_j u_j} p_t & \text{if total pressure is specified} \end{cases} \tag{4.59}$$

4.3.2 Force

The objective function that calculates the force exerted on the surface of the studied body, projected on a predefined direction $\hat{\boldsymbol{r}}$, is defined as

$$J_F = \int_{S^J} j_S dS = \int_{S_W} \left(p \hat{n}_k - \tau_{kj} \hat{n}_j \right) \hat{r}_k dS \tag{4.60}$$

In case the lift force is considered, $\hat{\boldsymbol{r}}$ is the appropriately signed unit vector normal to the free–stream velocity. With reference Eq. (4.2), the differentiation of the objective function yields

$$\begin{aligned} \frac{\delta J_F}{\delta b_i} = & \int_{S_W} \hat{n}_k \hat{r}_k \frac{\delta p}{\delta b_i} dS - \int_{S_W} \hat{r}_l \hat{n}_l \frac{\delta \left(\tau_{kj} \hat{n}_k \hat{n}_j \right)}{\delta b_i} dS - \int_{S_W} \hat{r}_l \hat{t}_l \frac{\delta \left(\tau_{kj} \hat{n}_k \hat{t}_j \right)}{\delta b_i} dS \\ & + \int_{S_W} \left(p \delta_j^k - \tau_{kj} \right) \hat{r}_k \frac{\delta (\hat{n}_j dS)}{\delta b_i} + \int_{S_W} \frac{\partial j_S}{\partial x_k} \frac{\delta x_k}{\delta b_i} dS \end{aligned} \tag{4.61}$$

The first 3 terms in Eq. (4.61) are objective function contributions to the ABCs, see Eqs. (4.45)–(4.47), while the remaining two to the SDs, namely Eq. (4.56).

4.3.3 Volume of Vapour in the Fluid Domain

An objective function is used that quantifies the presence of the vapour phase throughout the entire computational domain, aiming at the minimization of the intensity of a cavitating flow. Reducing the total vapour in the computational domain results in smaller/less intense cavitation bubbles. The objective function is expressed as

$$J_V = \int_\Omega j_\Omega d\Omega = \frac{1}{2}\int_\Omega \alpha_v^2 d\Omega = \frac{1}{2}\int_\Omega \left(1-\alpha_l\right)^2 d\Omega \tag{4.62}$$

and its derivative w.r.t. the design variables

$$\frac{\delta J_V}{\delta b_i} = \int_\Omega \left(\alpha_l - 1\right)\frac{\partial \alpha_l}{\partial b_i} d\Omega + \frac{1}{2}\int_S \left(1-\alpha_l\right)^2 \hat{n}_k \frac{\delta x_k}{\delta b_i} dS \tag{4.63}$$

Therefore, the field integral contributes to the FAEs in Eq. (4.25), while the second to the SDs in Eq. (4.56).

4.4 Concluding Remarks

In this chapter, the mathematical formulation of the adjoint problem for turbulent, cavitating two–phase flows was presented using the continuous approach. The resulting SDs expression is dependent solely on boundary integrals (SI approach). The flow model, whose adjoint counterpart was derived (FAEs, ABCs), consists of PDEs that express the volume and momentum conservation for the mixture, the volume conservation for the secondary (liquid) phase and the mixture turbulent variables transport equations ($k-\varepsilon$).

The formulation extends existing literature regarding the differentiation of the turbulence model, which was limited to single–phase incompressible flows (density variations are not included). Therefore, additional variations arise due to the mixture density and molecular viscosity. Furthermore, the special treatment of the wall functions, via linelets, employed in the CC method required additional considerations when deriving the adjoint wall functions. Furthermore, distance variations of the forcing points are axiomatically zero $\delta d^F = 0$ since the linelets are always constructed at the same wall–normal distance d^F.

In the presented TEM formulation, an additional PDE and the cavitation model were introduced and had their variations included in the adjoint problem as well. This leads to the

adjoint liquid volume fraction PDE and the adjoint cavitation model, respectively. Overall, the derived adjoint problem introduces terms that do not appear when single–phase flows with constant density and viscosity are considered. The mixture density variation terms that arise in homogeneous mixtures are similar to those in compressible flows, where density variations also exist but are, however, directly expressed w.r.t. the liquid volume fraction due to their algebraic relation Eq. (3.62). The same remarks also stand regarding mixture molecular viscosity variation; the arising terms share similarities with terms appearing if viscosity laws are considered [255].

Chapter 5

Shape Optimization of Single– and Two–Phase Flows using the Continuous Adjoint and the Cut–Cell method

THE adjoint method presented in Chapter 4 is used to compute the SDs in order to perform aero/hydrodynamic ShpO. In ShpO, the design variables control the surface of the body surface to be optimized. Thus, by updating the design variables, new better–designed body surfaces are created in each optimization cycle. However, computing the SDs (Eq. (4.56)) requires the derivatives of geometric quantities w.r.t. the design variables $\frac{\delta \mathcal{X}_s}{\delta b_i}$, the so–called Geometric Sensitivities (GSs).

The geometric quantities under consideration are the solid faces barycenter $\overline{\mathbf{x}}$, their unit normal vector $\hat{\mathbf{n}}$, and area dS, i.e.

$$\mathcal{X}_s = \left\{\overline{\mathbf{x}},\ \hat{\mathbf{n}},\ dS\right\} \tag{5.1}$$

The GSs can be computed either analytically, by differentiating the corresponding closed–form expressions w.r.t. the face vertices Eqs. (2.6), (2.7), or via FDs; this procedure involves purely geometric quantities defined over the body surface resulting in computationally cheap process [191].

The simplest approach in performing ShpO is by assuming that each discretized body sur-

face node* $\mathbf{x_s}^g$, $g \in \mathcal{G}$ is allowed to move independently and, therefore, each node coordinate becomes a design variable. This approach is usually followed to compute a sensitivity map on the body surface to be optimized since it provides information about the impact of infinitesimally displacing the body surface nodes along the normal direction to the considered objective function. ShpO runs using this approach can result in optimized body surfaces with crude/wavy boundaries and could, eventually, complicate the optimization process by leading the algorithm to non–realistic or non–manufacturable shapes. As a result, such approaches are usually combined with displacement/SDs smoothing or filtering techniques that may harm the optimization algorithm accuracy, as only the general trends of the computed SDs are obtained [106, 191, 225, 242]. Alternatively, parameterization techniques can be used that reduce the number of design variables by creating geometric dependencies between the body surface nodes that ensure smoothness and, to a certain level, *plasticity* to the parameterized shape [255].

5.1 Shape Parameterization in the Cut–Cell method

The aforementioned briefly describe the typical approach used in optimization tools that act upon traditional body–fitted meshes. In the CC method, these cannot be applied as–is. In the CC method, the mesh is generated by computing its intersections points $\mathbf{x_s}^n$, $n \in \mathcal{N}$ with the studied body surface, as described in Chapter 2. After each design variables update, new intersection points, constrained along the newly–created CCs edges, arise. Bearing this in mind, computing the GSs using FDs should be avoided in the CC method since infinitesimally perturbing the discretized body surface could lead to the emergence or disappearance of boundary faces and, thus, discontinuity in GSs [172, 226].

To overcome this, analytical expressions of the geometric quantities are, instead, differentiated. The intersection points at the CC edges are expressed w.r.t. the discretized body surface nodes and introduce additional dependencies and constraints (intersection points always lie along the CC edges). Taking into account the aforementioned, the GSs in the CC method are computed using

$$\frac{\delta \mathcal{X}_s}{\delta b_i} \equiv \frac{\delta \mathcal{X}_s}{\delta \mathbf{x_s}^g} \frac{\delta \mathbf{x_s}^g}{\delta b_i} = \frac{\delta \mathcal{X}_s}{\delta \mathbf{x_s}^m} \frac{\delta \mathbf{x_s}^m}{\delta \mathbf{x_s}^g} \frac{\delta \mathbf{x_s}^g}{\delta b_i}, \qquad m \in (\mathcal{G} \cup \mathcal{N}) \tag{5.2}$$

where $(\mathcal{G} \cup \mathcal{N})$ refers to the combined set of discretized body surface nodes $\mathcal{G}$ and the additional intersection points $\mathcal{N}$. The intersection points are accounted for through matrix $\frac{\delta \mathbf{x_s}^m}{\delta \mathbf{x_s}^g}$ and is zero

*$\mathbf{x_s}$ is used to refer to coordinates of nodes lying on the body surface

for the majority of $(m,\ q)$ pairs. $\frac{\delta \mathbf{x_s}^q}{\delta b_i}$ is included to facilitate cases where part of the discretized body surface defines the design variables (not all nodes or their components) . Otherwise, the last term is non–zero only when the body surface node coordinate coincides with the associated design variable.

The necessity of the additional matrix in Eq. (5.2) is illustrated using the explanatory $2D$ example of Figure 5.1a, showing a blow–up view close to the solid boundary. The part of the discretized body surface shown, consists of nodes $\mathbf{v}^q = (x^q, y^q)$, $q = 1, .., 4$, the intersection points $\mathbf{a} = (x^a, y^a), \mathbf{b} = \left(x^b, y^b\right)$ that are created during the generation of the CC and lead to the additional boundary faces. Focusing on the boundary face f, the additional dependencies can be seen by infinitesimally displacing node $\mathbf{v}^1$ (or $\mathbf{v}^2$) that leads to the displacement of the intersection point $\mathbf{a}' = \left(x^a, y^{a'}\right)$ along the CC edge, as shown in Figure 5.1b.

For this example, Eq. (5.2) can be rewritten without any simplifications as

$$\frac{\delta \mathcal{X}_s}{\delta b_i} = \frac{\delta \mathcal{X}_s^f}{\delta \mathbf{v}^1}\frac{\delta \mathbf{v}^1}{\delta \mathbf{v}^1}\frac{\delta \mathbf{v}^1}{\delta b_i} + \frac{\delta \mathcal{X}_s^f}{\delta \mathbf{a}}\frac{\delta \mathbf{a}}{\delta \mathbf{v}^1}\frac{\delta \mathbf{v}^1}{\delta b_i} + \frac{\delta \mathcal{X}_s^f}{\delta \mathbf{a}}\frac{\delta \mathbf{a}}{\delta \mathbf{v}^2}\frac{\delta \mathbf{v}^2}{\delta b_i} \tag{5.3}$$

and since $\mathbf{a}$ and $\mathbf{a}'$ are constrained along the same abscissa,

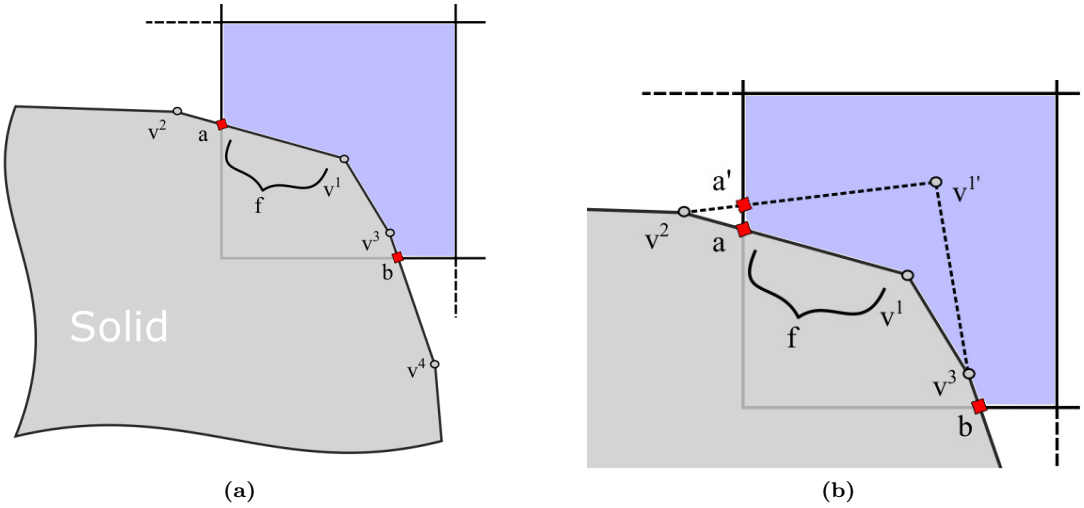
$$\frac{\delta \mathbf{a}}{\delta \mathbf{v}^1} = \begin{bmatrix} 0 & 0 \\ -\frac{x^a - x^2}{x^1 - x^2}\ \frac{y^1 - y^2}{x^1 - x^2} & \frac{x^a - x^2}{x^1 - x^2} \end{bmatrix} \tag{5.4}$$

Similar expressions are obtained for $\frac{\delta \mathbf{a}}{\delta \mathbf{v}^2}$ and intersection points that are constrained along the same ordinate, such as point $\mathbf{b}$. The geometric derivatives $\frac{\delta \mathcal{X}_s^f}{\delta \mathbf{v}^1}$ in Eq. (5.3) can then be computed by differentiating their analytical expressions. For instance, the geometric derivatives of the midpoint of the $2D$ boundary face are $\frac{\delta \overline{\mathbf{x}}^f}{\delta \mathbf{x_s}^q} = \frac{1}{2}\left(\delta_m^{v^1} + \delta_m^a\right)\mathbf{I}$, since $\overline{\mathbf{x}}^f = \frac{1}{2}\left(\mathbf{v}^1 + \mathbf{a}\right)$. When considering $3D$ GSs, the same approach is followed that results in similar expressions and can also be found in Samouchos [214]. In this way, GSs can be computed efficiently and without perturbing the discretized body surface, limiting the additional geometric computations entirely to the solid faces of the CC.

In this thesis, ShpO is performed on $2D$ body surfaces parameterized using Bézier–Bernstein polynomials [199]. The Bézier curves Control Points (CPs) coordinates $\mathbf{X}^c$ define the design variables, and are used to create the discretized body surface nodes $(\mathbf{x}^q)$ by inserting nodes along its curve. To compute the GSs using Bézier curves, the chain rule is employed leading to

$$\frac{\delta \mathcal{X}_s}{\delta b_i} \equiv \frac{\delta \mathcal{X}_s}{\delta \mathbf{X}^c}\frac{\delta \mathbf{X}^c}{\delta b_i} = \frac{\delta \mathcal{X}_s}{\delta \mathbf{x_s}^m}\frac{\delta \mathbf{x_s}^m}{\delta \mathbf{x_s}^q}\frac{\delta \mathbf{x_s}^q}{\delta \mathbf{X}^c}\frac{\delta \mathbf{X}^c}{\delta b_i} \tag{5.5}$$

where $\frac{\delta \mathbf{x_s}^q}{\delta \mathbf{X}^c}$ can be computed by differentiating their parametric expressions [191, 255, 282].



(a) (b)

Figure 5.1: Computation of GSs in the CC method - (a) Part of the discretized body surface intersecting a CC. $\mathscr{G} = \{\mathbf{v}^1,\ \mathbf{v}^2,\ \mathbf{v}^3,\ \mathbf{v}^4\}$: set containing the nodes of the discretized body surface; $\mathscr{N} = \{\mathbf{a},\ \mathbf{b}\}$: set containing the additional intersection points along the CC edges; $\mathbf{v}^1$, $\mathbf{v}^2$: body surface nodes; **a**, **b**: intersection points; f: solid face lying between $\mathbf{v}^1$ and **a**. (b) Displacement (exaggerated) of node $\mathbf{v}^1$ results in the displacement of the intersection point along the same abscissa.

5.2 Shape Optimization of Single–Phase Turbulent Flows

The developed continuous adjoint method and the shape parameterization technique mentioned above are implemented in three $2D$ single–phase turbulent flow cases. The objective functions applicable to single–phase flows are examined. These are the minimization of the volume–averaged total pressure losses J_{P_t} in 2 duct flows and the maximization of lift over an airfoil J_F. The baseline body surfaces are parameterized using Bézier curves; their CPs' coordinates that are allowed to change are the design variables of the optimization problem.

The ShpO algorithm iterates through optimization cycles, wherein the RANS, Eq. (3.54), and FAEs, Eq. (4.25), are successively solved with the appropriate boundary conditions (Section 4.2.5) to compute the SDs, Eq. (4.56). The GSs are computed using Eq. (5.5). The design variables are updated using steepest descent with a constant step size, computed at the first optimization cycle using a prescribed maximum allowable displacement and the maximum norm of the computed SDs.

5.2.1 Channel Turbulent Flow – min J_{P_t}

The first example concerns the ShpO of an initially straight channel, exhibiting turbulent flow, aiming at the minimization of the volume–averaged total pressure losses (Section 4.3.1).

The Reynolds number is $Re_W = 1 \times 10^5$, based on the channel width. Inlet conditions are: the velocity magnitude ($1\frac{m}{s}$), the velocity direction ($\theta_{XY} = 0°$), turbulent intensity (1%) and eddy viscosity ratio ($\frac{\mu_t}{\mu} = 5$). At the outlet, the static pressure is zero. The CC mesh created in the initial simulation amounts to approximately $60K$ cells and results in a maximum $y^+ \approx 27$ based on the examined flow conditions. Part of the upper and lower channel walls are symmetrically parameterized using two Bézier curves giving rise to 20 design variables, Figure 5.2. These ensure that the width of the channel at the inlet/outlet remains constant throughout the optimization, with no effect on the computed objective function.

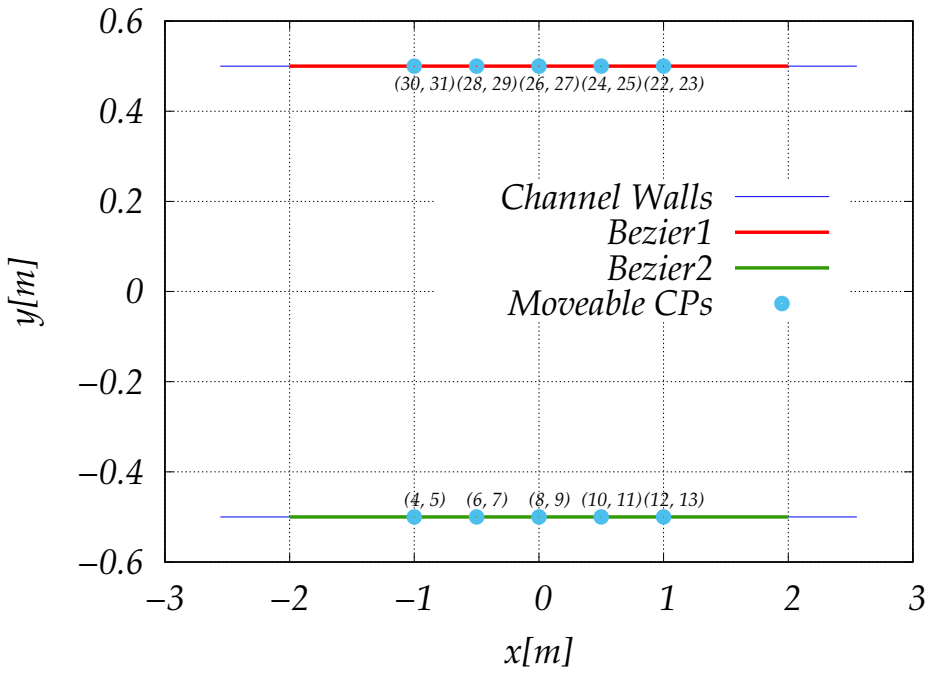


Figure 5.2: ShpO of a channel turbulent flow, $Re_W = 1 \times 10^5$ - Symmetrically placed Bézier curves along with their CPs that parameterize the channel walls. Displacing a CP modifies the part of the shape it controls.

In Figure 5.3, the flow and adjoint equation residuals are shown for the initial channel. The SDs are also computed using FDs, to be used as reference values and verify the accuracy of the implemented continuous adjoint method. The use of FDs method required a substantial amount of resources, compared to the adjoint method, since computing the SDs for 20 design variable required 40 flow simulations. Figure 5.4a shows the comparison between the two, where the anti–symmetric nature of the SDs between the upper and lower channel walls is visible. The x–coordinate SDs are seen to have zero value and shows that displacing the CPs along the

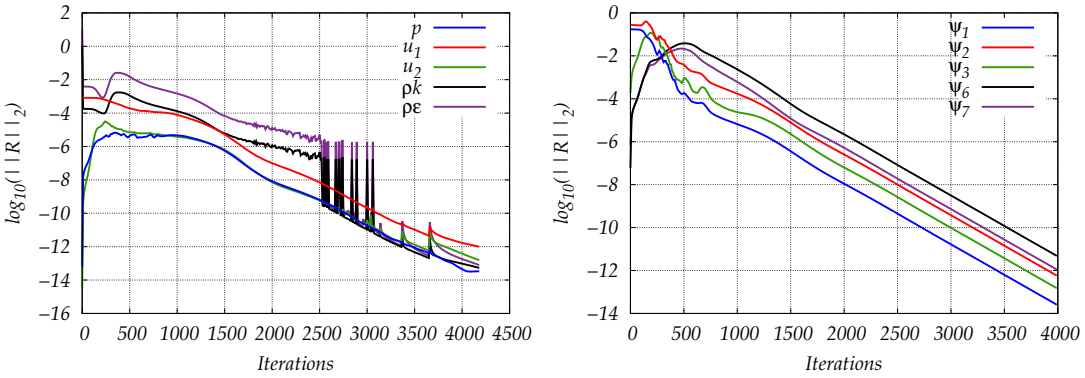
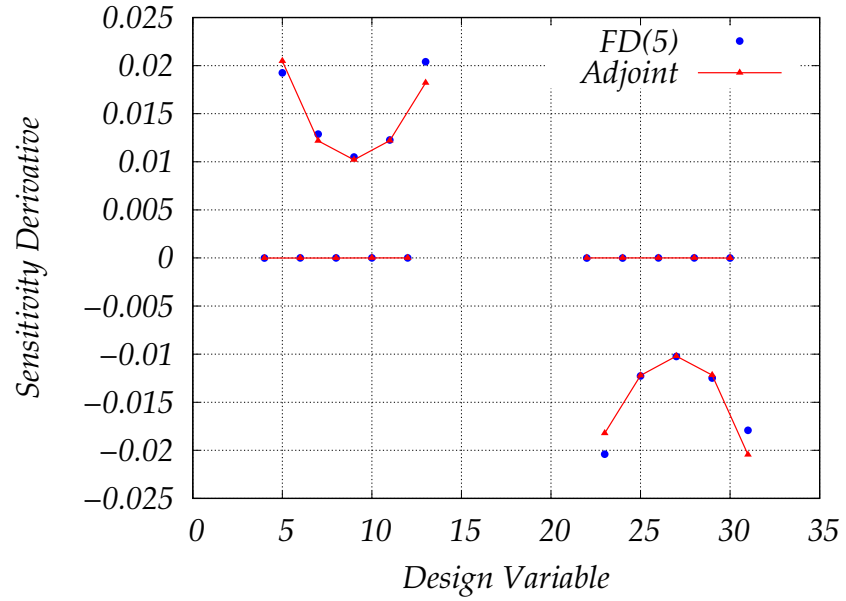


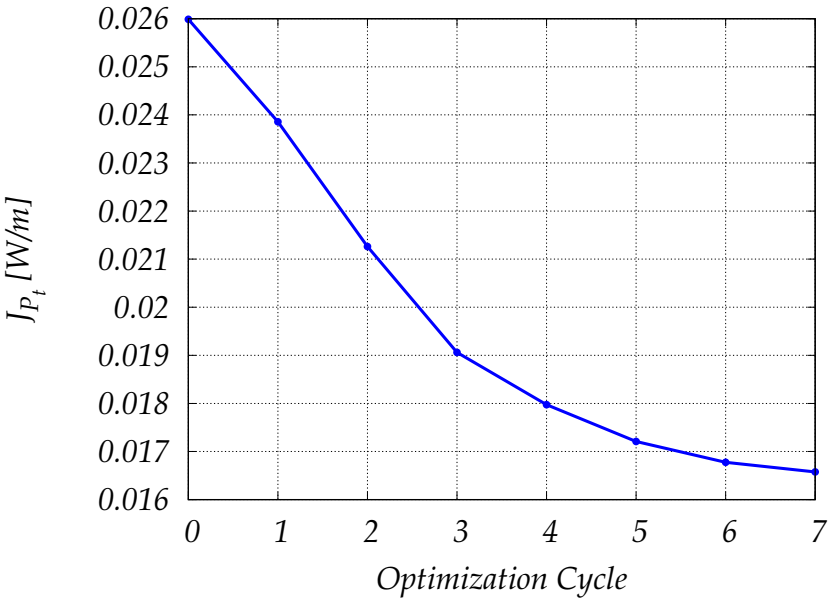
Figure 5.3: ShpO of a channel turbulent flow, $Re_W = 1 \times 10^5$ - Colors correspond to the (a) flow and (b) adjoint equations residual history at the first optimization cycle, i.e. ψ_1 refers to the adjoint pressure, $\psi_{2,3}$ to the adjoint velocity components and $\psi_{6,7}$ to the adjoint turbulence variables. A convergence of more than 10 orders is observed in both residuals. The run time for the solution of the flow and adjoint system of equations, altogether, is approximately 200 min on 24 AMD EPYC 7401 (2.0 Ghz) processors.

x–axis has no effect on the objective function value on the first optimization cycle, as this does not change the channel walls shape.

The optimization is performed for 7 cycles in which the objective function (J_{P_t}) was reduced by approximately 40%. Figure 5.4b shows the evolution of the J_{P_t} throughout the optimization cycles. During the optimization, the channel width gradually increased at the parameterized sections. Increasing the channel width reduces the bulk fluid velocity, which leads to smaller stresses at the larger portion of the solid walls. This translates to a reduction of mechanical losses inside the channel. In Figure 5.6, part of the initial (blue) and final (red) CC meshes generated during the ShpO can be seen. In each optimization cycle, the CC mesh automatically adapts on the current surface introducing new finite volumes at the expanded part. In contrast to conventional body–fitted meshes, this also allows significantly large displacements since the generated mesh does not need to be modified using mesh deformation techniques that can progressively deteriorate the mesh quality. In addition, the increase in the channel width leads to previously intersected Cartesian cells to reduce their resolution, i.e. reach a lower refinement level, saving computational resources. Finally, in Figure 5.5, the eddy viscosity iso–areas are plotted for the initial and optimized channel walls.



(a)



(b)

Figure 5.4: ShpO of a channel turbulent flow, $Re_W = 1 \times 10^5$ - (a) Comparison of the SDs computed using the continuous adjoint method and central FDs with a step–size of 10^{-5}. (b) Evolution of J_{P_t} through the optimization cycles.

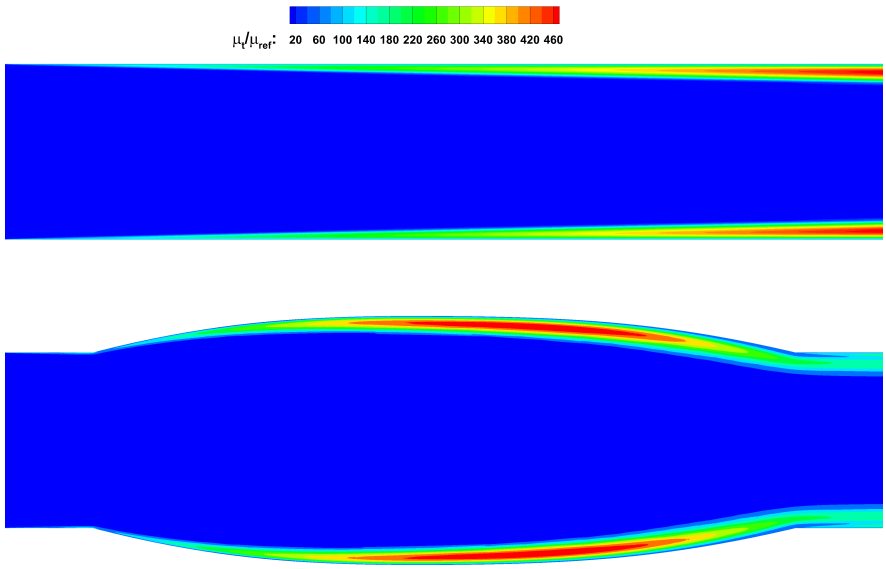


Figure 5.5: ShpO of a channel turbulent flow, $Re_W = 1 \times 10^5$ - Eddy viscosity iso–areas of the initial and optimized channel walls.

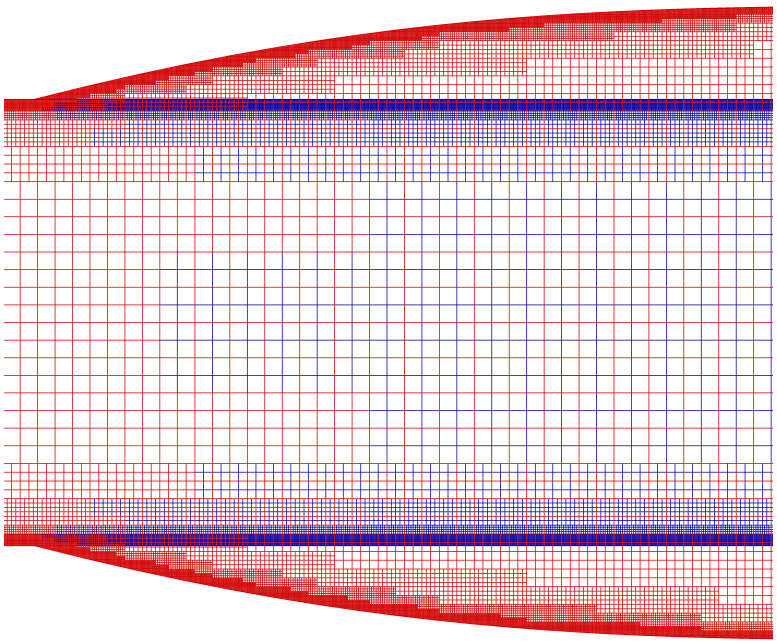


Figure 5.6: ShpO of a channel turbulent flow, $Re_W = 1 \times 10^5$ - Close–up view of the CC mesh generated for the initial (blue) and optimized (red) channel walls. The optimization algorithm expands the parameterized areas of the straight channel leading to the appearance of new cells. The inner mesh resolution automatically adapts to the new channel walls.

5.2.2 90° Curved Channel Turbulent Flow – min J_{P_t}

The second example concerns the ShpO of a turbulent flow in a 90° curved channel targeting the minimization of J_{P_t}. This case is used to assess the adjoint linelet technique developed especially for the CC method in the presence of curved surfaces and, therefore, irregular CC. The Reynolds number is equal to $Re_W = 1 \times 10^4$. The CC mesh created in the initial simulation amounts to approximately $12K$ cells and results in a maximum $y^+ \approx 40$, based on the specified flow conditions. These consist of specifying the total pressure ($0.5Pa$), the velocity vector direction ($\theta_{XY} = 0°$), the turbulent intensity (2%), and the eddy viscosity ratio $\left(\frac{\mu_t}{\mu} = 2\right)$ along the inlet and the static pressure ($0Pa$) at the outlet.

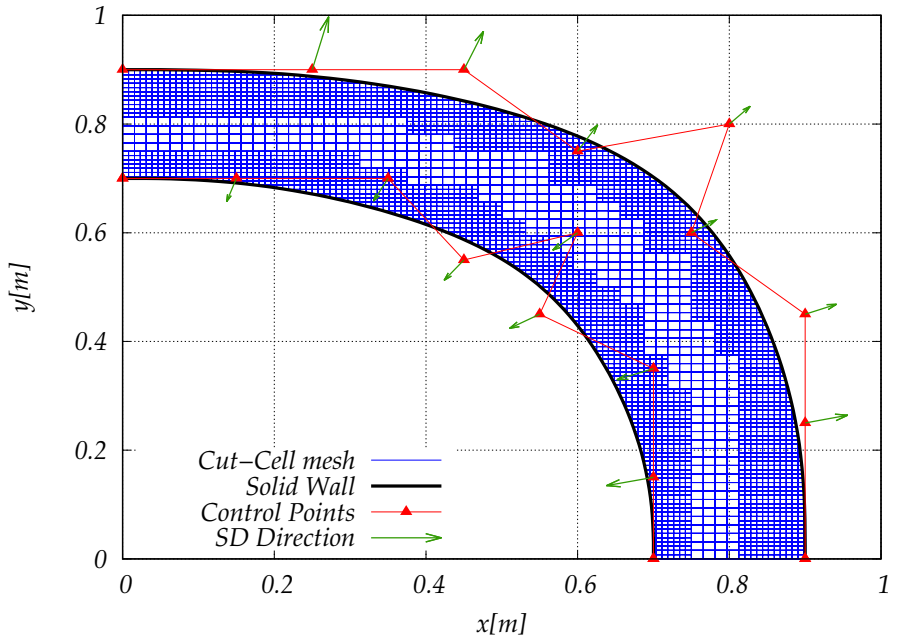


Figure 5.7: ShpO of a 90° curved channel turbulent flow, $Re_W = 1 \times 10^4$ - Close–up view of the curved channel showing the generated CC mesh of the initial channel walls along with the two Bézier curves used generate them. Vectors originating from the Bézier curves' CPs show the direction of displacement that leads to a reduced J_{P_t} in the first optimization cycle.

In Figure 5.7, a close–up view of the curved channel walls is shown along with the generated CC mesh. In the same figure, the computed SDs at the first optimization cycle are plotted for all CPs using the computed x– and y– SDs components to create the depicted vectors; note that the CPs with $x = 0$ or $y = 0$ are anchored. These vectors, in effect, show that in order to minimize the objective, the parameterized part of the channel needs to be expanded, similarly to the previous case.

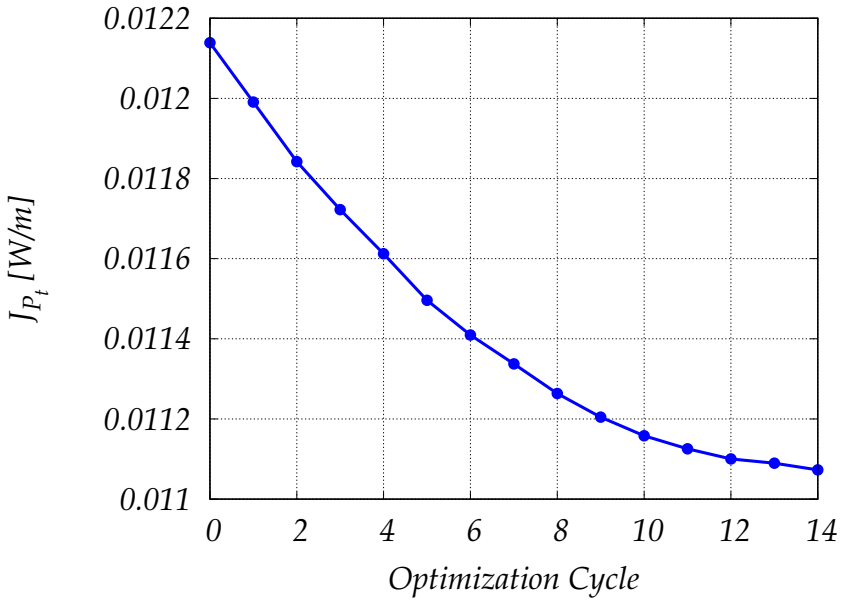


Figure 5.8: ShpO of a 90° curved channel in turbulent flow, $Re_W = 1 \times 10^4$ - Evolution of J_{P_t} through the optimization cycles.

The optimization is performed for 14 cycles throughout which the J_{P_t} was progressively reduced by a total of ∼10% (Figure 5.8). In Figure 5.9 the velocity magnitude iso–areas of the initial and final curved channel are shown. The fluid velocity is reduced at the expanded part of the channel, which leads to the reduced J_{P_t} values. Similar remarks can be made regarding the CC mesh generation process. However, this case demonstrates the ability of the CC method to optimize bodies featuring curved surfaces in turbulent flows, using the presented wall function technique. Furthermore, this allows for the use of relatively coarser computational meshes respecting the wall functions' y^+ limitations. Overall, the run time for the solution of both the flow and adjoint equations required about 30 minutes running on 24 AMD EPYC 7401 (2.0 Ghz) processors while the optimization time totaled about 7 hours.

Figure 5.9: ShpO of a 90° curved channel turbulent flow, $Re_W = 1 \times 10^4$ - Velocity magnitude iso–areas of the initial and optimized curved channel.

5.2.3 Turbulent Flow over the NACA 0012 – max J_F

The next example concerns the ShpO of the NACA 0012 airfoil targeting the maximization of lift force, i.e. $\hat{\boldsymbol{r}} = \left[-\sin\alpha_\infty \;\; \cos\alpha_\infty\right]^T$ in Section 4.3.2, using the NACA 0012 airfoil as the baseline shape to be optimized. A Reynolds number of $Re_c = 2 \times 10^6$ is specified. The free–stream conditions imposed are $p_\infty = 0Pa$, $U_\infty = 10\frac{m}{s}$, $\theta_{XY} = 2°$, $I_\infty = 2\%$ and $\frac{\mu_t}{\mu} = 10$. The initial CC mesh generated for the solution of the flow and adjoint equations consists of approximately $45K$ cells and yields a maximum $y^+ \approx 150$.

Two Bézier curves are used to parameterize the pressure and suction side of the airfoil. Since the initial airfoil shape is symmetric, the first Bézier curve, consisting of 8 CPs, is mirrored along the y–axis to create the second Bézier curve giving rise to a total of 16 CPs as shown in Figure 5.10. The CPs at the leading and trailing edge of the airfoil are fixed to their initial position, to maintain the same chord length c throughout the optimization. Thus, the design variables in the ShpO amount to 24.

In Figure 5.11, the evolution of J_F is shown during the optimization cycles, where a significant increase in the objective function is observed ($> 10\times$). This increase is expected due to the low–lift initial configuration. Starting with a symmetric airfoil shape at low AoA, small displacements break this symmetry leading to significant lift gains. In Figure 5.11, the change in drag force induced by the airfoil is also monitored, computed using $\hat{\boldsymbol{r}} = \left[\cos\alpha_\infty \;\; \sin\alpha_\infty\right]^T$ in Eq. (4.60). Interestingly, even though the ShpO successfully leads to designs with better

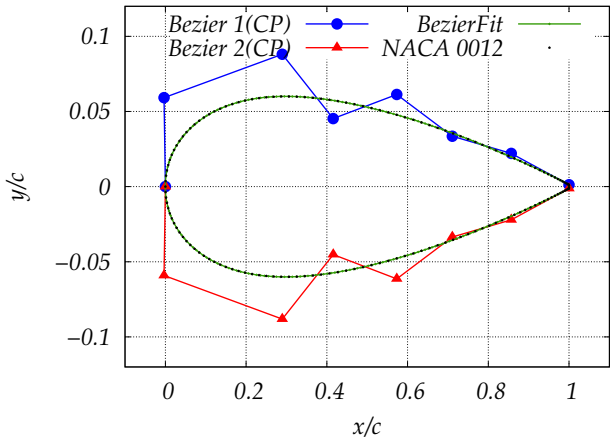


Figure 5.10: ShpO of the NACA 0012 airfoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 2°$ - Bézier curves used to parameterize the pressure and suction side of the airfoil (x and y not in scale). CPs at the leading and trailing edge are anchored.

objective function values, drag force is seen to increase. Specifically, the drag of the optimized design almost doubles compared to the initial shape. The entire ShpO requires approximately 10 hours on 48 AMD EPYC 7401 (2.0 Ghz) processors .

Figure 5.13 shows a comparison of the initial and optimized airfoil shapes. The key features of the optimized airfoil shape are the development of a cambered suction side and the convex shape at the airfoil pressure side near the trailing edge. These features change the performance of the optimized airfoil by decreasing the pressure along the suction side and increasing it along the pressure side. This is evident in Figure 5.12a, where the iso–bar areas of the initial and optimized airfoil shapes are presented. In Figure 5.12b the corresponding eddy viscosity iso–areas are depicted revealing a relative increase at the location of the appearance of the convex shape along the suction side that also contributes to the induced drag force.

Figure 5.14 focuses on the location where the convex feature appears over the optimized airfoil shape. The CC mesh generated for the initial (blue) NACA 0012 is superposed on that of the optimized one (red). During the ShpO, a significant displacement of the airfoil contours is observed that causes the appearance and disappearance of multiple finite volumes along with the pressure and suction side, respectively. In addition, the generated CC mesh is refined accordingly since Cartesian cells of higher refinement level appear at the top side (and Cartesian cells of lower refinement level at the bottom side) of the close–up view shown due to the displacement of the airfoil contour.

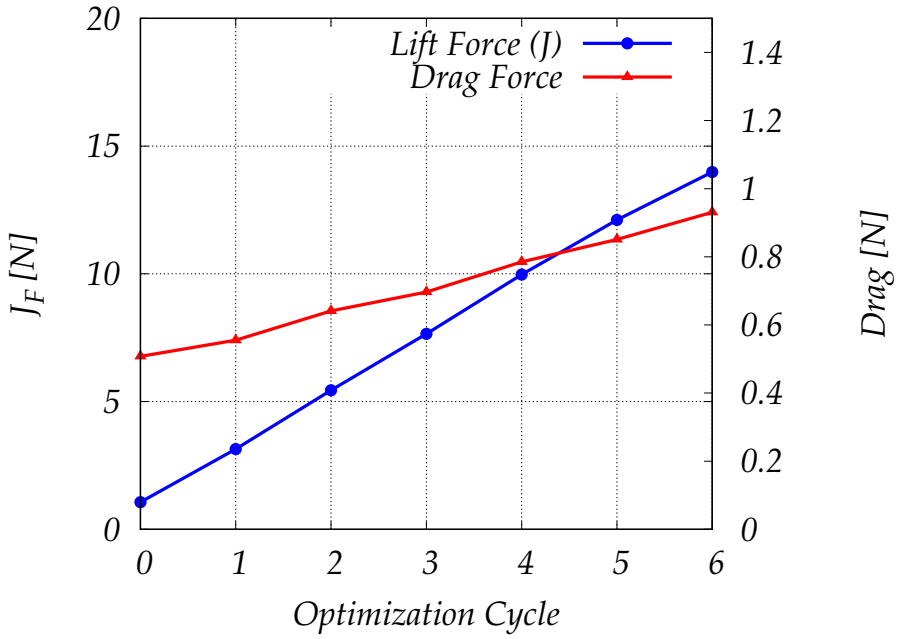
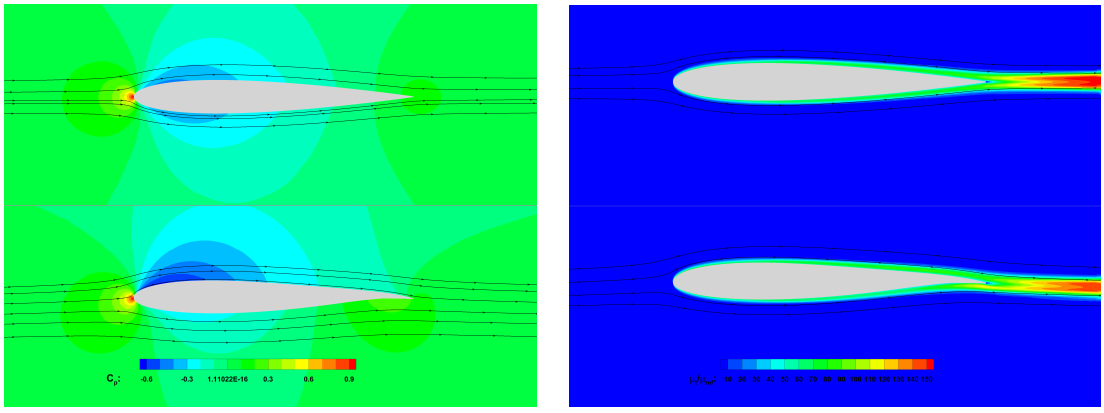


Figure 5.11: ShpO of the NACA 0012 airfoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 2°$ - Evolution of J_F through the optimization cycles, while also monitoring the drag force.



(a) **(b)**

Figure 5.12: ShpO of the NACA 0012 airfoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 2°$ - (a) Pressure and (a) eddy viscosity iso–areas of the initial and optimized airfoil shapes.

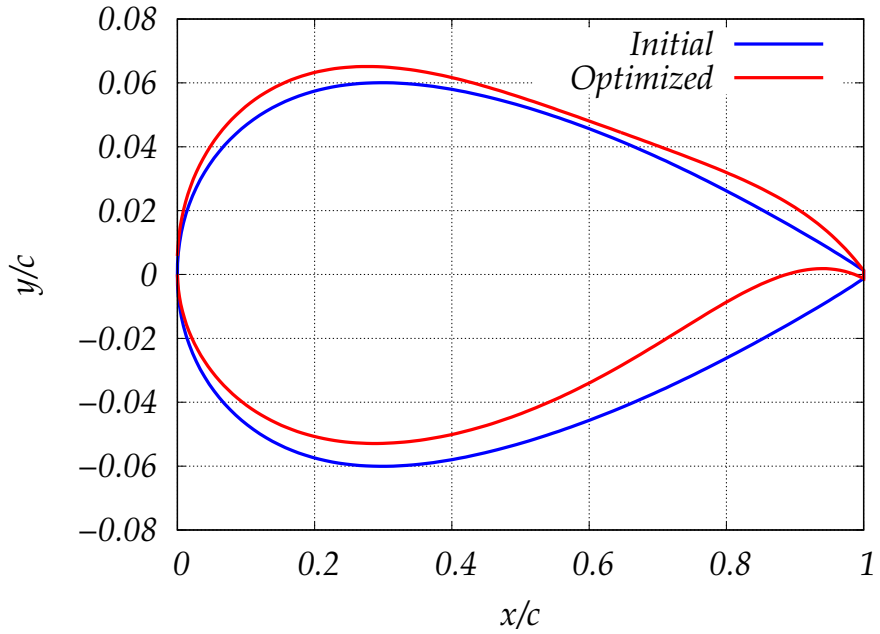


Figure 5.13: ShpO of the NACA 0012 airfoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 2°$ - Comparison of the initial and optimized airfoil shapes.

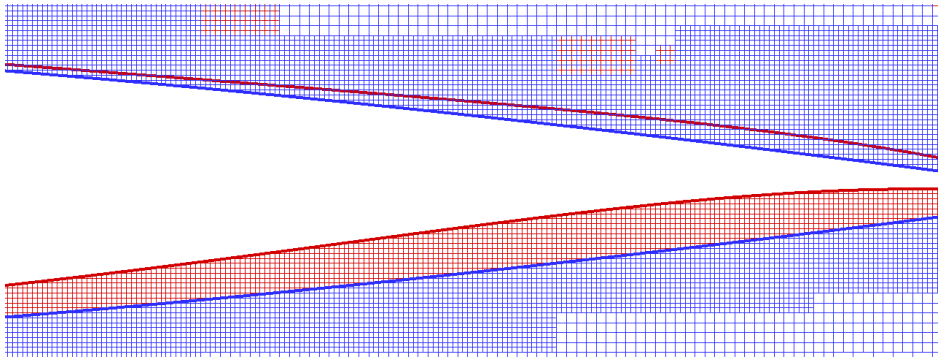


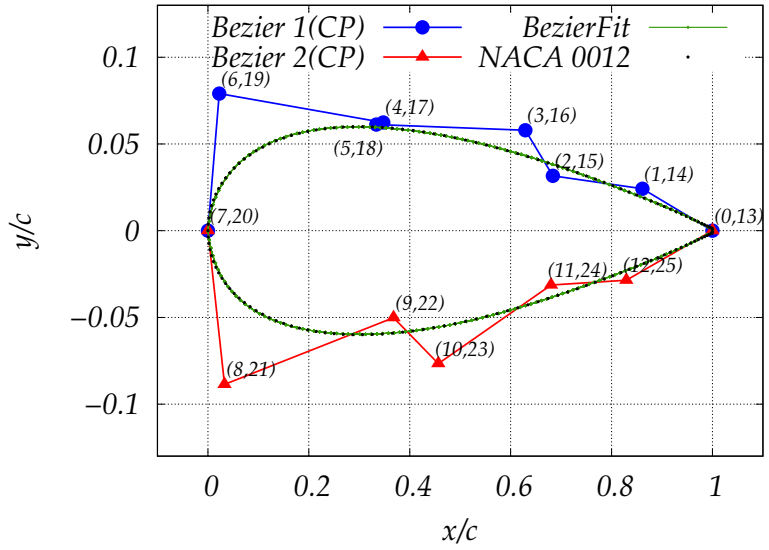
Figure 5.14: ShpO of the NACA 0012 airfoil, $Re_c = 2 \times 10^6$, $\alpha_\infty = 2°$ - Close–up view of the CC mesh generated for the initial and optimized airfoil shapes. The additional/reduced refinement due to the displacement of the airfoil shape can be seen.

5.3 Shape Optimization of Two–Phase Cavitating Flows

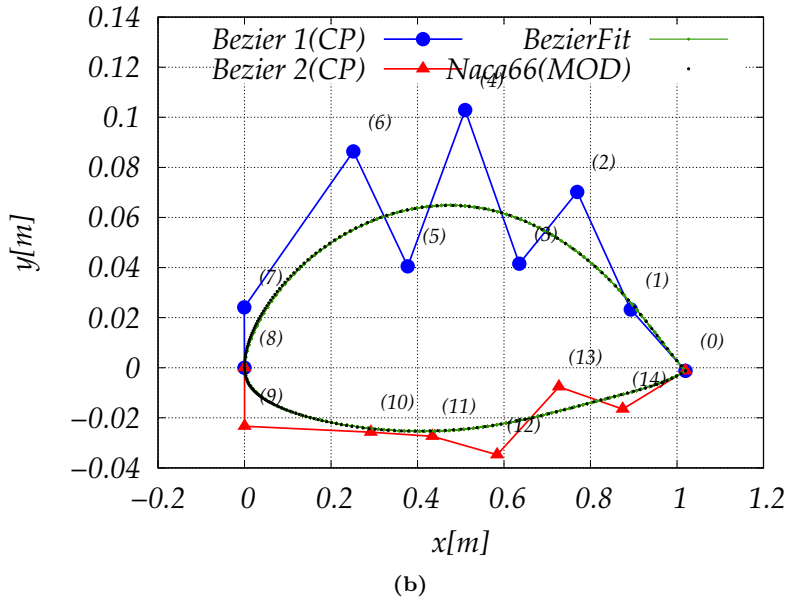
In case two–phase flows are considered, additional dependencies in the governing equations due to the introduction of the liquid volume fraction transport equation, the cavitation model that accounts for the phase transition, and the volume–averaged mixture properties due to the homogeneous model (Eqs. (3.62), (3.63)), arise. These lead to additional source terms in the FAEs, and two–phase specific terms in the expression of SDs, overviewed in Chapter 4 (c.f. [191, 214, 255]).

In this section, the derived continuous adjoint method is implemented for the ShpO of two well–known test cases, for which both numerical and experimental data are available, in inviscid and viscous flows over hydrofoils at cavitating conditions [36, 64, 223]. Initially, the two–phase flow solver accuracy is assessed by comparing results obtained with experimental and/or numerical data, and then, the adjoint two-phase solver accuracy is verified by comparing the SDs of the adjoint method against central FDs. Then, ShpOs are performed for the implemented objective functions, as overviewed in Section 4.3.

The studies concern the NACA 66(MOD) [40] and the NACA 0012 hydrofoils and are used as the baseline shapes for the subsequent ShpOs. Each hydrofoil contour is formed by two Bézier curves, with common control points at the leading and trailing edges. The coordinates of the initial Bézier curve control points are computed using a curve fitting technique by pre–defining the number of control points in each curve. Along the suction side, the number of control points is higher to increase the number of design variables close to the expected location of the cavity. This provides additional degrees of freedom to the hydrofoils' suction side parameterization. Figure 5.15 presents the fitted Bézier curves with their control points, along with the reference contours. For the ShpO, the x–coordinates of the CPs parameterizing the NACA 66(MOD) hydrofoil are fixed and, thus, only the y–coordinates are numbered.



(a)



(b)

Figure 5.15: Baseline hydrofoil shapes used for ShpO runs in two–phase cavitating flows - Hydrofoil shape (x and y not in scale) along with the fitted Bézier curves for the (a) NACA 0012 and (b) NACA 66(MOD) hydrofoils. The numbering of the design variables shown for each hydrofoil helps identify the computed SDs in subsequent figures.

5.3.1 Validation of Cut–Cell Two–Phase Flow Solver

5.3.1.1 Inviscid Cavitating Flow over the NACA 66(MOD) Hydrofoil

The first case considered to further validate the two–phase CC flow solver, concerns the simulation of an inviscid cavitating flow over the NACA 66(MOD) hydrofoil. A cavitation number of $\sigma = 0.38$ and an AoA of $\alpha_\infty = 1^\circ$ are specified, along with a freestream velocity of $U_\infty = 10.0\frac{m}{s}$ and a density ratio of $\frac{\rho_l}{\rho_v} = 100$. At these flow conditions, a mid–chord cavity appears and is convected downstream up until its closure. The cavitation model mass transfer time rate constants are set to $C_{dest} = 1 \times 10^3$ and $C_{prod} = 40$. Mesh–independent solutions were obtained using a CC mesh of $\sim 30K$ cells after a parametric study was conducted using three CC meshes of different resolution. A close-up view of the resulting mesh is shown in Figure 5.16, where the additional mesh refinement using the distance–based and window–based conditions. In Figure 5.17a, the computed surface pressure distribution is presented for the three CC meshes generated during the parametric study along with the experimental data of Shen & Dimotakis [228].

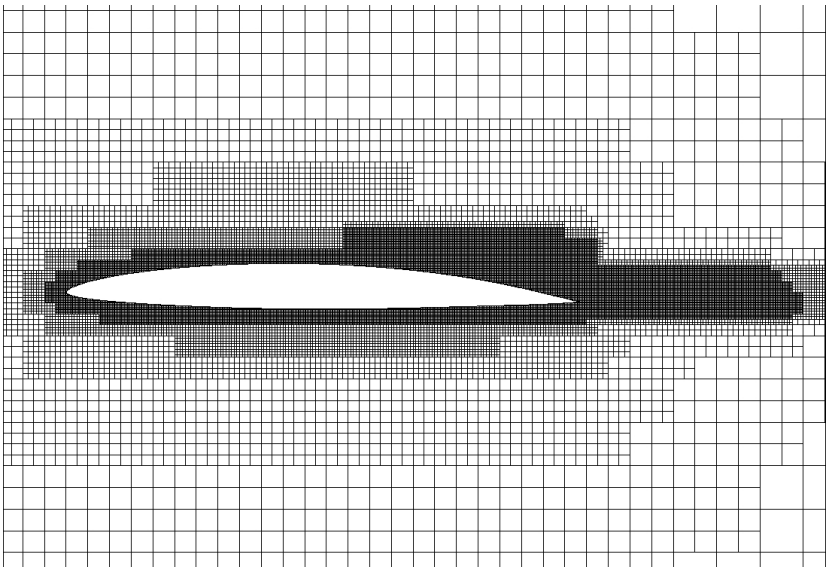
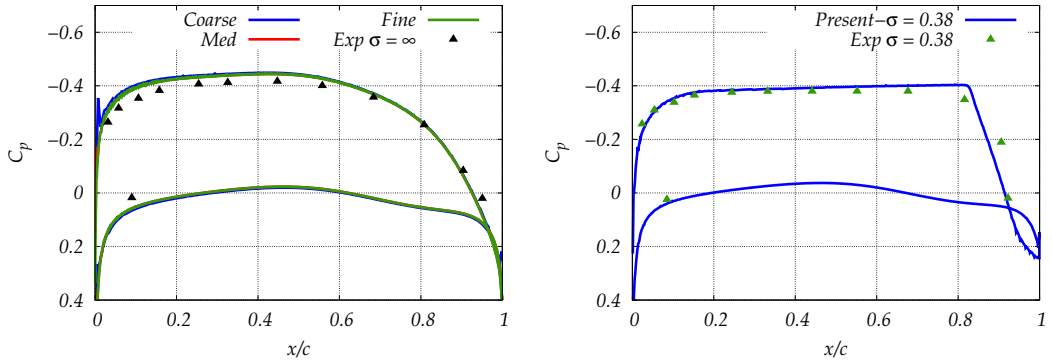


Figure 5.16: Inviscid flow over the NACA 66(MOD) hydrofoil, $\alpha_\infty = 1^\circ$, $\sigma = (\infty, 0.38)$ - Close–up of the hydrofoil showing the generated CC mesh with $\sim 30K$ cells. An extended cell refinement at the hydrofoil wake and expected cavity location is visible due to the application of the window–based refinement.

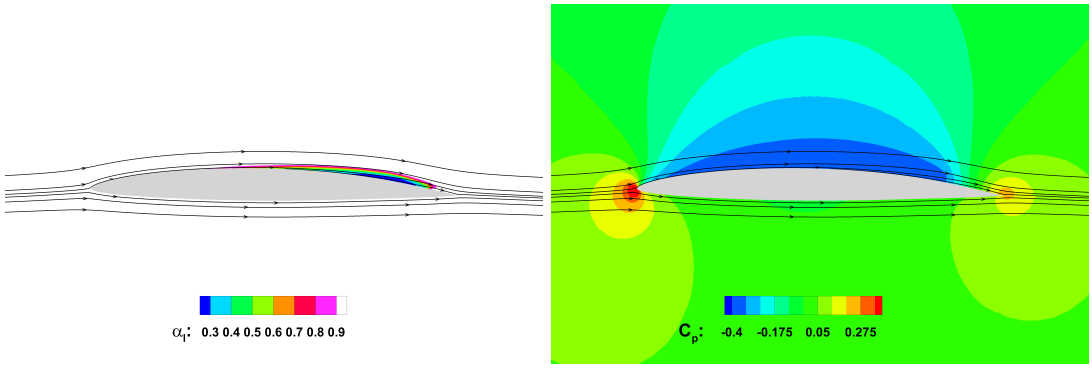
The comparisons are performed in terms of computed surface pressure distributions against experimental data, which are depicted in Figure 5.17b. Good agreement is observed in terms of the location of the cavity inception point and its length, being consistent with numerical results



(a) **(b)**

Figure 5.17: Inviscid flow over the NACA 66(MOD) hydrofoil, $\alpha_\infty = 1°$, $\sigma = (\infty, 0.38)$ - Comparison of the computed surface pressure distributions with measurements of Shen & Dimotakis [228] for (a) the non–cavitating ($\sigma = \infty$) and (b) cavitating ($\sigma = 0.38$) case.

obtained in the literature [35, 64, 73]. The most apparent discrepancies exist near the cavity closure region, that as stated in Deshpande *et al.* [64], cannot be exactly pinpointed from the measured experimental data.



(a) **(b)**

Figure 5.18: Inviscid, cavitating flow over the NACA 66(MOD) hydrofoil, $\alpha_\infty = 1°$, $\sigma = 0.38$ - Iso–areas of (a) liquid volume fraction and (b) pressure with streamlines over the hydrofoil with a mid–chord cavity.

The CC flow solver successfully predicts the mid–chord cavity when simulating the cavitating case ($\sigma = 0.38$). At approximately 30% from the hydrofoil leading edge, the liquid phase starts to evaporate, and this generates a thin pocket of vapour further downstream, as shown in Figure 5.18a. Figure 5.18b depicts the corresponding pressure iso–lines revealing that at the cavity location, a relatively constant pressure, approximately equal to the vapour pressure (p_v), is

observed. Finally, at the cavity closure region, no recirculation exists [221].

5.3.1.2 Laminar, Cavitating Flow over the NACA 0012 Hydrofoil

The second case deals with a laminar cavitating flow over the NACA 0012 hydrofoil. Two different cavitation numbers are considered, namely $\sigma = \infty$ (non-cavitating) and $\sigma = 0.5$, at an AoA of $\alpha_\infty = 4°$, a density ratio of $\frac{\rho_l}{\rho_v} = 1000$, and a Reynolds number of $Re_c = 500$, to be able to compare with numerical results presented in [93]. The mass transfer time rate constants of the cavitation model are set to $C_{dest} = 2 \times 10^4$ and $C_{prod} = 200$. The two–phase solver is compared with numerical results shown in Hejranfar *et al.* [93], that are obtained using the barotropic approach, i.e. phase transition is regulated by a barotropic state law instead of a cavitation model. The straightforward comparison of the results obtained with the two approaches is possible since both have been validated with experimental data [36, 167], and shown to yield similar results in similar numerical comparisons [94].

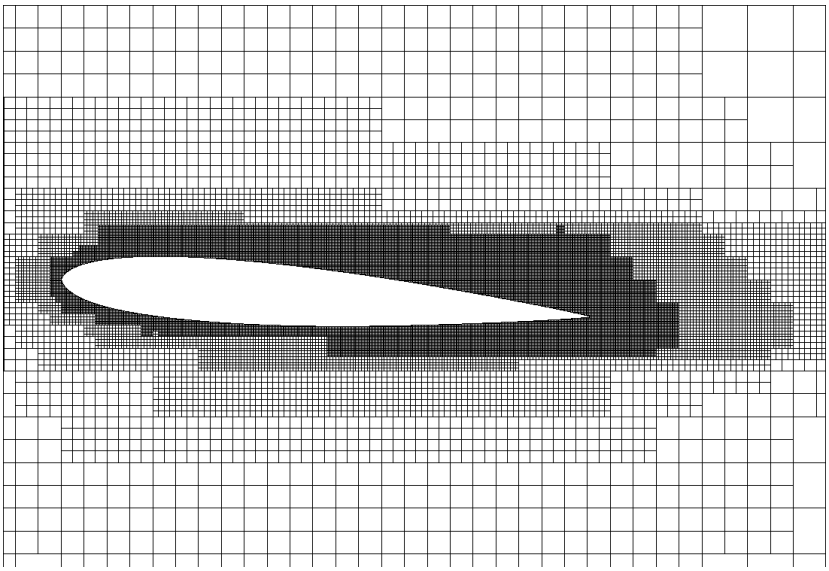


Figure 5.19: Laminar, cavitating flow over the NACA 0012 hydrofoil, $Re_c = 500$, $\alpha_\infty = 4°$, $\sigma = 0.5$ - Close–up of the hydrofoil showing the CC mesh with $\sim 30K$ cells. The application of the additional refinement conditions lead to denser Cartesian mesh near the expected cavity location that progressively coarsens.

For the simulations, a CC mesh, consisting of approximately $30K$ cells, was generated with sufficient resolution close to the hydrofoil and expected cavity location, as shown in the close–up view in Figure 5.19. In Figure 5.20, the computed surface pressure distributions are presented for both cases showing very good agreement with the numerical data. In the cavitating case,

a large cavity appears at the suction side of the hydrofoil depicted in Figure 5.21a. The CC solver accurately captures the expected length and inception point, while at the cavity location, the vapour pressure is recovered. Figures 5.21a and 5.21b respectively show the liquid volume fraction iso-areas and iso-bar areas for the cavitating case with selected streamlines revealing a big recirculation region aft the cavity.

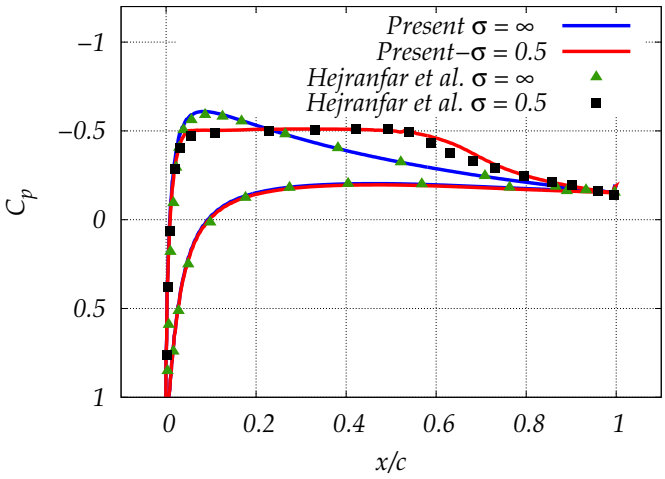
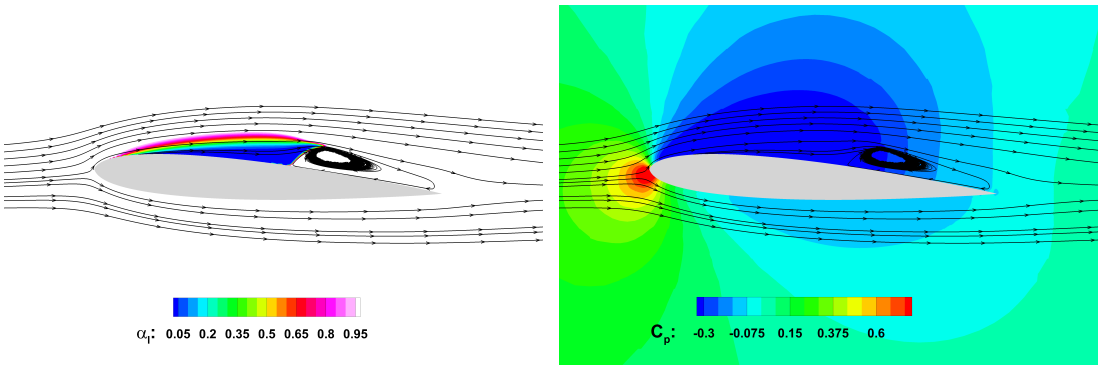


Figure 5.20: Laminar flow over the NACA 0012 hydrofoil, $Re_c = 500$, $\alpha_\infty = 4^\circ$ - Comparison of the computed surface pressure distributions with numerical data [93] for the non–cavitating ($\sigma = \infty$) and cavitating ($\sigma = 0.5$) case.



(a) **(b)**

Figure 5.21: Laminar, cavitating flow over the NACA 0012 hydrofoil, $Re_c = 500$, $\alpha_\infty = 4^\circ$, $\sigma = 0.5$ - Iso–areas of (a) liquid volume fraction and (b) pressure with streamlines over the cavitating hydrofoil.

5.3.2 Two–phase Shape Optimizations

5.3.2.1 Inviscid, Cavitating Flow over the NACA 66(MOD) Hydrofoil – min J_V

The first two–phase ShpO aims at minimizing the presence of cavitation by reducing the total vapour volume fraction throughout the computational domain (see Section 4.3.3). For validation purposes, the gradients of the J_V objective function are computed using the implemented adjoint method and FDs and are presented in Figure 5.22 for the first optimization cycle. This comparison reveals that the SDs computed by the adjoint method are in excellent agreement with FDs for both hydrofoil sides. The largest discrepancy exists at the design variable b_0, which corresponds to the CP at the trailing edge and could be attributed to the highly irregular CCs that are created due to the hydrofoil sharp edge.

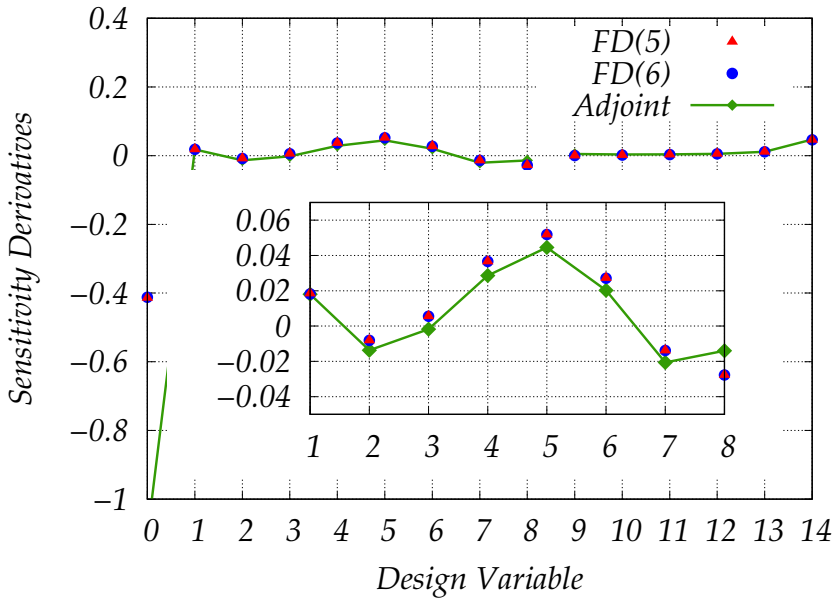


Figure 5.22: ShpO of the NACA 66(MOD) hydrofoil targeting minimization of vapour volume fraction (min J_V), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Comparison of the J_V SDs obtained by the continuous adjoint method and FDs, with a close-up view near the suction side. For reference to the design variable numbering see Figure 5.15b.

The entire ShpO required approximately 6 hours on 48 AMD EPYC 7401 (2.0 Ghz) processors to perform 10 optimization cycles. During the optimization, the lift force values of intermediate designs is also monitored using $\hat{\boldsymbol{r}} = \left[-\sin\alpha_\infty \;\; \cos\alpha_\infty\right]^T$ in Eq. (4.60). Figure 5.23 shows the objective function and lift force values in each optimization cycle starting from the NACA 66(MOD) hydrofoil. A very good convergence of the objective function is observed

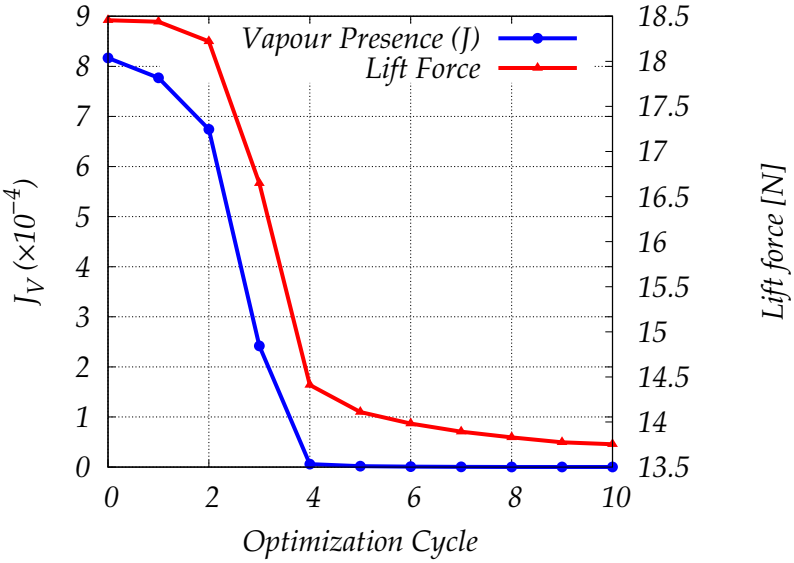
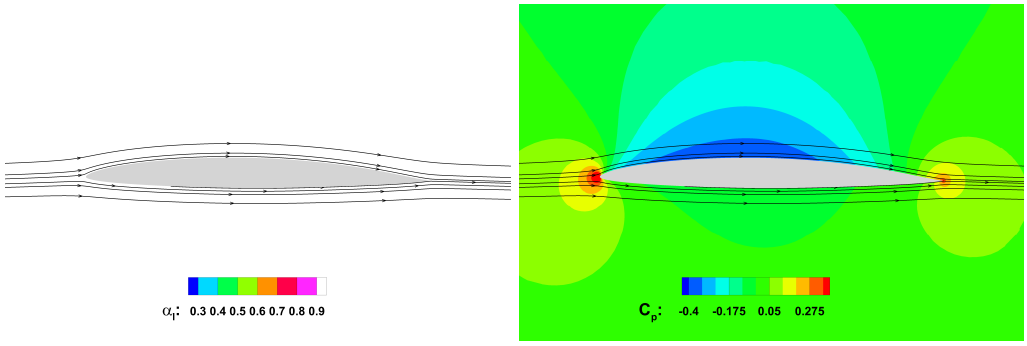


Figure 5.23: ShpO of the NACA 66(MOD) hydrofoil targeting minimization of vapour volume fraction (min J_V), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Evolution of the J_V values through the optimization cycles, while also monitoring the lift force.

as the cavitation is effectively eliminated after 5 optimization cycles. Afterwards, the objective function value asymptotically tends to zero. It appears that a minimum has been almost reached (flow with no vapour) and, therefore, the adjoint equations provide a more or less zero-field solution associated with practically negligible SDs and design variable updates.

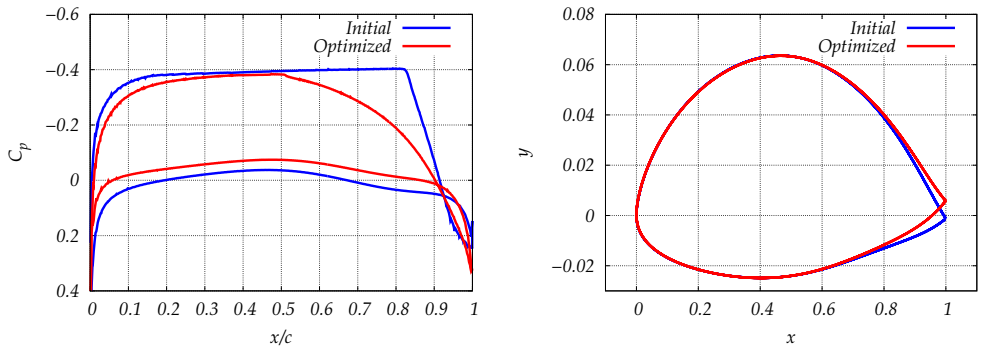
Comparison of the evolution of the hydrofoil lift and the objective function J_V in the same figure reveals a similar trend. This can be explained by comparing the surface pressure distributions of the initial and optimized hydrofoil shape in Figure 5.25a. In an effort to reduce the presence of vapour around the hydrofoil, the optimization algorithm ought to increase the pressure above the vapour pressure. As such, the pressure along the suction side is increased, reaching the vapour pressure to a lesser extent and inadvertently reducing the hydrofoil lift force, Figure 5.24b. This trend, where the lift induced by a hydrofoil follows the cavity length, was also documented by Knapp *et al.* [125] and Shen & Dimotakis [228]. Their findings suggest that the increase in lift induced by the hydrofoil follows the cavity length up to a certain point where lift breakdown happens. For reference, the lift breakdown for the NACA 66(MOD) hydrofoil occurs when the cavity length covers ∼83% of the chord [228]

In the optimized shape, the vapour pressure is barely reached at approximately 50% of the chord from the leading edge, resulting in the almost vapour–free flow shown in Figure 5.24a. A



(a) **(b)**

Figure 5.24: ShpO of the NACA 66(MOD) hydrofoil targeting minimization of vapour volume fraction (min J_V), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Iso–areas of (a) liquid fraction and (b) pressure with streamlines over the optimized hydrofoil.



(a) **(b)**

Figure 5.25: ShpO of the NACA 66(MOD) hydrofoil targeting minimization of vapour volume fraction (min J_V), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Comparison of (a) the computed surface pressure distributions for the (b) initial and optimized hydrofoil shapes (x and y not in scale).

comparison of the initial and optimized hydrofoil shapes, Figure 5.25b, shows that the design variable changes are more pronounced at the trailing edge, displacing it towards the free–stream flow angle. The observed increased sensitivity and displacement of the hydrofoil leading and trailing edges are also reported by Boger & Paterson [36], where a two–phase optimization algorithm is developed using the continuous adjoint method to the barotropic approach and examines similar configurations.

5.3.2.2 Inviscid, Cavitating Flow over the NACA 66(MOD) Hydrofoil – max J_F

The second two–phase ShpO example aims at the maximization of the lift force induced by a hydrofoil at cavitating conditions. Again, the NACA 66(MOD) hydrofoil, previously validated for the said conditions, is used as the baseline shape. The lift force in each design is computed via the projection vector $\hat{\boldsymbol{r}} = \left[-\sin\alpha_\infty \;\; \cos\alpha_\infty\right]^T$ (Eq. (4.60)). The CPs at the leading and trailing edge are fixed to avoid localized changes in the hydrofoil shape that appeared in the previous case. Therefore, the design variables consist of the y–coordinates of the remaining control points defining the hydrofoil shape $(1-7,\; 9-14)$.

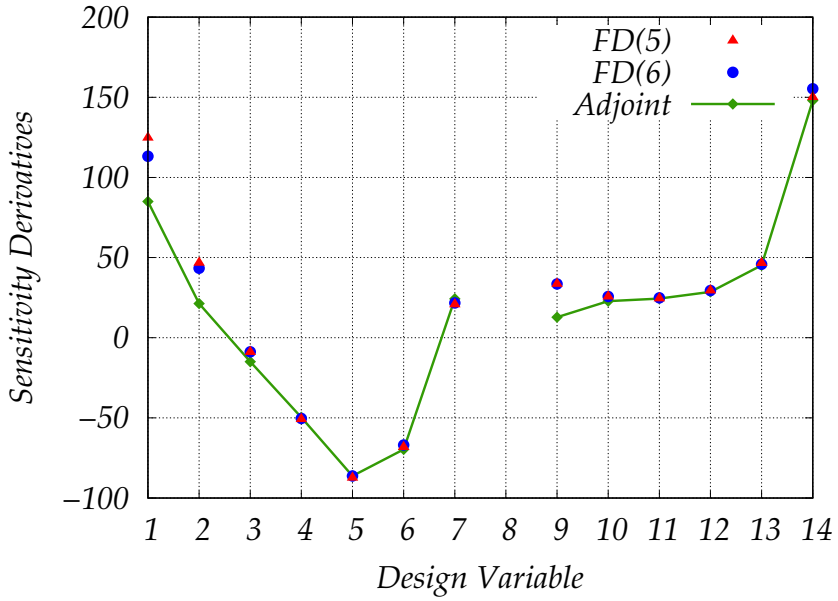


Figure 5.26: ShpO of the NACA 66(MOD) hydrofoil targeting maximization of the lift force exerted on the hydrofoil surface (max J_F), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Comparison of the J_F SDs obtained by the continuous adjoint method and FDs. For reference to the design variable numbering see Figure 5.15b.

Prior to performing the ShpO, the SDs were computed using FDs to compare them with those computed using the continuous adjoint method, Figure 5.26. The adjoint method SDs computed by the adjoint method match the outcome of FDs at all but three design variables defined at control points $(b_1,\; b_2,\; b_9)$. Figure 5.27 presents the adjoint liquid volume fraction ψ_5 for the baseline shape when lift maximization is considered.

The optimization termination criterion is based on a pre–specified number of optimization cycles (40), requiring approximately 30 hours to complete on 48 AMD EPYC 7401 (2.0 Ghz) processors due to the higher number of optimization cycles. The pre–specified number of

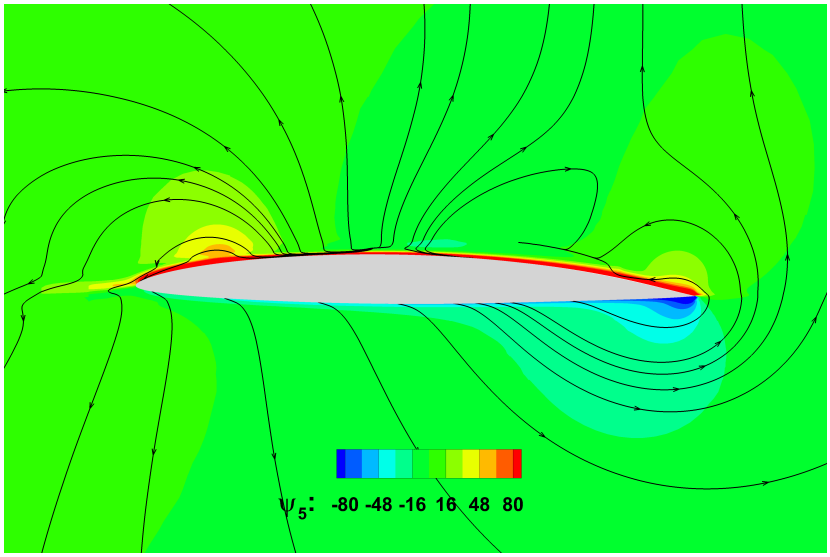


Figure 5.27: ShpO of the NACA 66(MOD) hydrofoil targeting maximization of the lift force exerted on the hydrofoil surface (max J_F), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Initial adjoint liquid volume fraction field ψ_5 for max J_F. Streamlines of the adjoint velocity (ψ_2, ψ_3) are also drawn.

optimization cycles is defined to focus on the validation of the computed SDs and the two–phase optimization algorithm aiming at the maximization of J_F, and to avoid the creation of unrealistic hydrofoil shapes. Alternatively, using constraints, such as a volume equality constraint, would make J_F converge to a specific value but add complexity to the optimization algorithm. The optimization is performed without constraints and leads to a new design that features ∼25% J_F value. The evolution of J_F is shown in Figure 5.28. In the same figure, the vapour presence in the domain (Eq. (4.62)) is also monitored. Initially, it features a slight increase, making the hydrofoil cavitation more intense, while in the following optimization cycles, this is mildly reduced. The initial increase can be explained by observing Figure 5.29a, where the liquid fraction iso–areas are depicted for the optimized shape. Specifically, the local pressure reaches the vapour pressure along the suction side closer to the leading edge, elongating the incepted cavity and is noticeable in Figure 5.30a that compares the initial and optimized computed surface pressure distributions. The reduction in vapour volume fraction could be credited to the increased velocities close to the less campered suction side that convect the cavity further downstream, causing it to condensate more. Overall, a thin cavity is observed that is thickest just before the trailing edge, where the hydrofoil's contour begins to camper.

Based on the above, the comparison of the initial and optimized hydrofoil shapes in Figure 5.30b reveals that the optimized design tends to a more flow-aligned suction side, while the

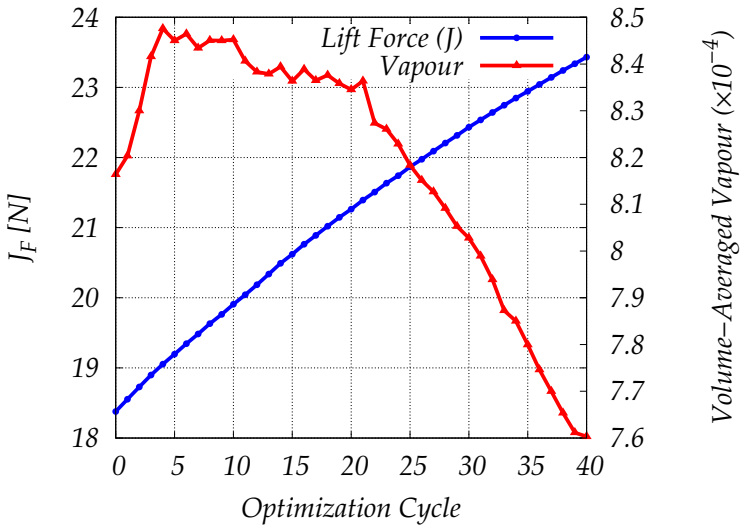
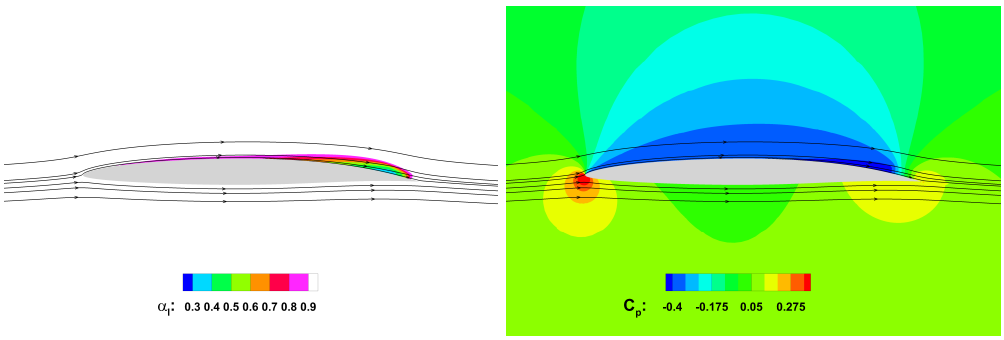


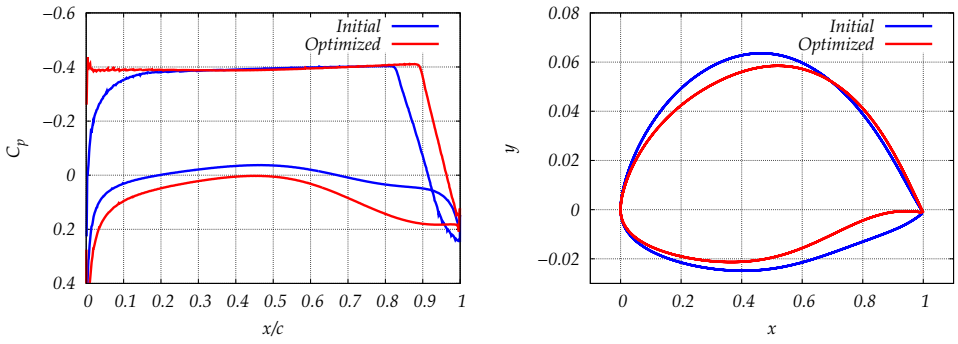
Figure 5.28: ShpO of the NACA 66(MOD) hydrofoil targeting maximization of the lift force exerted on the hydrofoil surface (max J_F), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Evolution of the J_F values through the optimization cycles.

pressure side exhibits the same features arising in single–phase optimizations (cf. Figure 5.13). Note that pressure along the suction side cannot pass the vapour pressure. Thus, once this value is reached along the suction side, changes in J_F are solely caused by modifying the hydrofoil's purely liquid pressure side.



(a) **(b)**

Figure 5.29: ShpO of the NACA 66(MOD) hydrofoil targeting maximization of the lift force exerted on the hydrofoil surface (max J_F), $\alpha_\infty = 1°$, $\sigma = 0.38$ - Iso–areas of (a) liquid fraction and (b) pressure with streamlines over the optimized hydrofoil.



(a) (b)

Figure 5.30: Lift maximization (max J_F) of the NACA 66(MOD) hydrofoil, $\alpha_\infty = 1°$, $\sigma = 0.38$ - Comparison of (a) the computed surface pressure distributions for the (b) initial and optimized hydrofoil shapes (x and y not in scale).

5.3.2.3 Laminar, Cavitating Flow over the NACA 0012 Hydrofoil – min J_V

Next, the accuracy of the adjoint method for the ShpO of a laminar, cavitating flow over the NACA 0012 hydrofoil targeting the minimization of the vapour volume fraction is assessed. Similar to the previous examples, the SDs obtained by the adjoint method are compared with FDs to verify their accuracy in laminar flows and are shown in Figure 5.31. The SDs refer to the $x-$coordinates of the CPs of the suction and the pressure side, followed by the $y-$coordinates in the same order. Good agreement is observed overall, except for the design variables corresponding to control points residing close to the leading edge and the cavity onset area (β_{19}, β_{20}). Figure 5.32 depict the adjoint liquid volume fraction iso–areas computed for J_V.

The ShpO is performed for a total of 30 optimization cycles, requiring approximately 25 hours on 48 AMD EPYC 7401 (2.0 Ghz) processors and leads to a new hydrofoil shape that is barely cavitating at these conditions. The evolution of the J_V through the optimization cycles is presented in Figure 5.33. After 25 cycles, the vapour generated in the new design is negligible and, thereafter, asymptotically tends to a zero value, as in the NACA 66(MOD) hydrofoil case (cf. Figure 5.23). The lift force exerted by the hydrofoil follows the same trend with J_V, implying that the remarks made in the inviscid case may stand true here as well. In Figure 5.35a, where the computed surface pressure distributions of the initial and optimized hydrofoil shapes are shown, reveal the location of lift loss due to the reduction of cavitation. Since the pressure along the pressure side does not practically change, lift loss is caused due to its increase along the suction side as a side–effect of avoiding evaporation. Local pressure

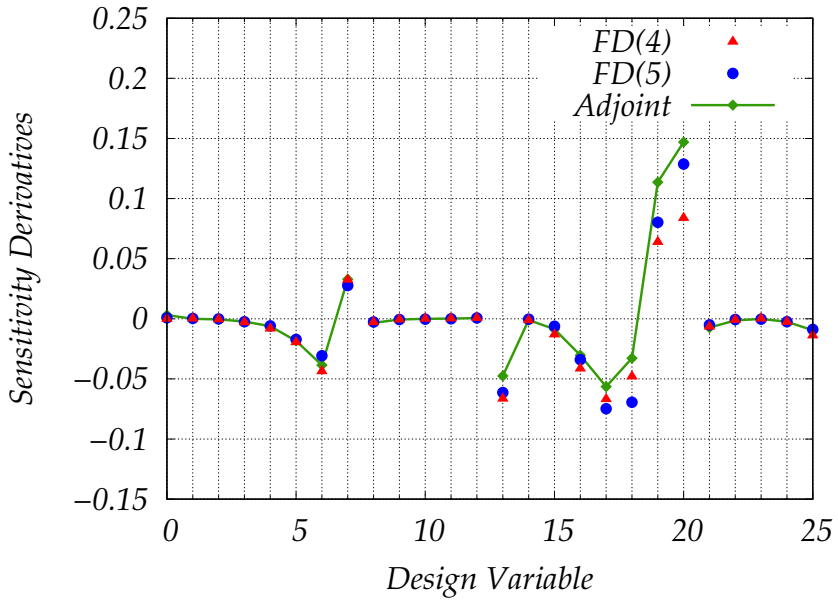


Figure 5.31: ShpO of the NACA 0012 hydrofoil targeting minimization of vapour volume fraction (min J_V), $Re_c = 500$, $\alpha_\infty = 4°$, $\sigma = 0.5$ - Comparison of the J_V SDs obtained by the continuous adjoint method and FDs. For the design variable numbering see Figure 5.15a.

differences between the baseline and optimized hydrofoil occur mainly at the cavity inception and onset area.

Figures 5.34b and 5.34a present the iso–bar areas and liquid volume fraction iso–areas of the optimized hydrofoil shape, respectively. The new hydrofoil is almost cavitation–free and features no recirculation, demonstrating the efficacy of the ShpO algorithm in laminar flows. Furthermore, in Figure 5.35b, a comparison of the initial and optimized hydrofoil shapes is given.

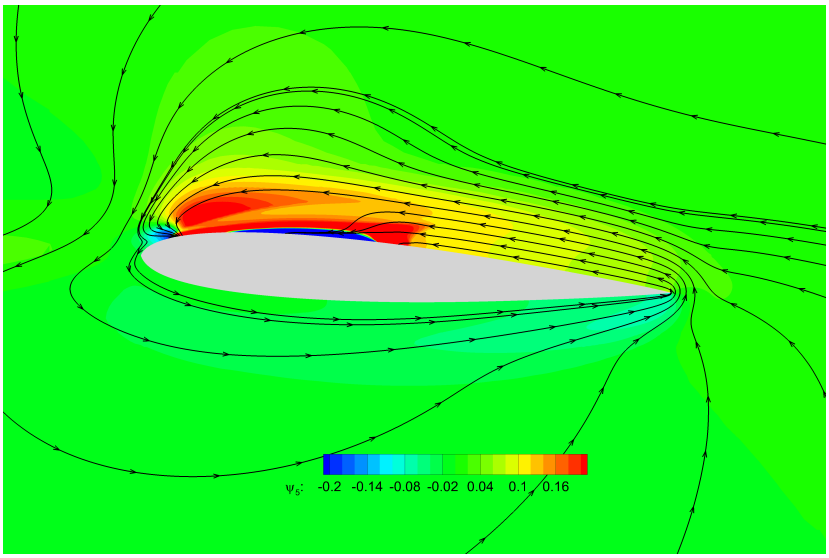


Figure 5.32: ShpO of the NACA 0012 hydrofoil targeting minimization of vapour volume fraction (min J_V), $Re_c = 500$, $\alpha_\infty = 4°$, $\sigma = 0.5$ - Initial adjoint liquid volume fraction field ψ_5 for min J_V. Streamlines of the adjoint velocity (ψ_2, ψ_3) are also drawn.

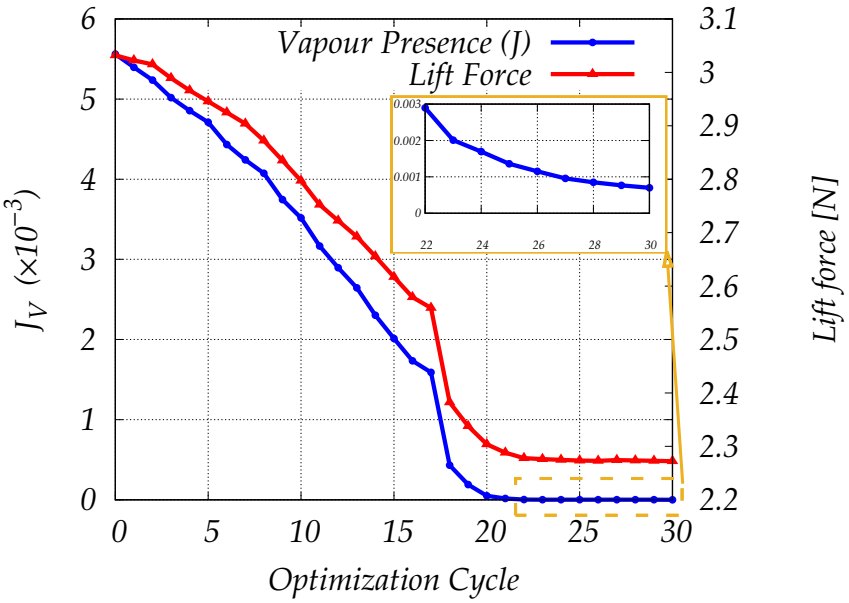
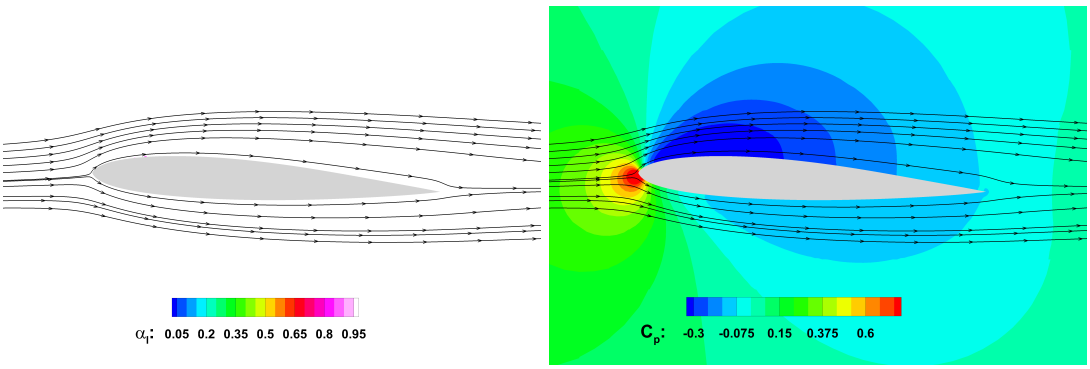
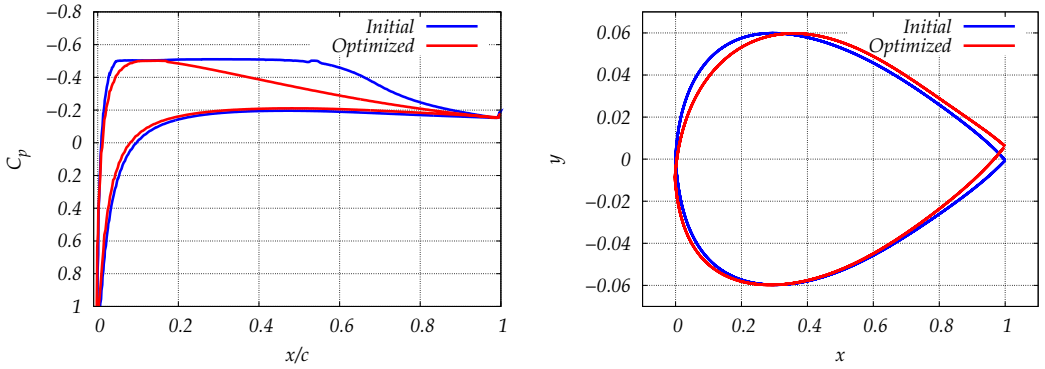


Figure 5.33: ShpO of the NACA 0012 hydrofoil targeting minimization of vapour volume fraction (min J_V), $Re_c = 500$, $\alpha_\infty = 4°$, $\sigma = 0.5$ - Evolution of the J_V values through the optimization cycles, by also monitoring lift force.



(a) **(b)**

Figure 5.34: ShpO of the NACA 0012 hydrofoil targeting minimization of vapour volume fraction (min J_V), $Re_c = 500$, $\alpha_\infty = 4°$, $\sigma = 0.5$ - Iso–areas of (a) liquid volume fraction and (b) pressure with streamlines over the optimized hydrofoil.



(a) **(b)**

Figure 5.35: ShpO of the NACA 0012 hydrofoil targeting minimization of vapour volume fraction (min J_V), $Re_c = 500$, $\alpha_\infty = 4°$, $\sigma = 0.5$ - Comparison of (a) the computed surface pressure distributions for the (b) initial and optimized hydrofoil shapes (x and y not in scale).

5.4 Concluding Remarks

In this chapter, the developed continuous adjoint method presented in Chapter 4 is implemented to support a ShpO algorithm for single– and two–phase flows. The ShpO algorithm combines the merits of the CC method by accounting for the variations of eddy viscosity and the adjoint counterpart of the two–phase formulation (homogeneous mixture model) that also incorporates the effects of cavitating flows, presented for the first time in the literature [266].

The use of the CC method requires special attention when linking the arising intersection points along the CC edges with the discretized body surface nodes but provides a versatile tool that enables large displacements during the optimization. The generated CC meshes, created throughout the optimization, are always valid and share the same quality. Furthermore, the entire optimization framework is fully automated and is limited only by the plasticity of the chosen parameterization, i.e. the range of shapes it can generate.

The significance of avoiding the frozen turbulence assumption by differentiating the turbulence model has been presented several times in the literature. The implemented ShpO algorithm was assessed on both internal and external aerodynamic cases for different objective functions and was able to obtain better–suited designs.

The extension of the optimization algorithm to incorporate cavitation was assessed on isolated hydrofoils and showed the accuracy of the implemented continuous adjoint method w.r.t. reference values computed using FDs. The comparisons show very good agreement but also reveal slight, localized deterioration of the SDs accuracy near the leading and trailing edges of the hydrofoils, as well as the immediate vicinity of the cavity onset area. The former is to be expected and can be attributed to the highly irregular CCs that often appear in those areas and damage the flow solution accuracy. Regarding the latter, these areas appear to have large flow and adjoint liquid volume fraction spatial derivatives (see Figures 5.21a and 5.32) and could potentially benefit from higher, localized mesh refinement. Overall, the optimization algorithm proves to be effective since, in the performed ShpOs, the evolution of the objective functions showed good convergence, optimizing the baseline hydrofoils in all cases. Furthermore, it allows for additional objective functions to be considered and enables the monitoring of relative quantities of interest.

Chapter 6

Topology Optimization using the Cut–Cell Method

In the following chapter, a novel CC–based method for performing TopO presented [267]. The new method builds on the CC method to extract fluid–solid interfaces that are used to create body–conforming CC meshes, overcoming shortcomings of porosity–based TopO methods. To this end, the CC generation algorithm, described in Chapter 2, is modified. In addition, the continuous adjoint method is augmented to facilitate the new design variables. Furthermore, the new TopO algorithm is extended to handle constraints, as they are commonly used in TopO, using the GCMMA [247]. The developed method is assessed via comparisons with the traditional porosity–based TopO method.

Overall, the essence of the CC–based TopO method lies in extracting the fluid–solid interfaces, generating computational meshes in the fluid domain, and solving the governing equations by accurately imposing solid wall boundary conditions, as opposed to other TopO methods [37, 70, 127, 181].

6.1 Topology Optimization Problem Definition

The TopO algorithm is used to compute the optimal connection between fixed inlets (S_I) and outlets (S_O) with fluid paths. A suitable objective function to be minimized that can incorporate the mechanical energy losses due to friction forces on the solid walls and among fluid particles inside the fluid domain is the volume-averaged total pressure losses (J_{P_t}), see Eq. (4.57). Furthermore, as it is common in TopO, a fluid volume constraint g_V should also be satisfied to avoid trivial solutions of enlarged fluid paths. As seen in the ShpO problems in

Sections 5.2.1, 5.2.2, minimizing J_{P_t} in ducted flows tends to expand the fluid paths to reduce friction losses. The volume constraint permits only the most rewarding parts of the design space to remain fluid. The volume in $3D$ (or area in $2D$) occupied by the fluid Ω_f should not exceed a pre–defined volume ratio (V_{tar}) of the computational domain Ω, i.e

$$g_V = \frac{\int_{\Omega_f} d\Omega_f}{\int_{\Omega} d\Omega} - V_{tar} \leq 0 \tag{6.1}$$

Finally, the following TopO examples of this chapter share the same type of boundary conditions, namely a fixed velocity at S_I and pressure at S_O. At the evolving solid walls, the no–slip boundary condition is imposed. The choice of a uniform velocity at S_I was favored because it was found that imposing a total pressure at S_I would sometimes lead to convergence difficulties of the flow solution during the optimization. Due to the rapidly changing fluid domain, the kinematic pressure near the inlets would substantially increase, leading to backflow problems.

6.2 Porosity–based Topology Optimization

Before continuing to the CC–based TopO method, the Standard Porosity (Brinkman) Topology Optimization (SPTopO) is presented to be used as reference when assessing the performance of the developed TopO method. In SPTopO, the governing flow equations arise by extending the incompressible Navier–Stokes equations, Eq. (3.4), with the so–called Brinkman Term, i.e.

$$\mathscr{R}_i^{SPTopO} \coloneqq \mathscr{R}_i + S_i^{BR} = 0 \tag{6.2}$$

with

$$\boldsymbol{S}^{BR} = \alpha_{pen}\alpha_b \begin{bmatrix} 0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{6.3}$$

where $\alpha_b^{\ell} \in [0,1]\,, \ell \in \mathscr{L}$. $\mathscr{L}$ denotes all finite volumes existing in the computational domain and α_b^{ℓ} characterizes the porosity of each finite volume ℓ and α_{pen} is a global (pre–defined) penalization parameter. Solid and fluid regions are approximated based on their permeability, namely impermeable finite volumes (high α_b^{ℓ} values) infer solid, while porous regions (low α_b^{ℓ} values) fluid regions. The penalization parameter α_{pen} is used to amplify the effect of the source term and causes the solid regions to become (almost) impermeable. Its value is defined based on

the non–dimensional Darcy number that express the ratio of viscous to porous friction forces. For a problem with a characteristic length L, an almost impermeable solid region is recovered for $Da = \frac{\nu}{\alpha_{pen} L^2} \leq 10^{-5}$, [179].

Numerically, the introduction of term $\boldsymbol{S}^{BR}$ in Eqs. (6.2) calls for the enforcement of a zero fluid velocity at the solid regions, emulating the absence of a fluid path. In fluid regions, this term is zero. However, fluid–solid interfaces, or else solid walls, are not explicitly defined and, thus, solid wall boundary conditions cannot be imposed. Furthermore, during the SPTopO, α_b^ℓ can also take intermediate values smearing the effect of the source terms across several computational cells, leading to potential inaccuracies in the obtained flow and, subsequently, adjoint solutions.

In the context of SPTopO [68, 127, 128, 179, 191], the design variables are the porosity values of each finite volume and, thus, the outcome of a SPTopO is a $\boldsymbol{\alpha_b}$ distribution that minimizes the objective function. Therefore, the adjoint problem is formulated by differentiating the governing equation w.r.t. α_b^ℓ, instead of $\mathbf{x_s}^m$. The differentiation of Eq. (6.2) is similar to that presented in Chapter 4 with two key differences. First, the porosity field is independent of the computational domain boundaries S and, therefore, boundary terms are obtained only by differentiating the objective function, and second, the differentiation of the Brinkman term is required. Since this term is linear, an adjoint Brinkman term emerges that acts on the adjoint solution in a similar manner, i.e. enforcing a zero adjoint velocity in the solid regions. Thus, the FAEs for the porosity–based approach are

$$\Gamma_{in} \frac{\partial \psi_i}{\partial \tau} - \frac{\partial \psi_n}{\partial x_j} A_{nmj} - \frac{\partial f_{nj}^{V\,\psi}}{\partial x_j} + S_n^\psi = 0 \tag{6.4}$$

with

$$\boldsymbol{S}^\psi = \alpha_{pen} \alpha_b \begin{bmatrix} 0 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} \tag{6.5}$$

Differentiation of the augmented objective function (4.4) w.r.t. the porosity value of each finite volume yields

$$\frac{\delta L}{\delta \alpha_b^\ell} = \frac{\delta J}{\delta {\alpha_b}^\ell} + \frac{\delta \mathscr{R}_n^{SPTopO}}{\delta \alpha_b^\ell} \psi_n \tag{6.6}$$

with

$$\frac{\delta J}{\delta \alpha_b^\ell} = -\int_{S_I} \begin{bmatrix} u_k \hat{n}_k \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \frac{\delta \boldsymbol{U}}{\delta \alpha_b^\ell} dS - \int_{S_O} \begin{bmatrix} u_k \hat{n}_k \\ u_1 u_k \hat{n}_k + p_t\ \hat{n}_1 \\ u_2 u_k \hat{n}_k + p_t\ \hat{n}_2 \\ u_3 u_k \hat{n}_k + p_t\ \hat{n}_3 \end{bmatrix}^T \frac{\delta \boldsymbol{U}}{\delta \alpha_b^\ell} dS \tag{6.7}$$

$$\frac{\delta \mathscr{R}_n^{SPTopO}}{\delta \alpha_b^\ell} \psi_n = \alpha_{pen} u_k^\ell \psi_{k+1}^\ell \Omega^\ell \tag{6.8}$$

Eq. (6.6) contributes to the ABCs at S_I and S_O to eliminate dependencies on $\frac{\delta \boldsymbol{U}}{\delta \alpha_b^\ell}$, while the second term defines the SDs of the SPTopO problem.

Finally, to impose the inequality constraint, the volume/area of the fluid region is approximated based on the volume–averaged sum of the porosity field

$$\Omega_f \approx \int_\Omega (1 - \alpha_b) d\Omega \tag{6.9}$$

The (constant) derivative of the volume constraint, Eq. (6.1), of each finite volume ℓ can be then computed as

$$\frac{\delta g_V}{\delta \alpha_b^\ell} = -\frac{1}{\int_\Omega \Omega} \tag{6.10}$$

At each optimization cycle, the SPTopO flow and adjoint equations, Eqs. (6.2) and (6.4), are solved to compute the objective and constraint function derivatives, Eqs. (6.8) and (6.10). Then, by employing the GCMMA algorithm [246], an optimizer that successively solves a number of approximate convex sub-problems (see Appendix C), the design variables are updated.

6.2.1 Example of SPTopO in a Single Inlet–Single Outlet

This example demonstrates the implemented SPTopO algorithm to find the optimal fluid path in a TopO problem where an inlet channel exists at the west and an outlet channel at the south, shown in Figure 6.1. The objective and constraint functions previously described (Eqs. (4.57),(6.1))are used to define the optimization problem. The optimization is performed in a laminar flow with Reynolds number $Re_W = 200$, based on the inlet width, and $\alpha_{pen} = 100$. This case will subsequently be used as a reference to compare with the new CC–based TopO method. During the optimization, the design variables values are allowed to change only in the highlighted area Ω_D. The area constraint defined requires the fluid paths to encompass only 25% of the highlighted area, ergo $\frac{\Omega_f{}^*}{\Omega_D} = 0.25$. When accounting for the fluid paths of the inlet and outlet channels, it corresponds to $V_{tar} = 0.155$. The optimization stopping criteria were

based on changes in $\boldsymbol{\alpha_b}$, namely $|\Delta\boldsymbol{\alpha_b}|_\infty < 10^{-3}$, and the satisfaction of the area constraint, $g_V \leq 0$.

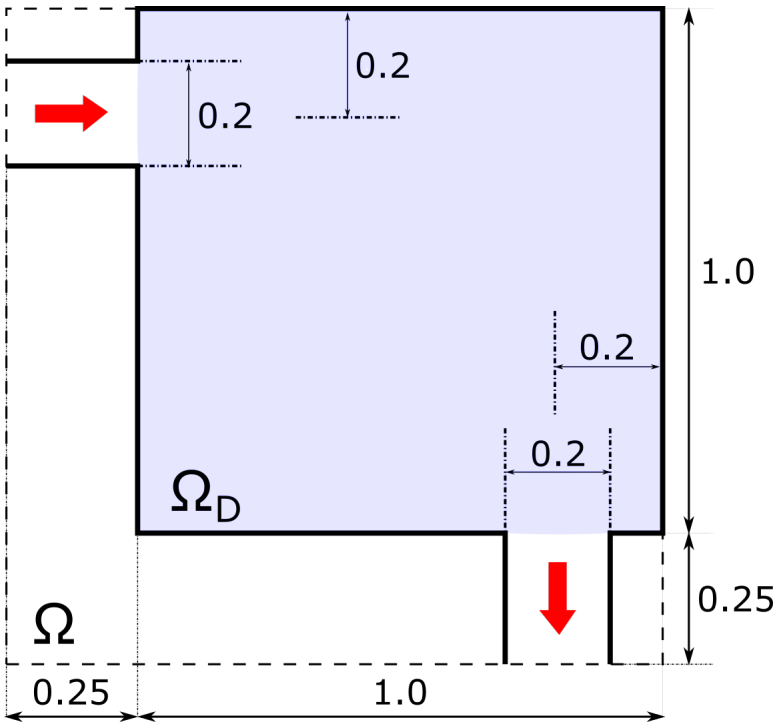


Figure 6.1: SPTopO – Single Inlet–Single Outlet case, $Re_W = 200$ - A fluid path is sought connecting a single inlet with a single outlet. The inlet is positioned at the western side of the domain Ω, while the outlet at its southern side. The highlighted area specifies the domain in which α_b is allowed to change during the optimization and defines the design domain Ω_D. Dimensions are in m.

The TopO was perform on 24 AMD EPYC 7401 (2.0 Ghz) processors and required approximately 130 minutes to complete. Figure 6.2 presents the evolution of the objective and constraint functions. The SPTopO algorithm successfully obtains a feasible solution with $g_V \approx O\left(10^{-8}\right)$ and an objective value of $J^{SPTopO} = 1.603 \times 10^{-4} \frac{W}{m}$.

Starting from a fully fluid design domain, the initial J_{P_t} value is very low since the fluid is rapidly decelerating in the open fluid areas resulting in large recirculating areas of low–velocity fluid that reduce the shear stresses near the solid walls, Figure 6.3a. The obtained solution is infeasible as the area constraint is not satisfied. The fluid areas begin to solidify to satisfy the area constraint, i.e. have their α_b values increased (Figure 6.3b), and a duct with almost impermeable solid regions comes up. In Figure 6.4, the velocity vectors of an intermediate (infeasible) solution are depicted. Even though no boundary conditions are imposed between

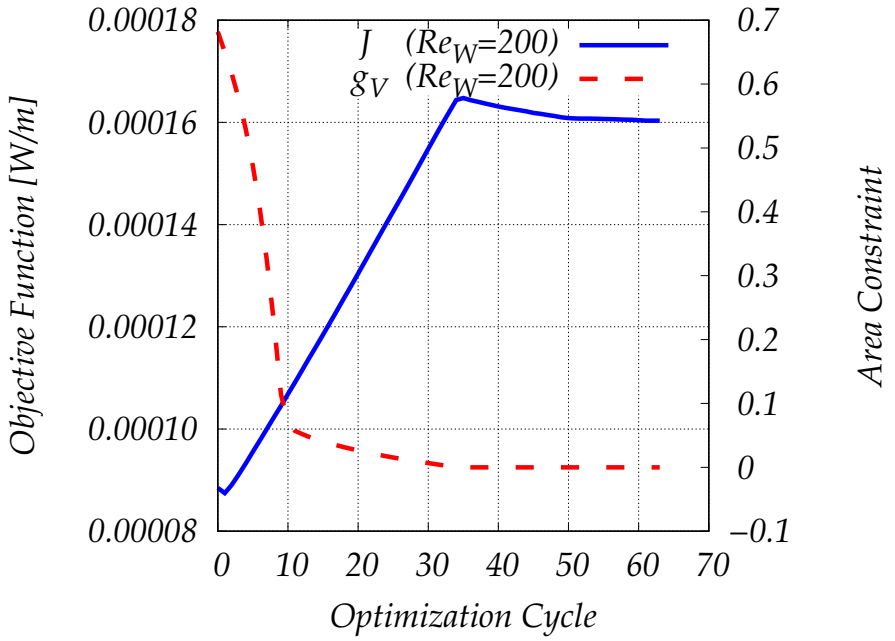
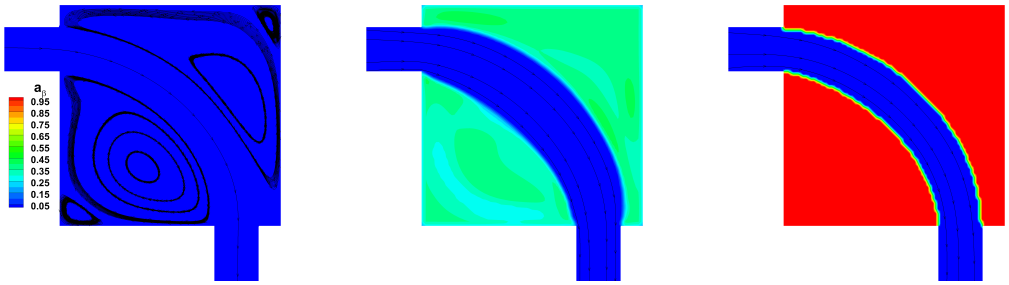


Figure 6.2: SPTopO – Single Inlet–Single Outlet case - Evolution of objective and constraint functions converging to $J^{SPTopO} = 1.603 \times 10^{-4} \frac{W}{m}$ and $g_V \approx O\left(10^{-8}\right)$ for $Re_W = 200$.



(a) **(b)** **(c)**

Figure 6.3: SPTopO – Single Inlet–Single Outlet case, $Re_W = 200$ - Iso–areas of α_b for different optimization cycles. Plots correspond to (a) initial, (b) cycle 8, and (c) final cycle.

the fluid and solid regions, the α_{pen} value is sufficient to approximate the solid wall effects.

Overall, the principal advantages of SPTopO lie in its minor implementation burden, owing to the independence of α_b to geometric quantities that simplifies the adjoint problem formulation and its ability to rapidly change the design domain. The latter can also be seen in Figure 6.3b where α_b is changed throughout the design domain due to its ability to extract SDs contributions from each finite volume.

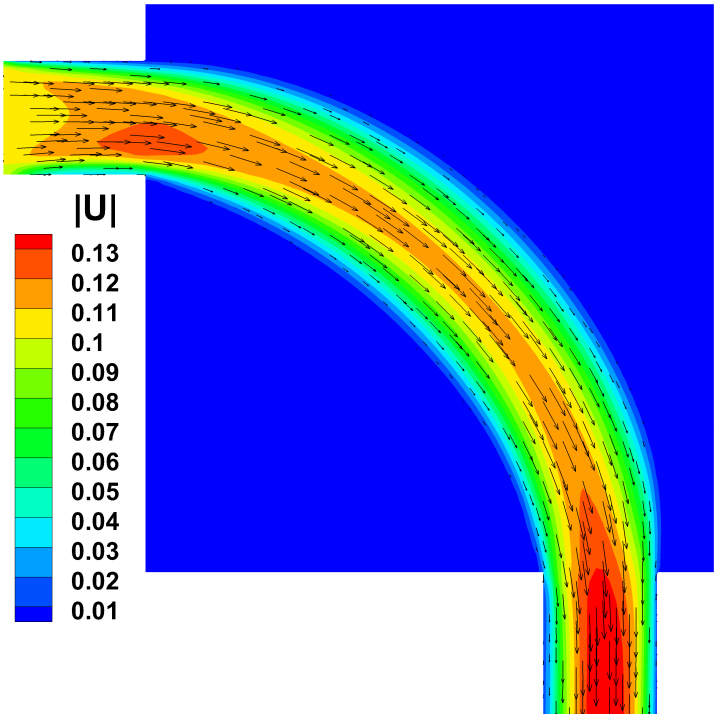


Figure 6.4: SPTopO – Single Inlet–Single Outlet case, $Re_W = 200$ - Velocity magnitude iso–areas of an intermediate solution. At the solidified areas, a minute velocity exists showing the effectiveness of the method in simulating solid regions.

6.2.1.1 Motivation – A Closer Look at the SPTopO Main Weakness

The most noticeable weakness of SPTopO is its inability to impose (exact) boundary conditions along the sought solid walls. This is well–known in the literature and multiple studies were conducted investigating its effect on the obtained flow solution accuracy compared to those on body-fitted meshes. For example, in Kreissl *et al.* [133], the SPTopO method is used in unsteady flows revealing that errors in the pressure fields over the solid areas of Ω can arise when insufficiently refined meshes are used that can significantly affect the TopO path. Also, in Kreissl & Maute [131], where a U–bend duct is optimized using a SPTopO approach, it is shown that the narrow wall between the incoming and outgoing flow leads to non–physical designs. They illustrate, through a parametric study, the difficulty of the Brinkman method to prevent flow from passing through the said narrow wall. Furthermore, they also note that spurious pressure diffusion is observed, resulting in erroneous objective function computations. In Koch *et al.* [127], the transition from an optimized porosity field using SPTopO to ShpO is presented that also illustrated differences in the computed objective function values when the

solution is re–evaluated using body–fitted meshes.

A comparative study of the accuracy of the obtained flow solutions using the porosity–(Eq. (6.2)) and the CC–based (Eq. (3.4)) flow solvers is performed to showcase this weakness of the porosity–based approach. With that intention, the incompressible laminar channel flow with two identical rectilinear internal bodies is considered, Figure 6.5. Furthermore, two slightly different cases are examined to investigate the porosity–based flow solver dependency upon α_{pen} and the effect of intermediate α_b values. In the first case, the internal bodies are positioned having their boundaries coinciding with the Cartesian mesh lines, resulting in a binary $\boldsymbol{\alpha_b}$ distribution, whereas, in the second case, the bodies are shifted by half the size of the cell edge in both dimensions so that some finite volumes take on intermediate α_b values, Figures 6.6a and 6.6b.

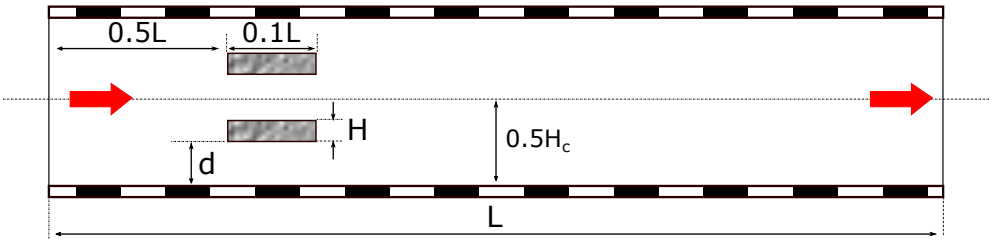


Figure 6.5: Comparative accuracy study between the porosity–based and CC method - Channel flow with two rectilinear internal bodies problem description.

A steady–state flow solution is obtained for Reynolds number $Re_H = \frac{u_m\, H}{\nu} = 50$, blockage ratio $r = \frac{H}{H_c} = 0.125$ and gap parameter $\gamma = \frac{d}{H} = 2.0$ and 2.04 for cases (i) and (ii), respectively [1]. u_m is the velocity at the inlet, H_c the height of the channel, H the height of the bodies, and d the distance of the solid bodies from the closest channel wall. The comparative study includes four simulations using the porosity–based flow solver with different α_{pen} values and a single solution using the CC flow solver for each case.

The computed flow fields are compared in terms of volume-averaged total pressure losses J_{P_t} for both flow solvers, shown in Figure 6.7. The value of α_{pen} greatly affects the obtained flow fields, as small penalization values lead to a porous body failing to enforce zero velocities in its interior, damaging the accuracy of flow predictions. On the other hand, penalization cannot be excessively high as it might cause convergence issues or stall the TopO algorithm [131]. Furthermore, from Figure 6.7, one may conclude that for the same α_{pen} values, the SPTopO flow solutions in cases (i) and (ii) have noticeable differences in the computed quantity of interest, contrary to the CC–based flow solutions that show an expected minute difference.

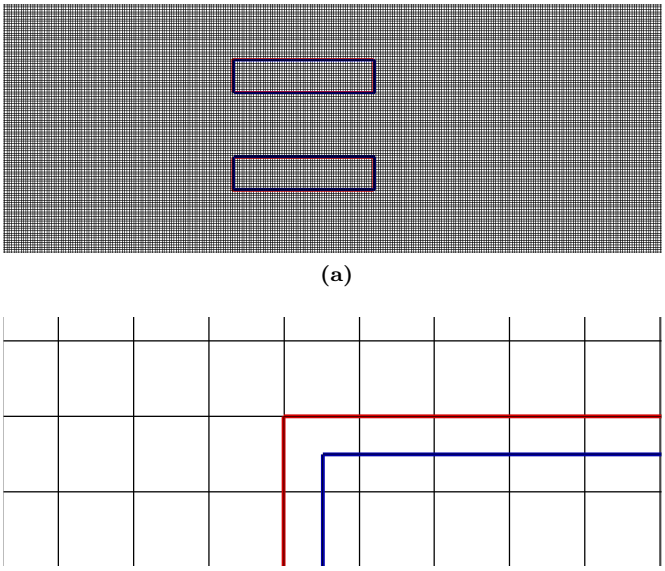


Figure 6.6: Comparative accuracy study between the porosity–based and CC method - (a) Close-up view of the computational domain and generated mesh. (b) Positioning detail for case (i) in which the body contour (red line) coincides with mesh lines to give an ideal 0/1 porosity field and the shifted contour (blue line) for case (ii) in which the porosity at some finite volumes takes on intermediate values.

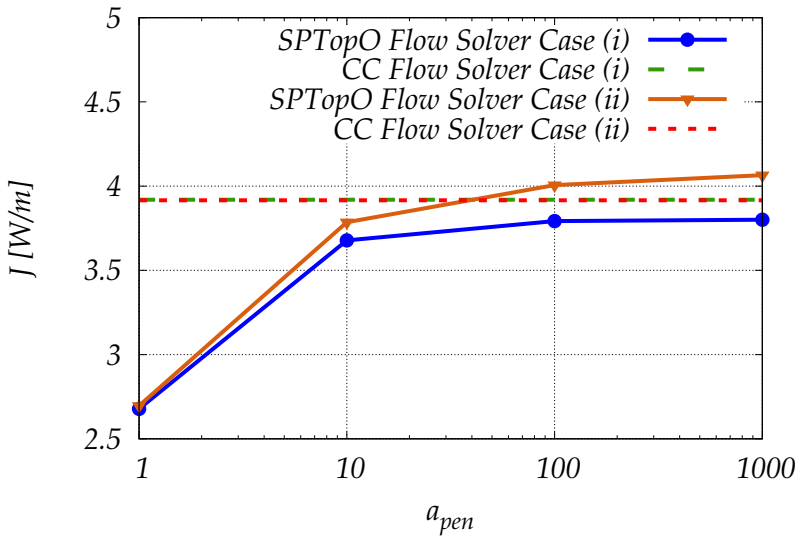
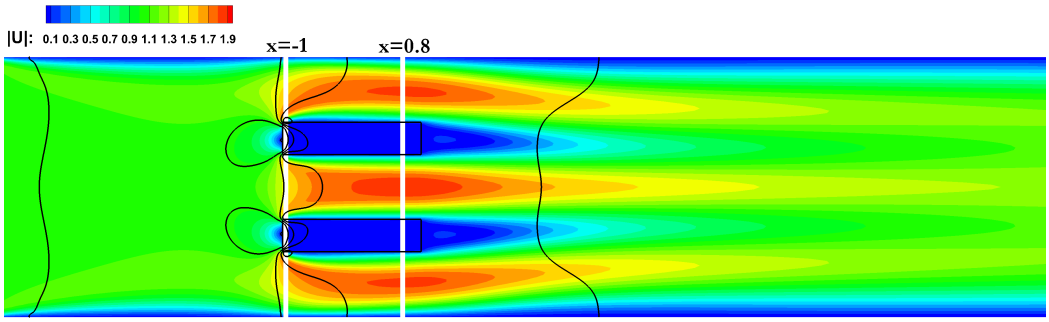
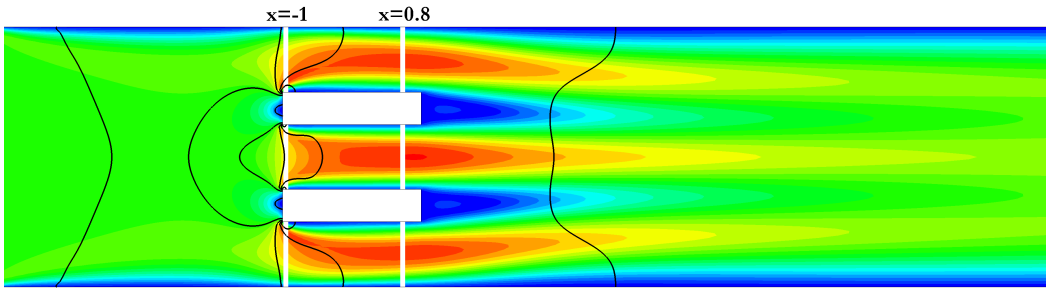


Figure 6.7: Comparative accuracy study between the porosity–based and CC method - Comparison of the computed J_{P_t} values with the two flow solvers for an incompressible, laminar flow inside a straight channel with two rectilinear internal bodies, $Re_H = \frac{u_m H}{\nu} = 50$. The accuracy of the porosity–based flow solver seems to be significantly influenced by the specified α_{pen} values, as well as the positioning of the internal bodies.

The porosity–based solver flow fields with $\alpha_{pen} = 1000$ in cases (i) and (ii) are further compared with the CC–based flow fields by extracting transversal pressure profiles at predefined locations. Starting with case (i), Figures 6.8a and 6.8b show the velocity magnitude iso–areas along with pressure iso–lines, while in Figures 6.9a and 6.9b the two pressure profiles at $x = -1m$ and $x = 0.8m$ are plotted. At $x = -1m$, Figure 6.9a, the pressure profiles are comparable, despite some minor differences near the bodies. At $x = 0.8m$, Figure 6.9b, the porosity–based flow solver consistently over–estimates the pressure, compared to the CC–based flow solver. The same pressure profiles are also shown for case (ii) in Figures 6.9c and 6.9d. With the introduction of intermediate porosity values, differences from the CC–based flow solution at $x = -1m$ become pronounced and are not isolated near the bodies anymore. At $x = 0.8m$, pressure differences are smaller but the pressure transversal derivatives are different.



(a)



(b)

Figure 6.8: Comparative accuracy study between the porosity–based and CC method - Velocity magnitude iso-areas along with pressure iso-lines obtained using the (a) psorosity ($\alpha_{pen} = 1000$) and (b) CC–based flow solvers for the aligned case (i).

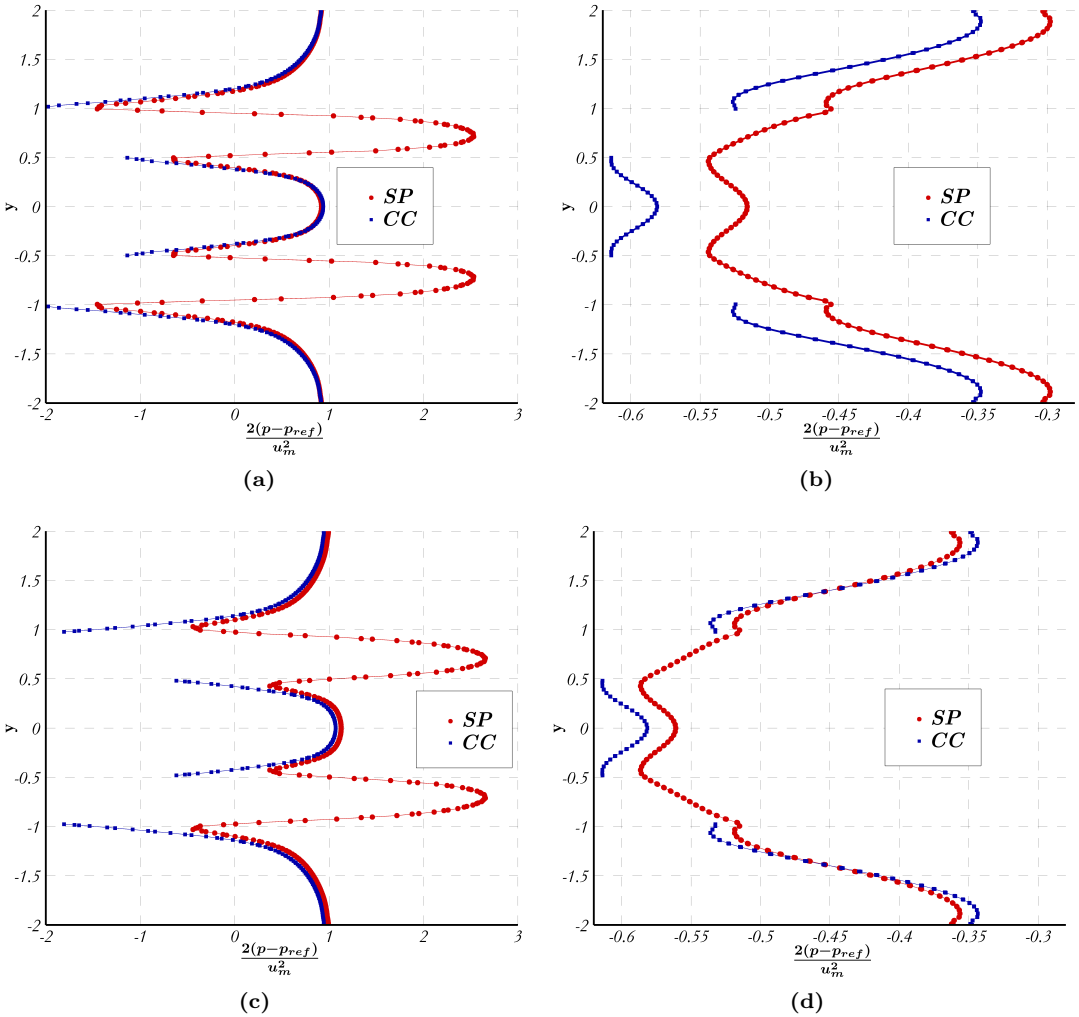


Figure 6.9: Comparative accuracy study between the porosity–based and CC method - Transversal pressure profiles at (a),(c) $x = -1m$ and (b), (d) $x = 0.8m$ for case (i) (top) and case (ii) (bottom). p_{ref} is the reference pressure defined at the outlet. Cross–sections, where pressure profiles are plotted, are shown by white vertical lines in Figures 6.8a and 6.8b.The porosity–based solution corresponds to $\alpha_{pen} = 1000$.

6.3 The Cut–Cell TopO Algorithm and its Steps

To alleviate the aforementioned weaknesses of the porosity–based approach, a new TopO method, with the CC flow solver at its core, is implemented wherein the solid walls are explicitly defined and allows for exact solid boundary conditions to be imposed. As such, the governing equations Eq. (3.4) are solved only in the evolving fluid domain, avoiding the introduction of the Brinkman term and the pertinent assumption of a porous material. To that end, a method of extracting the fluid domain in each optimization cycle is introduced, in which a CC mesh is generated.

In the Cut–cell TopO algorithm, henceforth referred to as Cut–Cell Topology Optimization (CCTopO), a bounded material field $0 \leq \alpha^m \leq 1,\ \forall m = 1, ..., M$, that exists on the M nodes of a background Cartesian mesh and serves as the design variables of the optimization problem, is introduced. This material field $\boldsymbol{\alpha}$ is an auxiliary field* that, before each flow solution, is mapped onto boundary indicator variables $\boldsymbol{\phi}$ to track the fluid–solid interfaces, compute their intersections with the background mesh over Ω and, create the CCs. Therefore, flow and adjoint equations are solved on the CC mesh with clearly defined solid walls, formed by CCs and Cartesian cells entirely residing in the fluid domain as shown in Figure 6.10. In contrast to SPTopO, flow and adjoint equations need to be satisfied only in the fluid domain Ω_f. Furthermore, due to dependencies of the solid walls to the design variables $\boldsymbol{\alpha}$, the adjoint problem is formulated by differentiating J_{P_t} w.r.t. the coordinates of points on the solid boundaries of the CCs ($\mathbf{x_s}^m$), similar to Section 5.1. The principal difference is that $\mathscr{G} = \emptyset$ since no body surface is defined. The adjoint solution provides a sensitivity map on the currently defined solid walls, and the $\boldsymbol{\alpha}$ field is updated accordingly. The background Cartesian mesh is used for both storing $\boldsymbol{\alpha}$ and creating the CCs for the following flow and adjoint equations solution. The constituents of the CCTopO method are analyzed, one–by–one, below.

6.3.1 Computation of the Boundary Indicator Field ϕ

The nodal material field $\boldsymbol{\alpha}$ is mapped onto the boundary indicator variables $\boldsymbol{\phi}$ using a two–step approach, based on filtering techniques commonly used in TopO [232, 271], instead of solving a HJE to displace/march the zero–ϕ interface. This approach circumvents several challenges of the HJE such as the time–consuming re–initialization step of the Level–Set function to Signed

*The physical meaning of the auxiliary field $\boldsymbol{\alpha}$ arises as a result of the determined mapping function. The mapping function used herein suggests that low and high α values tend to indicate fluid and solid nodes, respectively

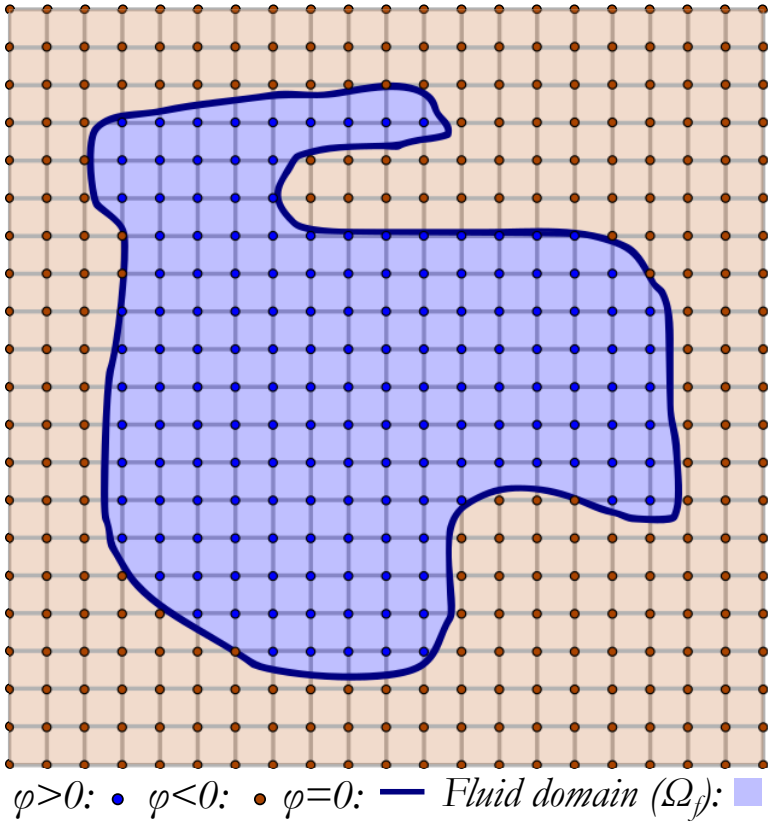


Figure 6.10: Extraction of the fluid domain in CCTopO - Based on the nodal ϕ^m values the fluid domain, along with the fluid–solid interfaces, can be extracted prior to the flow solution. The fluid–solid interfaces coincide with the zero–ϕ iso–line.

Distance function required after several timesteps [43] and the restrictive CFL condition used when solving the HJE, that can lead to slow convergence of the optimization problem. In addition, one advantage of the decided approach is that it offers provides increased flexibility when defining the optimization problem and the design variables, since new ones can easily be included or excluded [264].

In the first step, α^m is projected onto a smoothed intermediate variable, $\tilde{\alpha}^m$, [41, 233],

$$\tilde{\alpha}^m = \frac{\sum\limits_n \alpha^n w^n}{\sum\limits_n w^n}, \qquad w^n = r - \mathrm{d}(n,m) \tag{6.11}$$

where the summation is performed over all nodes lying within an *r–sphere* with radius $r = d^r h$, $w^n \geq 0$, centered at node m. d^r is a user-defined multiplier and h is the length of a Cartesian cell edge (Eq. (2.4)). These can easily be found by traversing the generated Octree data structure

in Section 2.1.1. The purpose of this step is to accelerate the convergence of the optimization by increasing the zone of influence of the design variables, to avoid checkerboard patterns in the design variables, as well as to avoid the generation of crude solid boundaries [232, 259, 264].

Following, the second step is performed in which the smoothed variable $\tilde{\alpha}^m$ is steepened using an approximate Heaviside function, [233],

$$\phi^m = \frac{1}{2} - \frac{\tanh(0.5\beta) + \tanh(\beta(\tilde{\alpha}^m - 0.5))}{2\tanh(0.5\beta)} \tag{6.12}$$

centered at $\tilde{\alpha}^m = 0.5$, and mapped onto the boundary indicator field, such that $\phi^m > 0$ refers to fluid, $\phi^m < 0$ to solid, and $\phi^m = 0$ to the fluid–solid interface. The performed studies showed that $\beta = 12$ and $3 \leq d^r \leq 6$ lead to sufficiently smooth boundaries and facilitate the convergence of the optimization [16, 132].

6.3.2 Generation of Cut–Cells based on ϕ

Having obtained the nodal ϕ^m values, intersection points of the zero–ϕ contour can be computed along edges of the Cartesian cell. These are located on edges whose nodes have ϕ values with opposite signs. The CCs are generated by computing these intersection points and connecting them with the Cartesian cell nodes that correspond to the fluid region (have $\phi^m > 0$), similarly to the marching cubes algorithm [150] but with interpolations between the cell vertices. The necessary geometric quantities of CCs, such as cell volume, surface areas and unit vectors normal to faces, can then be computed based on the points that surround the fluid part inside the Cartesian cell, as described in Chapter 2. Based on the above, the coordinates of the intersection points $\mathbf{x_s}^n$ and their derivatives w.r.t. ϕ^m are computed as

$$\mathbf{x_s}^n = \mathbf{x}^{A^{(n)}} - \frac{\mathbf{x}^{B^{(n)}} - \mathbf{x}^{A^{(n)}}}{\phi^{B^{(n)}} - \phi^{A^{(n)}}}\phi^{A^{(n)}} \tag{6.13}$$

$$\frac{\partial \mathbf{x_s}^n}{\partial \phi^i} = \frac{\mathbf{x}^{B^{(n)}} - \mathbf{x}^{A^{(n)}}}{\left(\phi^{B^{(n)}} - \phi^{A^{(n)}}\right)^2}\left(\phi^{A^{(n)}}\frac{\partial \phi^{B^{(n)}}}{\partial \phi^i} - \phi^{B^{(n)}}\frac{\partial \phi^{A^{(n)}}}{\partial \phi^i}\right), \qquad i = A^{(n)}, B^{(n)} \tag{6.14}$$

where $A^{(n)}$ and $B^{(n)}$ are the two mesh nodes used to compute the n^{th} intersection point, connected by an edge with $\phi^{A^{(n)}}\phi^{B^{(n)}} \leq 0$. In Figure 6.11, an illustrative $2D$ example is given where a triangular CC is created based on the given ϕ values of the Cartesian cell nodes.

The above–mentioned can be easily extended to three dimensions, Figure 6.12a. However, in $3D$, Cartesian cells having more than 3 intersection points along their edges occur frequently. Steiner points are introduced at the arithmetic mean of the computed intersection points inside

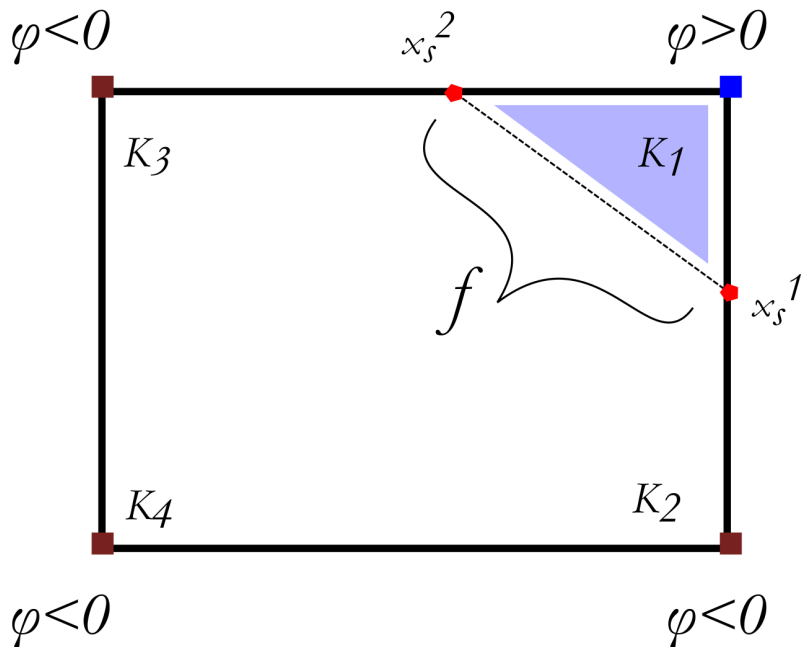


Figure 6.11: CC creation based on nodal ϕ values - $2D$ Cartesian cell with ϕ values on its nodes, (K_1, K_2, K_3, K_4). The opposite ϕ signs at K_1, K_2, and K_1, K_3 lead to the zero-ϕ intersection points $\mathbf{x_s}^1$ and $\mathbf{x_s}^2$. The resulting CC is a triangle with its vertices at $K_1, \mathbf{x_s}^1$ and $\mathbf{x_s}^2$.

the Cartesian cell to avoid the creation of faces defined by points that are not co–planar on the solid boundary that could compromise the solution accuracy. These additional points are then connected with the computed intersection points along the cell edges to triangulate the solid boundary, Figure 6.12b. In case a Cartesian cell is intersected twice by the solid boundary, double–cut cells are formed, Figure 6.12c; for them, the intersection points are split into two clusters, and a Steiner point is introduced for each cluster. To identify them, an attempt to edge–connect the Cartesian cell solid nodes (ϕ <0) is made during the CC generation process. Finally, contrary to the standard CC generation method for a given body surface (Chapter 2) CCs intersected by the solid boundary more than twice cannot exist, as in such a case, the solid boundaries would have merged.

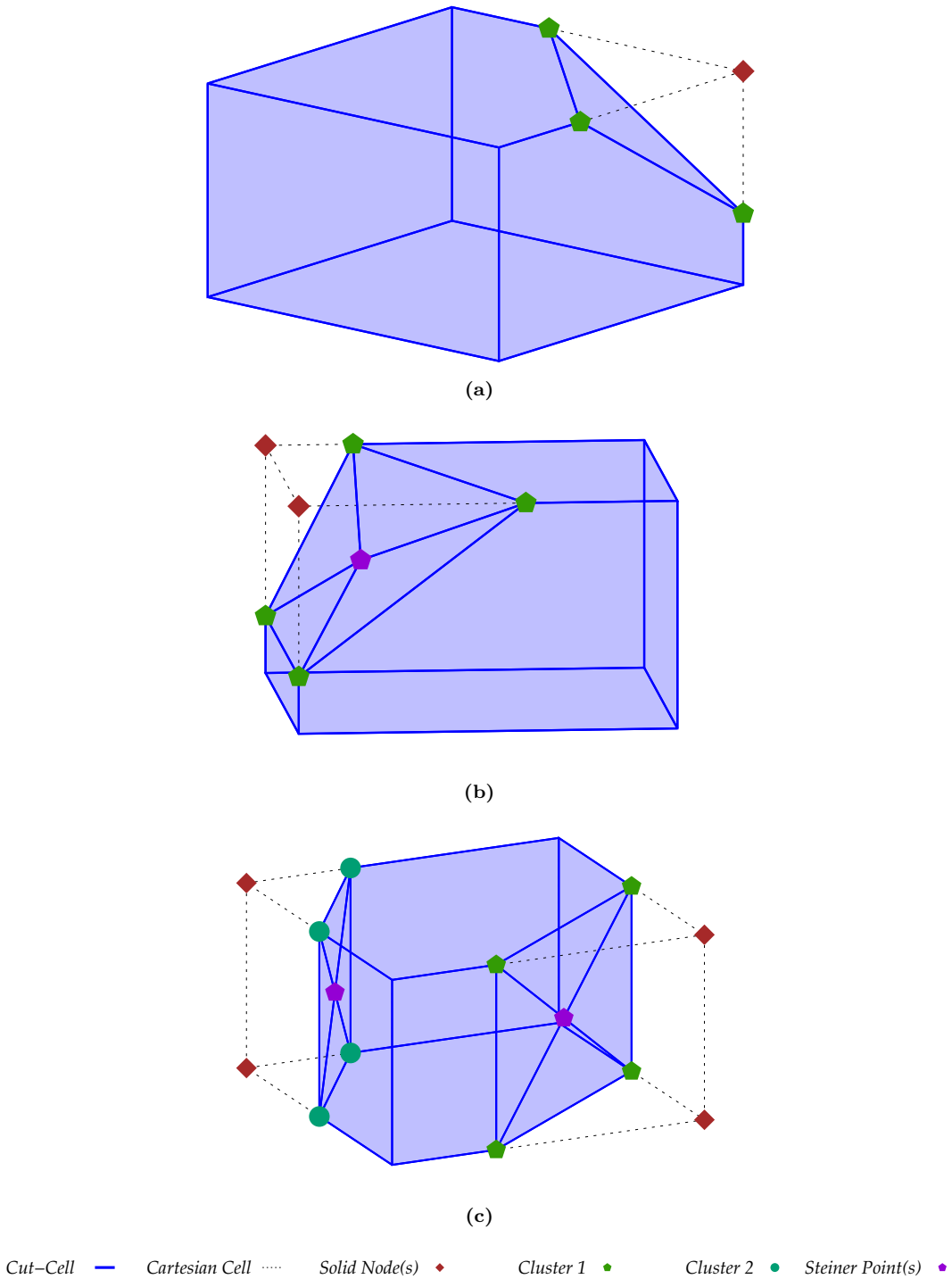


Figure 6.12: Indicative examples of generated $3D$ CCs for which the volume encompassed by the blue lines is considered fluid - In (a), no additional point is introduced since the intersection points form a triangular solid face. In (b), having 4 non–co–planar intersection points, a Steiner point is introduced to triangulate the solid boundary into 4 solid faces leading to a CC with 10 faces in total. In (c), a double–cut cell is identified since the solid nodes of the Cartesian cell cannot be edge–connected. The intersection points split into two separate clusters with two Steiner points, and a CC with 14 faces in total is created.

In practice, the resulting CCs can be very small, and this may negatively impact stability. Thus, the cell merging algorithm, overviewed in Chapter 2 is also used to merge small CCs with a neighbouring cell, Figure 6.13. Again, connectivity information and geometric quantities pass on to the merged supercell that replaces the constituent cells.

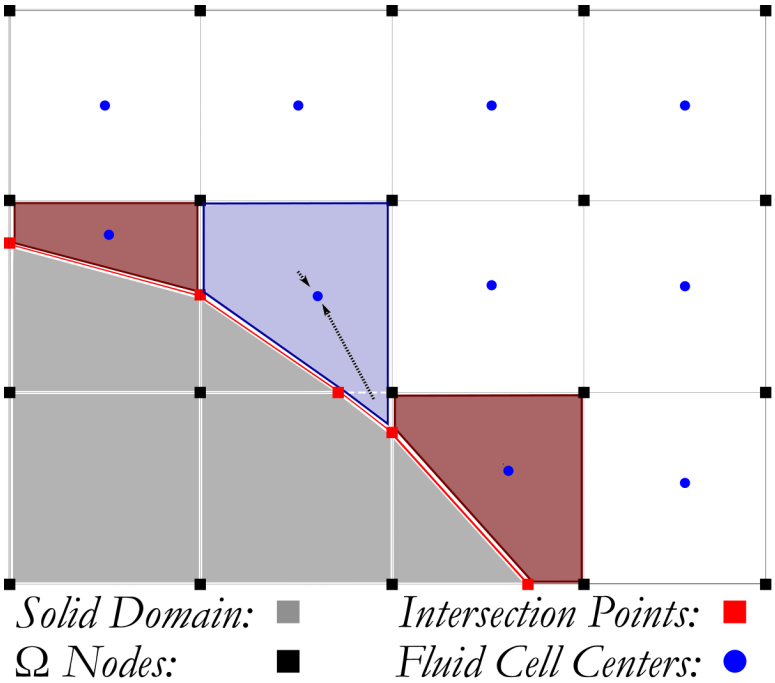


Figure 6.13: $2D$ schematic of the near–wall topology used in the geometry–free mesh generation process - Part of the solid boundary, the intersection points, some non–merged CCs (brown), and a merged CC (ciel). Black filled squares stand for the Cartesian mesh nodes where the $\boldsymbol{\alpha}$ and $\boldsymbol{\phi}$ field is stored, whereas all flow and adjoint variables are stored at cell centers (blue filled circles).

6.3.3 Governing Equations

The CCTopO algorithm is used on single–phase, laminar flows of a incompressible fluid. Therefore, the Navier–Stokes equations (Eq. (3.4)) are solved in Ω_f, as defined by the available α^m field, in each optimization cycle. For clarity, these are repeated below

$$\mathscr{R}_i := \Gamma_{in}\frac{\partial U_n}{\partial \tau} + \frac{\partial f_{ij}^I}{\partial x_j} - \frac{\partial f_{ij}^V}{\partial x_j} = 0 \tag{6.15}$$

6.3.4 Formulation of the Adjoint Problem

The adjoint method is employed to compute a sensitivity map, i.e. the derivative of the objective function on the solid boundary points $\frac{\delta J_{P_t}}{\delta \mathbf{x_s}^n}$, by solving the appropriate FAEs Eq. (4.25) in the

fluid domain Ω_f and imposing the ABCs along its boundaries, as derived in Section 4.2.5. Due to the sole dependency of the solid boundary points w.r.t. the boundary indicator field Eq. (6.13), the chain rule is used to obtain the derivatives of the objective function w.r.t. the boundary indicator variables $\frac{\delta J_{P_t}}{\delta \phi}$ and, then, considering Eqs. (6.12) and (6.11) w.r.t. the α^m field. Thus, in the CCTopO method, a sensitivity map is computed on the reconstructed, by the CC method, solid surfaces and, then, transformed to derivatives of J_{P_t} w.r.t. the α^m field, i.e. the design variables that implicitly describe the solid surfaces.

The ABCs imposed make $\frac{\delta J_{P_t}}{\delta \mathbf{x_s}^n}$ independent of the derivatives of the flow variables w.r.t. $\mathbf{x_s}^n$ along S_I, S_O and S_W, as overviewed in Chapter 4. The differentiation of J_{P_t} introduces additional terms along S_I and S_O, 4.3.1. Based on the aforementioned imposed flow boundary conditions, these are

$$\frac{\delta J_{P_t}}{\delta \mathbf{x_s}^n} = -\int_{S_I} \begin{bmatrix} u_n \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \frac{\delta \boldsymbol{U}}{\delta \mathbf{x_s}^n} dS - \int_{S_O} \begin{bmatrix} u_n \\ u_1 u_n + p_t \; n_1 \\ u_2 u_n + p_t \; n_2 \\ u_3 u_n + p_t \; n_3 \end{bmatrix}^T \frac{\delta \boldsymbol{U}}{\delta \mathbf{x_s}^n} dS \tag{6.16}$$

By satisfying the adjoint boundary conditions, the sensitivity map is obtained using

$$\frac{\delta L}{\delta \mathbf{x_s}^n} = -\int_{S_w} \left(\psi_1 \; n_k + \tau_{ik}^{\psi} \; n_i \right) \frac{\partial u_j}{\partial x_m} \frac{\delta x_m}{\delta \mathbf{x_s}^n} dS \tag{6.17}$$

Then, the computation of the gradient of J_{P_t} w.r.t. α^m requires the use of the chain rule to transform the sensitivity map computed at the solid boundary points $\mathbf{x_s}^n$ to the expression of SDs at the mesh nodes of Ω, as follows

$$\frac{\delta J_{P_t}}{\delta \alpha^m} = \sum_l \sum_n \frac{\delta L}{\delta \mathbf{x_s}^n} \frac{\partial \mathbf{x_s}^n}{\partial \phi^l} \frac{\partial \phi^l}{\partial \tilde{\alpha}^l} \frac{\partial \tilde{\alpha}^l}{\partial \alpha^m} \tag{6.18}$$

where the first summation (l) considers all nodes within the selected smoothing radius according to Eq. (6.11) and the second (n) all solid boundary points. Term $\frac{\partial \mathbf{x_s}^n}{\partial \phi^l}$ is given in Eq. (6.14), while terms $\frac{\partial \phi^l}{\partial \tilde{\alpha}^l}$ and $\frac{\partial \tilde{\alpha}^l}{\partial \alpha^m}$ are computed as

$$\frac{\partial \phi^l}{\partial \tilde{\alpha}^l} = -\frac{\beta}{2} \frac{\left(1 - \tanh^2\left(\beta \left(\tilde{\alpha}^l - 0.5\right)\right)\right)}{\tanh\left(0.5\beta\right)} \tag{6.19}$$

$$\frac{\partial \tilde{\alpha}^l}{\partial \alpha^m} = \frac{w^l}{\sum_n w^n} \tag{6.20}$$

Similarly to the SPTopO algorithm, the differentiation of the constraint function is also necessary. The derivatives of the fluid volume of CCs w.r.t. $\mathbf{x_s}^n$ can be computed analytically,

resulting in the corresponding derivatives of the constraint function

$$\frac{\delta g_V}{\delta \mathbf{x_s}^n} = \frac{1}{\int_\Omega d\Omega} \sum_l \frac{\delta \Omega_l^c}{\delta \mathbf{x_s}^n} \tag{6.21}$$

Summation is performed over the CCs of which point n lies on their solid boundary face; $\frac{\delta \Omega_l^c}{\delta \mathbf{x_s}^n}$ is the derivative of the l^{th} CC volume w.r.t. the coordinates of boundary point n, computed using expression Eq. (5.2). Note that, in contrast to the SPTopO, the volume constraint is not constant since the fluid volume of the CCs changes during the optimization process. Geometric derivatives $\frac{\delta x_k}{\delta \mathbf{x_s}^n}$ and $\frac{\delta \Omega^c}{\delta \mathbf{x_s}^n}$, required by Eqs. (6.17) and (6.21), are again computed as closed–form expressions (Section 5.1), [174, 214, 266].

6.3.5 Overview of the Cut–Cell Topology Optimization Algorithm

All constrained optimization problems are solved using the GCMMA algorithm [246] (Appendix C). The termination criterion is based on the maximum change in $\boldsymbol{\phi}$ optimization cycle, namely $\sqrt{\Delta \boldsymbol{\phi}}_\infty < 10^{-3}$, and the requirement of a feasible solution, $g_V \leq 0$.

The developed CCTopO algorithm can be summarized in the following steps:

1. Initialize the nodal material field α^m on the fixed background mesh Ω.
2. Map α^m onto boundary indicator variables ϕ^m, using Eqs. (6.11) and (6.12).
3. Compute coordinates of intersection points $\mathbf{x_s}^n$ between the Cartesian cell edges, and the zero–ϕ iso–line using Eq. (6.13) and create CCs corresponding to the body–fitted mesh in Ω_f.
4. Solve the flow and adjoint equations.
5. Compute derivatives of J_{P_t} and g_V w.r.t. $\mathbf{x_s}^n$, based on Eqs. (6.17) and (6.21).
6. Use Eq. (6.18) to compute derivatives w.r.t. α^m.
7. Employ GCMMA to update α^m.
8. Go to step 2 unless convergence criteria are met.

6.3.6 Parallelization of the Cut–Cell TopO Algorithm

The computation of the boundary indicator variable takes place on the master processor and can be parallelized due to the absence of any race conditions; having a α^m field, sequential

computations lead to the computation of the nodal values of ϕ^m, using Eqs. (6.11) and (6.12). First, this requires finding and storing the neighboring nodes within the specified r–sphere utilizing the Octree data structure. Even though this process occurs only once, it is associated with a notable computational effort that scales w.r.t. the mesh size and d^r [144, 271]. Hence, OpenMP directives are introduced to parallelize these tasks. Similarly, parallelism is introduced for the computation of SDs (Eq. (6.18)) using OpenMP directives, as it obtains contributions from each node inside its r–sphere, Eq. (6.20). Furthermore, the cell–cutting procedure (Section 6.3.2) occurs multiple times during the CCTopO and benefits from additional parallelization using OpenMP directives. Overall, the achieved gain scales linearly w.r.t. the number of threads for the parallelized tasks. For example, on a background mesh consisting of approximately $2.2M$ nodes, the task of finding the neighboring nodes in a sphere of $d^r = 5$ would take about 15 minutes, however, when this task is parallelized and executed on a hardware with 12 available threads, it requires approximately 75 seconds, which translates to $12\times$ reduced time (Section 6.4.5).

6.4 Topology Optimization using the Cut–Cell Method

In the following Section, the CCTopO algorithm is used on benchmark TopO cases and is compared with solutions obtained using the SPTopO approach (Section 6.2). The comparisons both illustrate the capabilities of the CC–based method and highlight its differences from SP-TopO approaches due to the solid boundary tracking and imposition of accurate solid boundary conditions in each optimization cycle. For the comparisons to be as clear as possible, the optimized SPTopO solutions, henceforth referred to as re–evaluated Standard Porosity (r–SP) solutions, are first re–evaluated by tracking the solid–fluid interfaces, similarly to the CCTopO approach. The re–evaluation procedure is briefly described in Appendix D. In detail, the optimized porosity field is mapped onto nodal material values and, then, onto boundary indicator variables. Then, the CCs can be created to extract the fluid domain and solve the governing equations using the CC flow solver. The described re–evaluation process is fully automated. Contrarily, using traditional body–fitted meshes would first require the extraction of an explicit representation of the solid wall segments, c.f. [127].

In all cases considered, the optimal connection (in terms of J_{P_t}, Eq. (4.57)) of the fixed inlets and outlets with fluid path(s) is targeted. The volume constraints are imposed to match the ones specified in the literature, $\frac{\Omega_f{}^*}{\Omega_D}$. The V_{tar} values of the volume constraint (Eq. (6.1)) are adjusted to exclude the fluid volume that exists at the I/O fluid channels.

6.4.1 CCTopO – Single Inlet–Single Outlet Case

The first case revisits the case optimized by SPTopO, Section 6.2.1. The area constraint specified is $V_{tar} = 0.155$, which corresponds to a fluid volume in the design space of $\frac{\Omega_f{}^*}{\Omega_D} = 0.25$. The optimization is performed on a background mesh with $16.5K$ nodes at three Reynolds numbers $Re_W = 2,\ 20,\ 200$, computed based on the inlet width W, and a smoothing radius of $d^r = 4$. The optimization wall–clock time amounted to approximately 200 minutes on 24 AMD EPYC 7401 (2.0 Ghz) processors for all three Re_W. This includes all additional CCTopO requirements, such as extraction of the solid walls, CC mesh generation and solution of both the flow and adjoint equations in each optimization cycle.

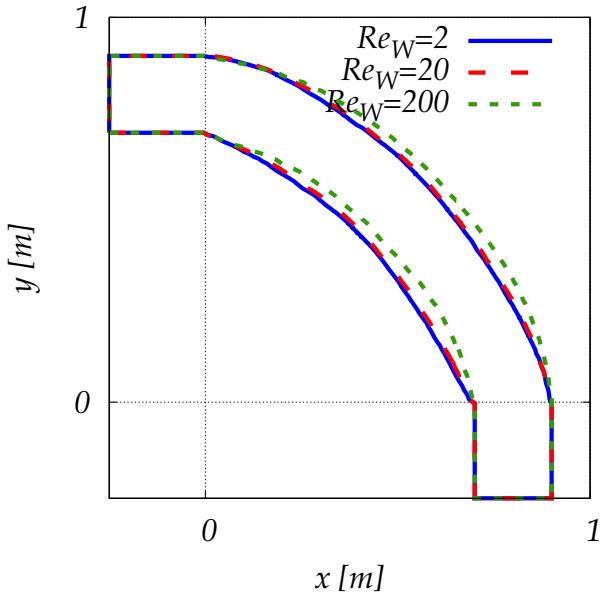


Figure 6.14: CCTopO – Single Inlet–Single Outlet case - Solid boundaries of the optimized fluid paths for $Re_W = 2,\ 20,\ 200$.

The optimized designs are shown in Figure 6.14 and agree well with observations made in the literature [82, 132, 200]. By increasing the Re_W, minor losses become more pronounced favoring a longer, more round duct. In contrast, in lower Re_W a shorter, wider duct is preferred due to smaller shear stresses that reduce the dominating major losses. In Figure 6.15, the evolution of the objective function value and area inequality constraint is presented for $Re_W = 2$. The reduced initial objective value corresponds to an infeasible design since the constraint is not satisfied. The fluid region is constricted by attempting to satisfy the constraint, leading to increased pipe losses and, thus, objective function values. However, once a feasible solution

is obtained, Figure 6.16b, the GCMMA algorithm leads the optimization to better qualified feasible designs. The optimized designs in Figure 6.14 correspond to $J_{P_t} = 8.86 \times 10^{-3} \frac{W}{m}$ for $Re_W = 2$, $J_{P_t} = 9.39 \times 10^{-4} \frac{W}{m}$ for $Re_W = 20$, and $J_{P_t} = 1.11 \times 10^{-4} \frac{W}{m}$ for $Re_W = 200$, while g_V is satisfied.

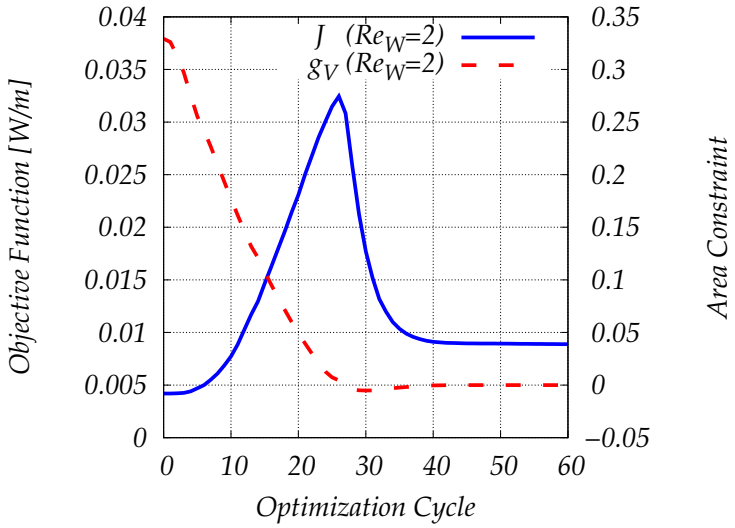
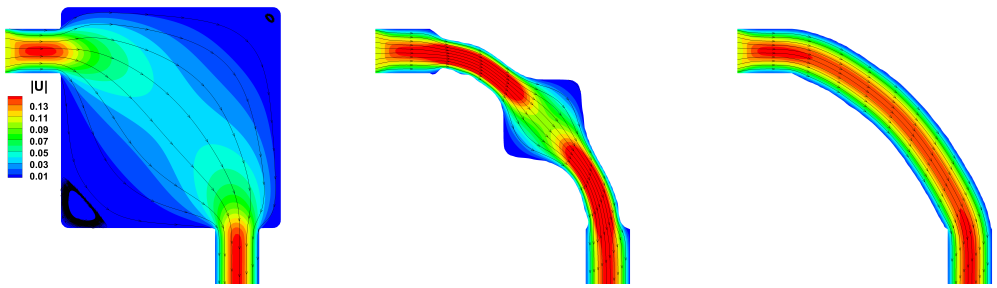


Figure 6.15: CCTopO – Single Inlet–Single Outlet case, $Re_W = 2$, $d^r = 4$ - Evolution of objective and constraint functions converging to $J_{P_t} = 8.86 \times 10^{-3} \frac{W}{m}$ and $g_V \approx O(10^{-5})$.



(a) **(b)** **(c)**

Figure 6.16: CCTopO – Single Inlet–Single Outlet case, $Re_W = 2$ - Velocity magnitude iso–areas of different optimization cycles. Plots correspond to (a) initial, (b) cycle 30, and (c) final cycle.

Following, the optimized design of CCTopO for $Re_W = 200$ is compared with the outcome of SPTopO. The optimized SPTopO solution exhibited a binary porosity distribution, and is shown in Figure 6.17a along with the generated solid walls after the re–evaluation process. The generated CC mesh in Ω_f had practically the same area as the SPTopO optimized solution $\left(\Delta\Omega_f \leq 10^{-3}{\Omega_f}^{SPTopO}\right)$. The computed J_{P_t} values in both cases read $J_{P_t}^{SPTopO} = 1.6 \times 10^{-4} \frac{W}{m}$

and $J_{P_t}^{r-SP} = 1.2 \times 10^{-4} \frac{W}{m}$. Their difference illustrates the importance of imposing accurate boundary conditions along the solid walls, since objective function values computed by SPTopO may have differences from that computed by generating a body–fitted mesh and re–evaluating the obtained design on it.

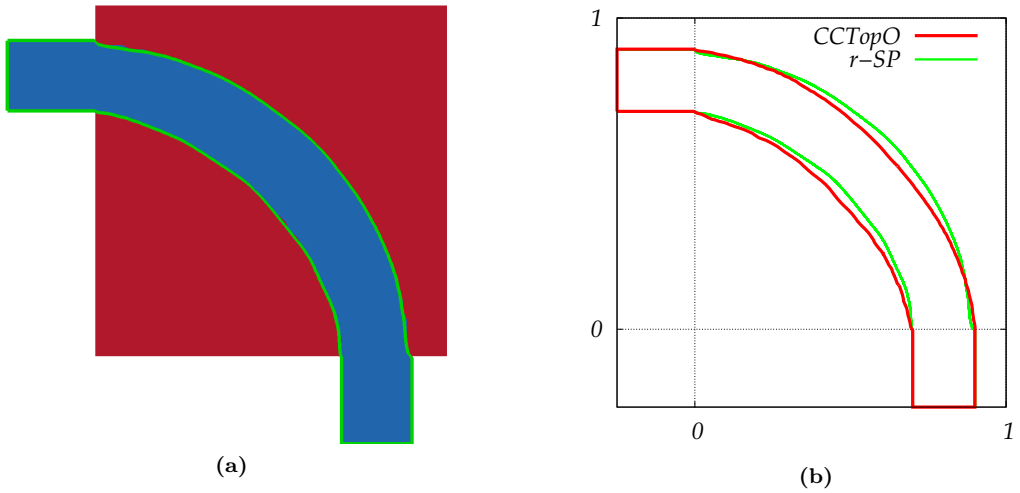


Figure 6.17: Comparison of SPTopO and CCTopO in the Single Inlet–Single Outlet case at $Re_W = 200$ - (a) r–SP solid boundaries (green line) with the area with $a_b = 1$ in red and that with $a_b = 0$ in blue. (b) Comparison of the CCTopO and r–SP solid boundaries.

Comparison of the re–evaluated solution $J_{P_t}^{r-SP}$ and the outcome of CCTopO, $J_{P_t}^{CCTopO} = 1.11 \times 10^{-4} \frac{W}{m}$, shows that the design optimized by CCTopO exhibits 8% extra reduction in J_{P_t}. The solid boundaries of the CCTopO design, Figure 6.17b, corresponds to a slightly shorter and wider duct that could explain the difference in the J_{P_t} value.

Figures 6.18a–6.18d show the computed SDs, w.r.t. the porosity field α_b^m (Figures 6.18a–6.18c) and w.r.t. the material field α^m (Figures 6.18b–6.18d), at the first and last optimization cycle for SPTopO and CCTopO, respectively. The main difference between them becomes clear. In SPTopO, the SDs are computed throughout Ω_D, whereas in CCTopO, a sensitivity map is obtained along the solid walls which is then, transformed, to nodal sensitivities of the material field. Thus, the sensitivity derivatives exist only in the vicinity of the current fluid–solid interface and depend on the smoothing radius d^r. Furthermore, the design variables in the two methods influence the objective function differently and, thus, an immediate comparison of scale should not be performed.

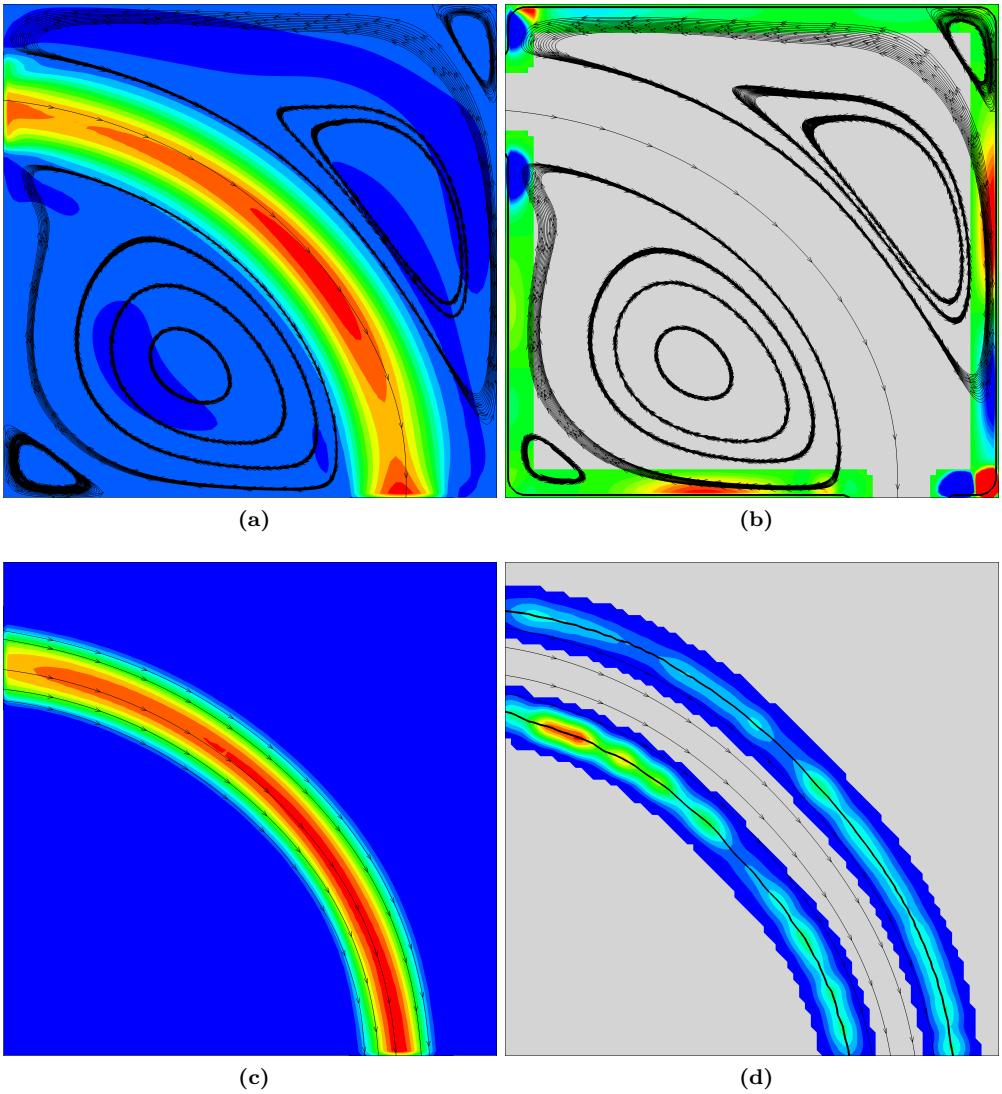


Figure 6.18: Comparison of SPTopO and CCTopO in the Single Inlet–Single Outlet case at $Re_W = 200$ - Initial and final sensitivity field, w.r.t. the porosity field α_b^m in (a),(c) and w.r.t. the material field α^m in (b), (d), computed using SPTopO and CCTopO. Derivatives at the gray areas of (b) and (d) are not computed.

6.4.2 Single Inlet–Single Outlet Case with Obstacle

In TopO applications, several manufacturing constraints can arise. The most common one is limiting the total volume of the fluid domain Eq. (6.1), since, in this way, optimized solutions become area efficient, i.e. only the most beneficial fluid regions remain. However, cases in which regions cannot become fluid can also arise. Several such cases could appear in cross–

flow heat exchanges. To incorporate simplified such applications in the CCTopO algorithm, an impassable region (orange square in Figure 6.19) is introduced imitating the cross–flow duct. TopO is performed for two different obstacle positions for, $Re_W = 2$ and the same $V_{tar} = 0.155$. In the two cases, the obstacle is positioned at $(\delta_x, \delta_y) = (0.25, 0.25)$ and $(\delta_x, \delta_y) = (0.39, 0.29)$ and leads to different optimized solutions.

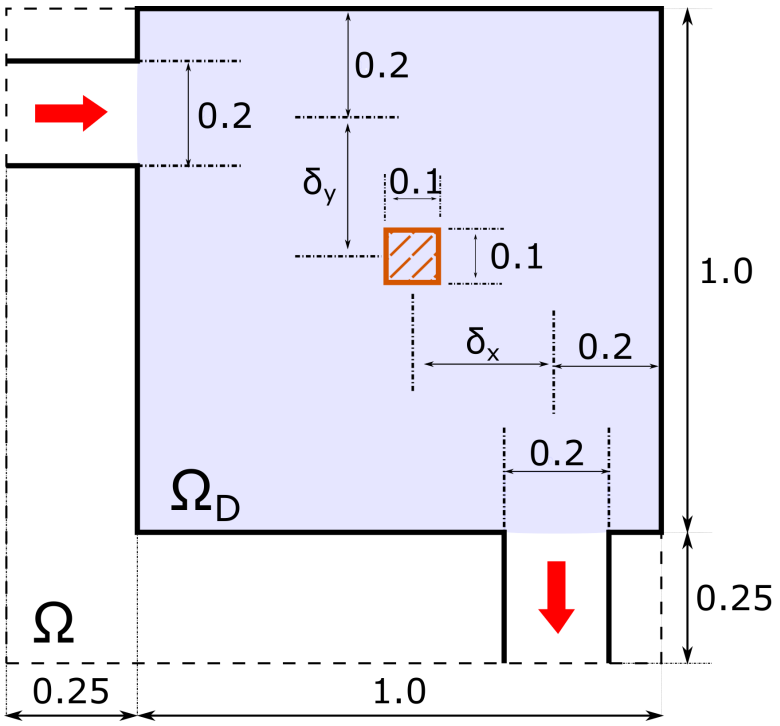


Figure 6.19: CCTopO – Single Inlet–Single Outlet case with additional obstacle - The orange square imitates an impassable region that remains solidified. Thus, the arising fluid path needs to circumvent the obstacle. The highlighted area specifies part of the domain in which the design variables are allowed to change during the optimization and defines the design domain Ω_D. Dimensions are in m.

6.4.2.1 Single Inlet–Single Outlet Case with Obstacle $(\delta_x, \delta_y) = (0.25, 0.25)$

In the first case, the obstacle is positioned at equal distances from the domain boundaries. The evolution of J_{P_t} and g_V shows similarities with the case where no obstacle is included, Figure 6.20. Starting from an infeasible design with a low J_{P_t} value, the optimization rapidly decreased the constraint function value, leading to an increase in J_{P_t}. After a feasible design is found, the J_{P_t} is seen to decrease up to the value of $J_{P_t} = 8.97 \times 10^{-3} \frac{W}{m}$.

In Figure 6.21 four different snapshots of the TopO are shown. In the initial solution,

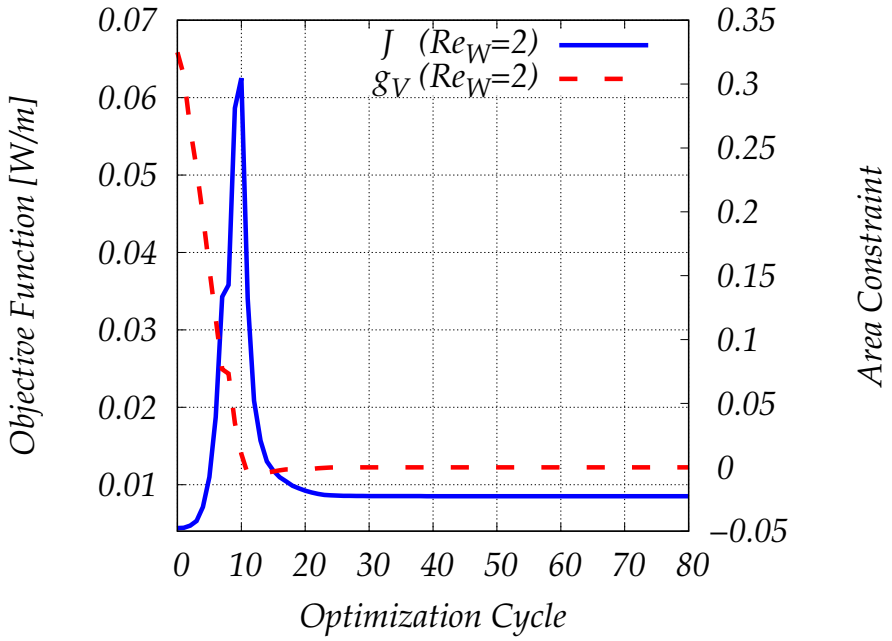


Figure 6.20: CCTopO – Single Inlet–Single Outlet case with additional obstacle, $(\delta_x, \delta_y) = (0.25, 0.25)$ - Evolution of the objective and constraint functions converging to $J = 8.97 \times 10^{-3} \frac{W}{m}$ and $g_V \approx O(10^{-5})$ for the $Re_W = 2$ and $d^r = 4$.

Figure 6.21a, the presence of the obstacle forces the fluid paths to split. In Figure 6.21b, even though a feasible solution is obtained (see Figure 6.20), the solid boundaries are rather crude, and the duct consists of two steep bends that cause a relatively high J_{P_t} value. After 11 more optimization cycles, Figure 6.21c, the duct geometry observed starts to straighten and form, but some roughness still exists at the solid boundaries. Then, the optimization algorithm performs some fine–tuning of the geometry boundaries up until the termination criterion is met, leading to the smoother, optimized duct shown in Figure 6.21d. Comparison of the computed J_{P_t} with the case without the obstacle reveals an increase of approximately 1%.

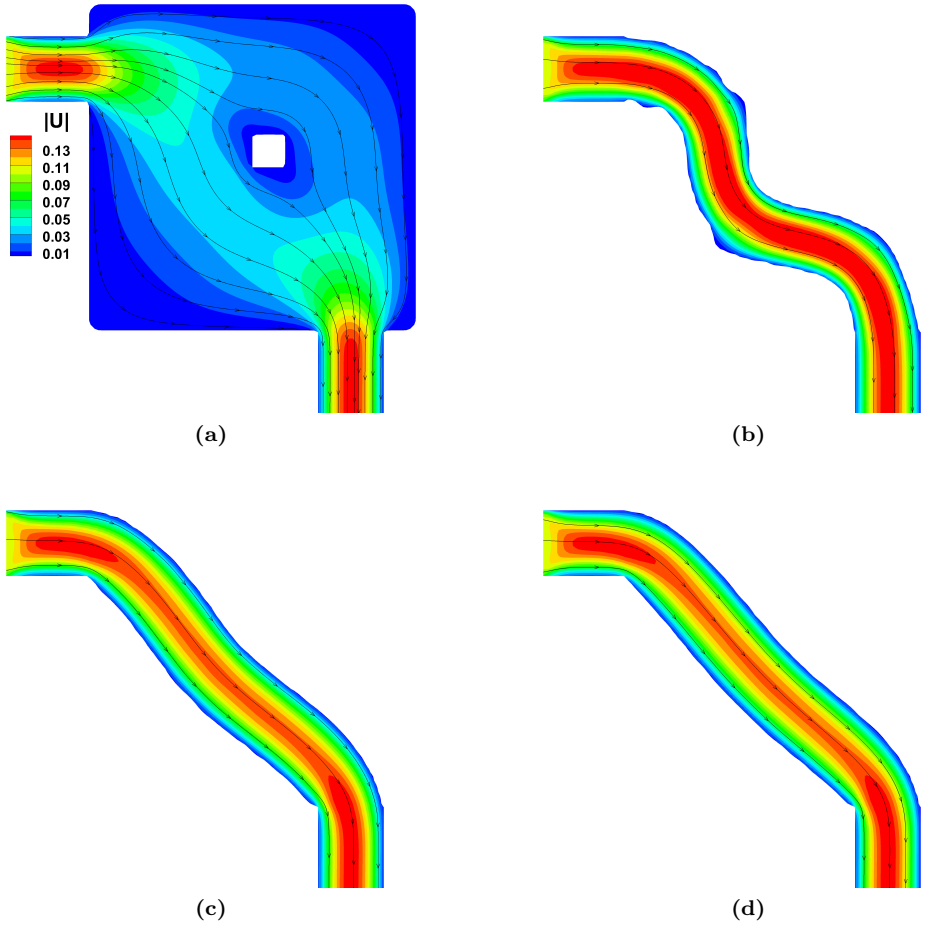


Figure 6.21: CCTopO – Single Inlet–Single Outlet case with additional obstacle, $(\delta_x, \delta_y) = (0.25, 0.25)$ - Velocity magnitude iso–areas of different optimization cycles. Plots correspond to (a) initial, (b) cycle 15, (c) cycle 26, and (d) final cycle.

6.4.2.2 Single Inlet–Single Outlet with Obstacle $(\delta_x, \delta_y) = (0.39, 0.29)$

In the second case, the obstacle is placed off–centered to try and obtain a fluid path that passes over the obstacle. In Figure 6.22 the evolution of the objective and constraint functions is presented for the performed TopO. Again, the same trend as in the previous cases is observed. After the TopO is terminated, the objective function value is $J_{P_t} = 1.0105 \times 10^{-2} \frac{W}{m}$.

In Figure 6.22, several snapshots of the TopO are shown. Starting from the same fully fluid design domain, Figure 6.23a, the presence of the obstacle again forces the flow to split into two different paths, Figure 6.23b. However, the majority of the fluid seems to go through the 'outer' fluid path resulting in the gradual closing/solidification of the 'inner' path due to the

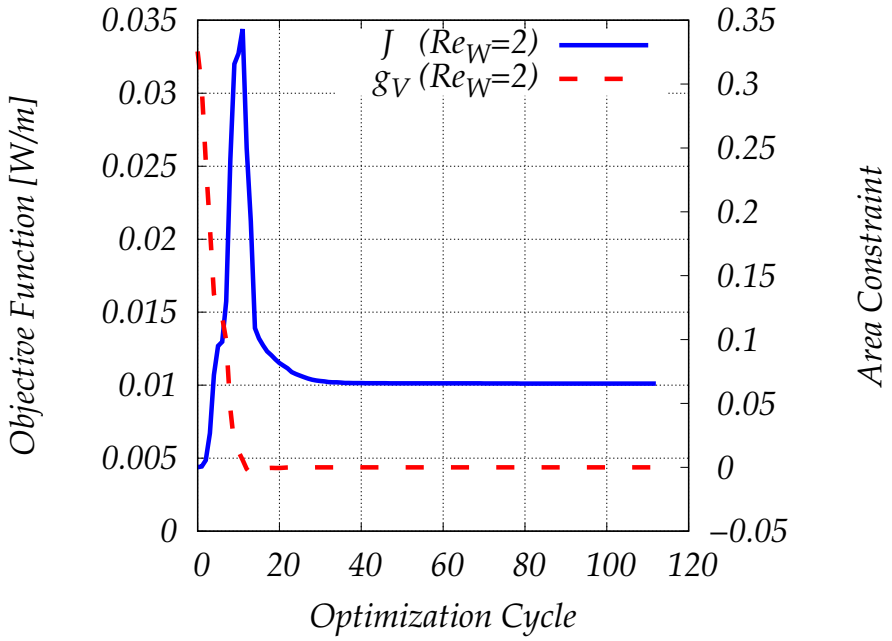
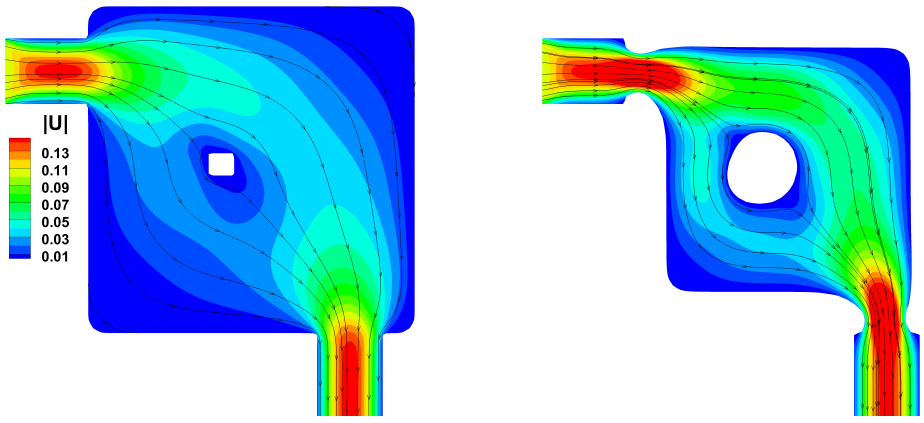
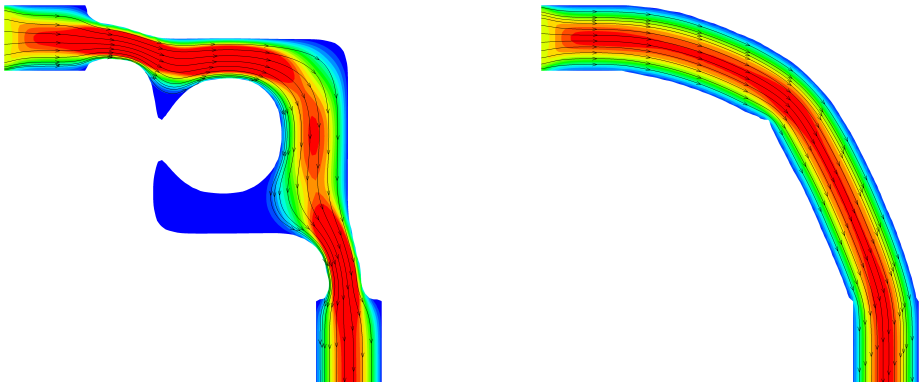


Figure 6.22: CCTopO – Single Inlet–Single Outlet case with additional obstacle, $(\delta_x, \delta_y) = (0.39, 0.29)$ - Evolution of the objective and constraint functions converging to $J_{P_t} = 1.0105 \times 10^{-2} \frac{W}{m}$ and $g_V \approx O(10^{-7})$ for the $Re_W = 2$ and $d^r = 4$.

area constraint, Figure 6.22c. Finally, in Figure 6.22d a smooth, elongated duct is obtained that passes over the obstacle and, compared to the previous case, has a 12% higher objective function value.



(a) **(b)**



(c) (d)

Figure 6.22: CCTopO – Single Inlet–Single Outlet case with additional obstacle, $(\delta_x, \delta_y) = (0.39, 0.29)$ - Velocity magnitude iso–areas of different optimization cycles. Plots correspond to (a) initial, (b) cycle 4, (c) cycle 9, and (d) final cycle.

6.4.3 Single Inlet–Two Outlet Case

The next TopO case consists of obtaining the fluid paths connecting a single inlet with two outlets, Figure 6.23. The CCTopO is performed on a background mesh of $66K$ nodes at $Re_W = 200$ and $V_{tar} = 0.15$, Koch *et al.* [127], in which a porosity–based method is used for the same TopO problem. The area constraint requires at most 30% of the design domain to be fluid, i.e. $\frac{\Omega_f{}^*}{\Omega_D} = 0.3$. The so–optimized design is performed on 24 AMD EPYC 7401 (2.0 Ghz) processors for 35 hours and results in a $J_{P_t}^{CCTopO} = 0.109\frac{W}{m}$ after 100 cycles.

In Figure 6.24, the evolution of the objective and area inequality constraint function of CCTopO is presented. In the first cycles, designs fails to meet the area constraint. Thus, the fluid domain is restricted, leading to an increase in the J_{P_t} values. The reduction in J_{P_t} starts once a feasible design is found. At the 40^{th} cycle, the optimized design shape was mostly settled, with slight changes occurring at the area where the duct splits. Due to the higher sensitivities in that area, changes in the design variables alter the splitting angle affecting the obtained J_{P_t} values. The optimal porosity distribution, obtained by SPTopO, is used to trace the solid walls to re–evaluate this solution using the CC method. The r–SP fluid area is again essentially the same as that of SPTopO, i.e. $\left(\Delta\Omega_f \approx 10^{-3}{\Omega_f}^{SPTopO}\right)$.

Comparison of the computed velocity fields for both the SPTopO and r–SP, Figures 6.26a and 6.26b, leads to a few interesting observations. First, a separation zone manifests in both solutions near the right outlet due to the steepness of the connecting duct, Figure 6.26c. Fur-

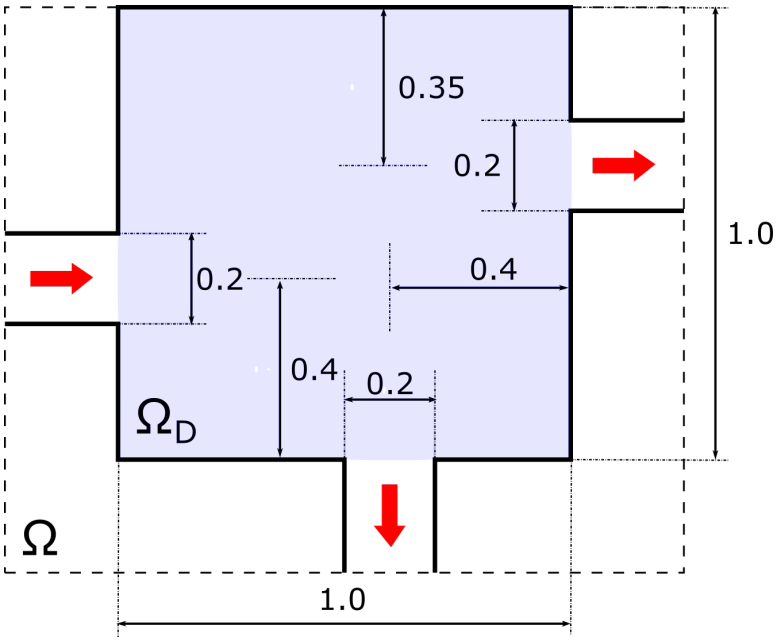


Figure 6.23: Single Inlet–Two Outlet case TopO problem description - A fluid path is sought connecting a single inlet with two outlets. The inlet is positioned at the western side of the computational domain Ω, while the outlets are non–equidistantly positioned at its southern and eastern side. The highlighted area specifies the domain in which the nodal material field α^m is allowed to change during the optimization and defines the design domain Ω_D. Dimensions are in m.

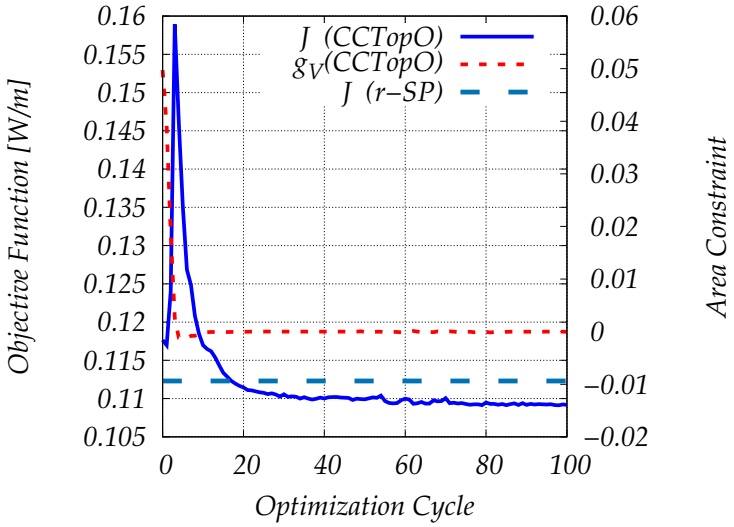
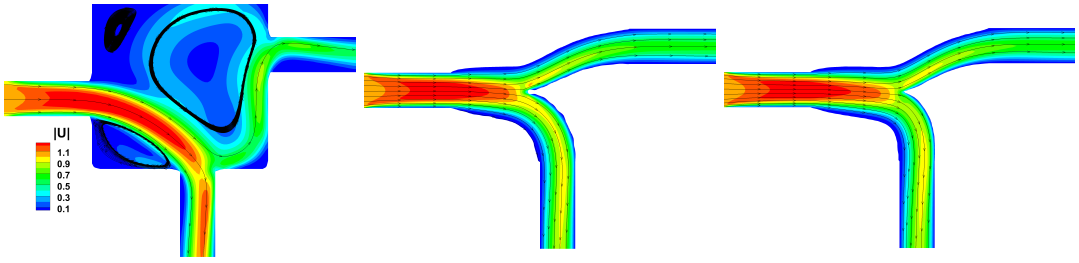


Figure 6.24: CCTopO – Single Inlet–Two Outlet case, $Re_W = 200$, $d^r = 5$ - Evolution of the objective and constraint functions converging to $J^{CCTopO} = 0.109\frac{W}{m}$ and $g_V^{CCTopO} \approx O(10^{-6})$.



(a) (b) (c)

Figure 6.25: CCTopO – Single Inlet–Two Outlet case, $Re_W = 200$, $d^r = 5$ - Velocity magnitude iso–areas of different optimization cycles. Plots correspond to (a) initial, (b) cycle 70, and (c) final cycle.

thermore, in the r–SP solution, a second separation zone, non–existent in the SPTopO solution, can be seen in Figure 6.26d due to the accurate boundary conditions imposed along the solid walls. This variation in the obtained flow patterns reflects upon the difference (by approximately 3%) in J_{P_t}, since $J_{P_t}^{SPTopO} = 0.115\frac{W}{m}$ and $J_{P_t}^{r-SP} = 0.112\frac{W}{m}$. Comparing the J_{P_t} values of r–SP and CCTopO, Figure 6.24, a small improvement of 2% in J_{P_t} is observed. However, the attained designs, shown in Figures 6.25c and 6.26b, are quite different. The shape obtained by r–SP, the first part of the duct is inclined towards the bottom outlet, giving rise to the steepened duct that connects to the right outlet and causes flow separation. In contrast, the CCTopO retains an almost horizontal first part that splits into a wider lower and a narrower upper duct. The part connecting to the right outlet changes gradually, avoiding thus flow separation. Furthermore, the first part splits into two parts much earlier and is wider near the inlet, decelerating the bulk flow. Overall, the optimal design of the CCTopO is more inline with the optimized design presented in [127], using ShpO, on a much denser body–fitted mesh, after transitioning from a TopO to ShpO. This highlights the ability of CCTopO to solve TopO problems while retaining the accuracy of CFD simulations that impose accurate boundary conditions, as is usually the case in ShpO.

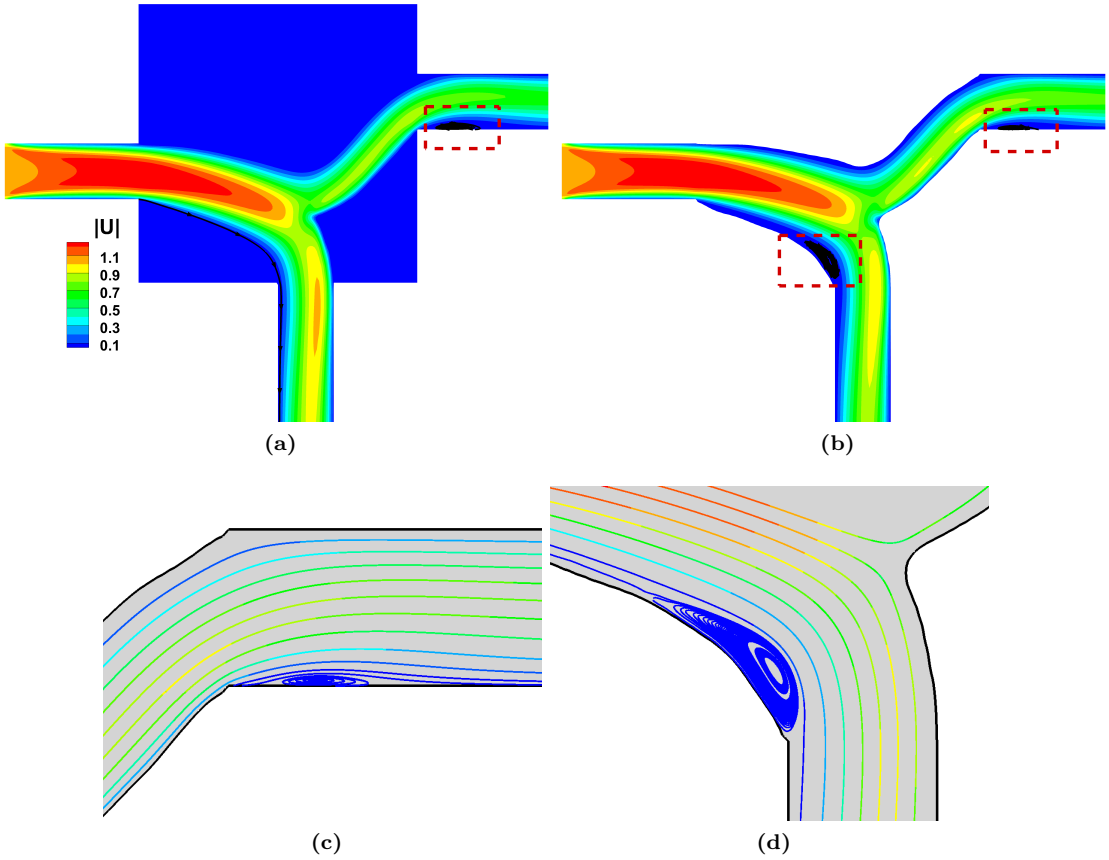


Figure 6.26: Comparison of the SPTopO and the r–SP flow solutions - Velocity magnitude iso–areas computed by the (a) SPTopO and (b) r–SP. (c) Flow separation close to the right outlet. (d) Second flow separation zone appearing only in r–SP solution.

6.4.4 Two Inlet–Two Outlet Case

The two inlet–two outlet TopO case is optimized for two different δ values, Figure 6.27, both at $Re_W = 17$ and a $V_{tar} = 0.33$. This problem has been also studied in [37, 148, 200] and, depending on δ, different fluid path layouts emerge, demonstrating TopO's flexibility in finding topologically different designs. Since δ describes the width–to–height ratio, there exists a sufficiently small δ value for which the two fluid paths favor a short and wide duct and remain separate. For larger δ values, it is favorable to merge the two fluid paths to create a joined duct. CCTopO is performed for $\delta = 1$ in which the two flow paths remain separate, and for $\delta = 1.5$ in which they merge into a joined duct.

In Figure 6.28, the evolution of the objective and constraint functions are shown for both cases. The stopping criterion is reached after 120 ($\delta = 1$) and 47 ($\delta = 1.5$) cycles, while in

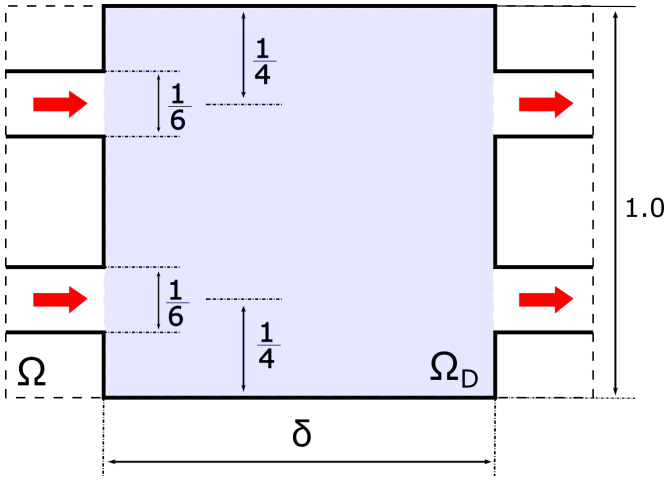


Figure 6.27: Two Inlet–Two Outlet case problem description - A fluid path is sought connecting two inlets positioned at the west side with two outlets positioned at the west side. Based on the distance between the inlets and outlets δ, topologically different optimized solutions arise. The highlighted area specifies the domain in which the nodal material field α^m is allowed to change during the optimization and defines the design domain Ω_D. Dimensions are in m.

both cases the area constraint is satisfied $g_V \leq 10^{-5}$. In the $\delta = 1$ case a larger number of optimization cycles is required, since the solid boundaries between the inlets need to merge with those between the outlets to form two separate fluid paths [148], Figure 6.29b. This is also reflected upon the total optimization time which amounts to approximately 30/16 hours on 24 AMD EPYC 7401 (2.0 Ghz) processors.

In Figure 6.29, different snapshots of the velocity magnitude iso–areas for the $\delta = 1$ TopO are shown, with its initial infeasible solution in Figure 6.29a. Figure 6.29b presents an intermediate solution, in which the solid boundaries between inlets and outlets constrict toward the center to merge and, then, separate into two distinct fluid paths leading to the optimized solution of Figure 6.29c.

Figure 6.30 shows the corresponding snapshots of the velocity magnitude iso–areas for the merging TopO case ($\delta = 1.5$). The initially fully fluid design domain is shown in Figure 6.30a. The fluid domain is constricted due to the area constraint, leading to the merged, intermediate flow paths shown in Figure 6.30b. Even though they correspond to an infeasible solution, the difference with the separating case is obvious. Finally, a feasible, symmetric design is obtained after 47 optimization cycles, Figure 6.30c. Both optimized designs presented, match those of [37, 148, 200]. The computed objective function values were $J^{\delta=1} = 1.82\frac{W}{m}$ and $J^{\delta=1.5} = 1.84\frac{W}{m}$.

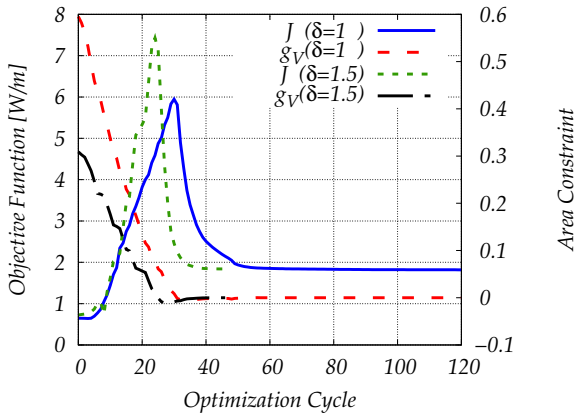
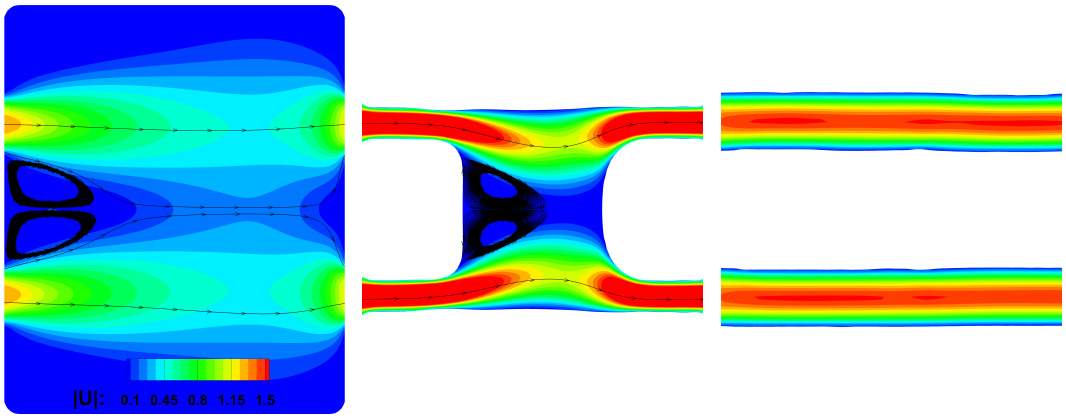
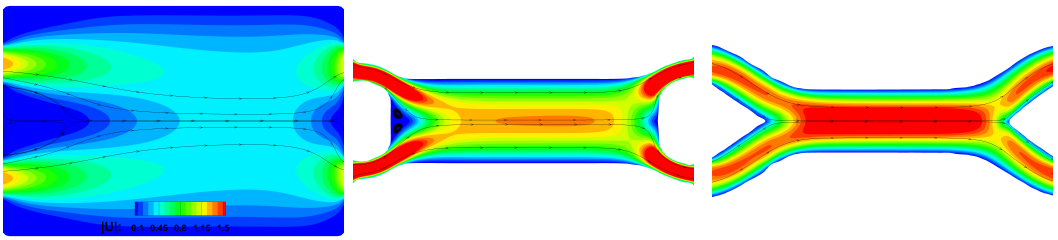


Figure 6.28: CCTopO – Two Inlet–Two Outlet case, $Re_W = 17$, $d^r = 4$ - Evolution of objective and constraint functions for two different δ values.



(a) **(b)** **(c)**

Figure 6.29: CCTopO – Two Inlet–Two Outlet case, $Re_W = 17$, $\delta = 1$ - Velocity magnitude iso–areas of different optimization cycles. Plots correspond to (a) initial, (b) cycle 30, and (c) final cycle.



(a) **(b)** **(c)**

Figure 6.30: CCTopO – Two Inlet–Two Outlet case, $Re_W = 17$, $\delta = 1.5$ - Velocity magnitude iso–areas of different optimization cycles. Plots correspond to (a) initial, (b) cycle 25, and (c) final cycle.

6.4.5 3D Manifold TopO

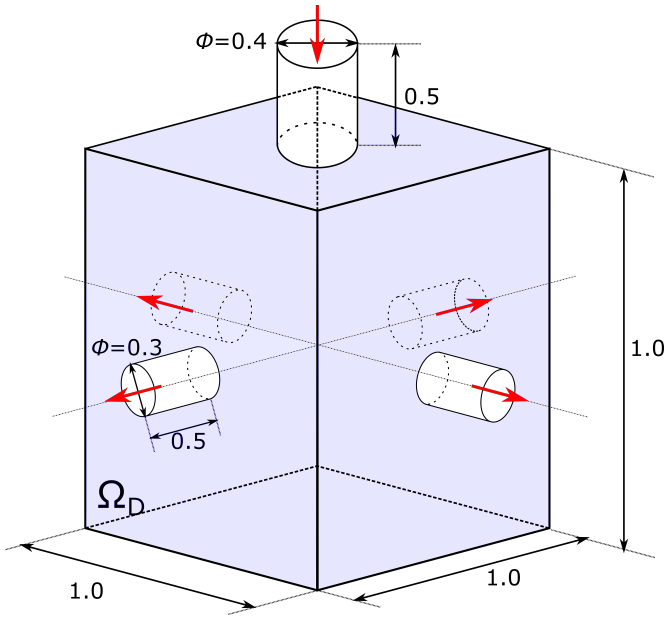


Figure 6.31: $3D$ manifold case problem description - A fluid path is sought connecting a single inlet positioned at the top with four symmetrically place outlets along the z–plane. The inlet and outlet channels are excluded from the design space, thus the α^m field is allowed to change only in the highlighted cubic domain, Ω_D. Dimensions are in m.

In this section, the CCTopO method is used for the design of a $3D$ manifold. The case consists of a single inlet at the top and four symmetrically placed outlets on the $z = 0$ plane, Figure 6.31. The Reynolds number is $Re_D = 4$, based on the inlet diameter D. The problem is setup with $V_{tar} = 0.05$ target, corresponding to $\frac{\Omega_f{}^*}{\Omega_D} = 0.2$. The initial design domain Ω_D is defined as fluid. The background Ω mesh consists of approximately $2.2M$ nodes, resulting in an initial Ω_f mesh with approximately $250K$ cells.

In Figure 6.32, the evolution of the objective and constraint functions are shown for the $3D$ manifold case. Due to the increased size of the TopO problem, more than two million design variables and relatively costly flow solutions, a more aggressive GCMMA algorithm is set up so that a reduced number of optimization cycles is obtained. This translated to an optimization time of about 5 days on 48 AMD EPYC 7401 (2.0 Ghz) processors . The optimization termination criteria was met after 29 cycles with $J_{P_t} = 3.49 \times 10^{-3} W$ and a $g_V \approx O\left(10^{-6}\right)$, obtaining the design shown in Figures 6.33 and 6.34 that show different views on the obtained flow solution.

In Figure 6.33, the isometric view is presented, showing the velocity field inside the optimized

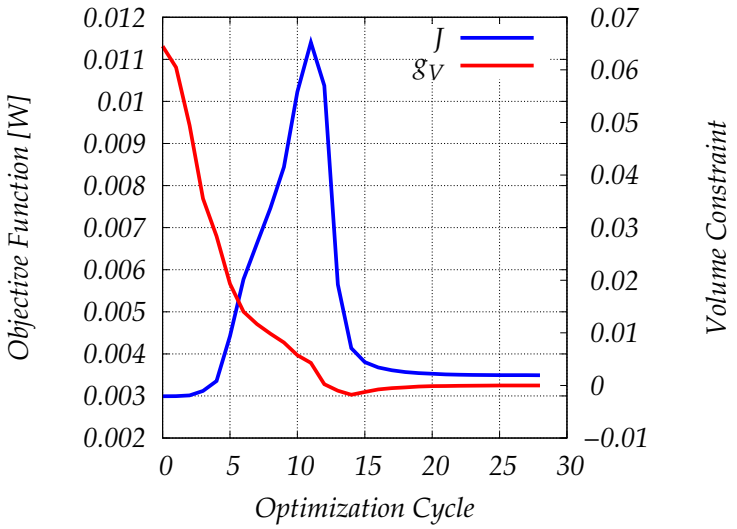


Figure 6.32: CCTopO – $3D$ manifold case, $Re_D = 4$ - Evolution of objective and constraint functions for min J_{P_t}.

shape, while in Figure 6.34 several pressure iso–surfaces are shown, colored by the velocity field revealing the symmetry of the optimized design. A closer look at the evolution of J_{P_t} and g_V, Figure 6.32, shows a similar evolution with the $2D$ cases. The initial solution corresponds to an infeasible design with quite a low J_{P_t} value, as Ω_D (Figure 6.31) was initialized as fluid. Similarly to the previous cases, the violation of the volume constraints leads to increased J_{P_t} values by restricting the fluid domain. Once it is no more violated, GCMMA focuses on minimizing J_{P_t} by obtaining continuously better–suited feasible designs. This was achieved by eliminating stagnated flow areas by allocating their 'effective' volume near the inlet and outlet positions, resulting in the obtained tuboid design. In the optimized design, the flow is attached and equally distributed to the four outlets.

Figure 6.33 shows the velocity magnitude iso–areas of the optimized design along the $x = 0.5$ and $y = 0.5$ planes, indicating the absence of flow recirculation. Figures 6.35a and 6.35b present xy and yz planar views of the optimized design showing the resulting fluid path connections. In Figure 6.36 the solid boundary of the optimized design at the bottom side is shown, with a close–up view of some solid faces created during the CC mesh generation. Each triangle corresponds to a solid face on which no–slip boundary conditions are imposed.

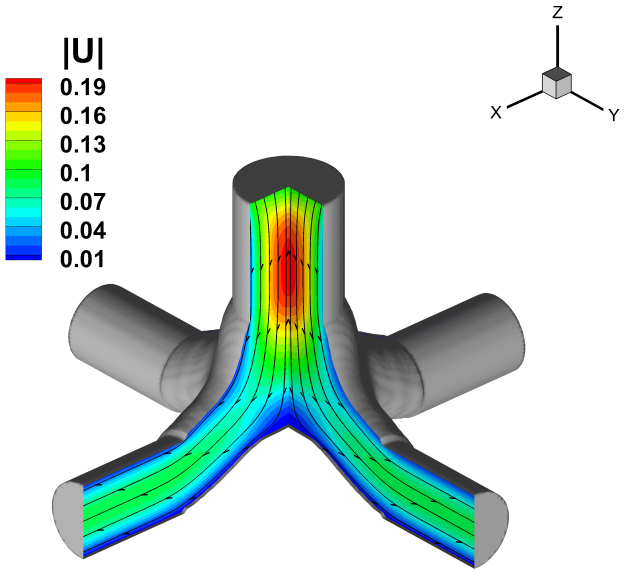


Figure 6.33: CCTopO – $3D$ manifold case, $Re_D = 4$ - Velocity magnitude iso–areas of the optimized manifold along the $x = 0.5,\ y = 0.5$ planes.

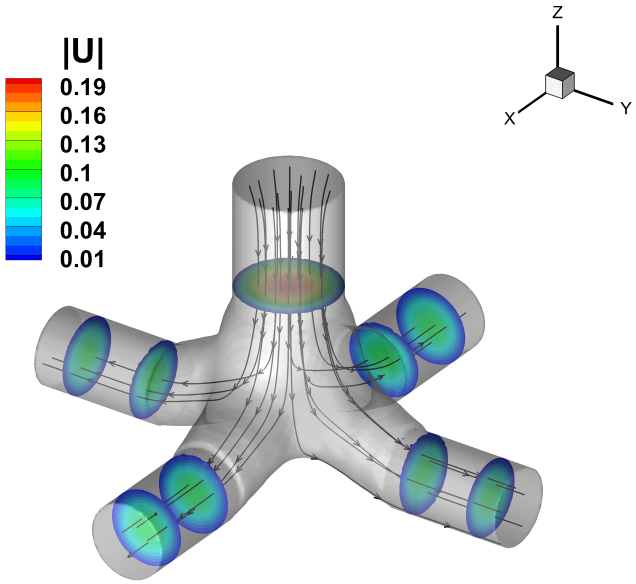
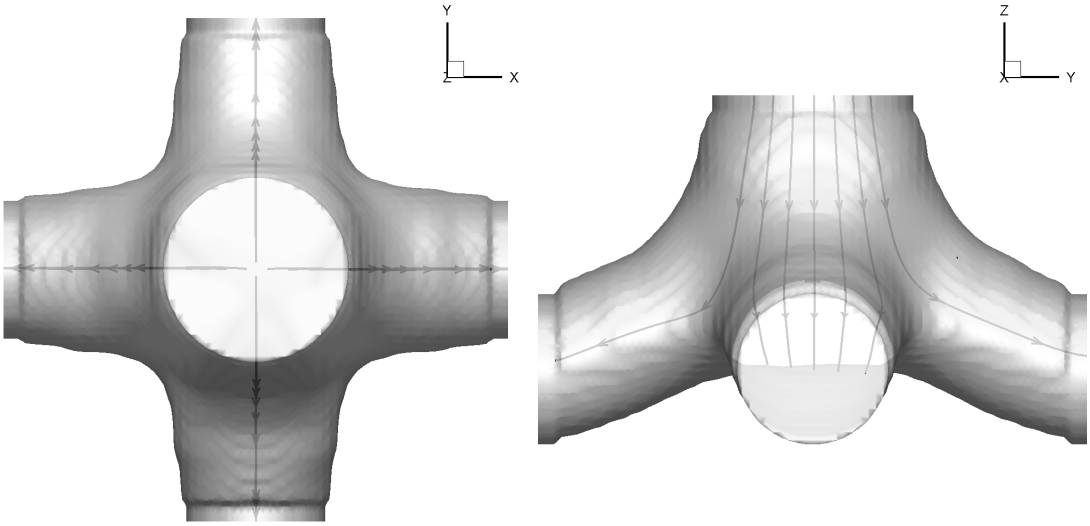


Figure 6.34: CCTopO – $3D$ manifold case, $Re_D = 4$ - Pressure iso–surfaces $[0.02,\ 0.07,\ 0.16]$ of the optimized manifold colored by the velocity magnitude, showing a symmetric flow solution.



(a) **(b)**

Figure 6.35: CCTopO – $3D$ manifold case, $Re_D = 4$ - Different views of the $3D$ manifold optimized design. View normal to the (a) $xy-$plane and (b) $yz-$plane for the optimized design with selected velocity streamlines, showing the symmetrical solution.

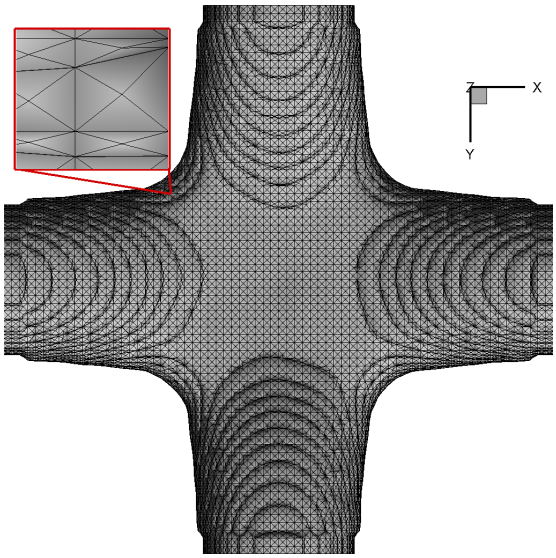


Figure 6.36: CCTopO – $3D$ manifold case, $Re_D = 4$ - Bottom view of the $xy-$plane with a close–up on some solid faces of the generated CC mesh during the last optimization cycle.

6.5 Concluding Remarks

In this chapter, a new CC–based TopO algorithm that reconstructs the sought solid boundaries and enforces accurate boundary conditions during the optimization, was presented. First, a comparative study of the accuracy of flow solutions obtained using the Brinkman and the CC method revealed that the Brinkman method's ability to approximate solid walls effects is significantly influenced by both the penalization parameter and the existence of intermediate porosity values.

The new TopO algorithm relies on an auxiliary background field that defines the design variables of the optimization. The boundary indicator field, which implicitly describes the fluid, solid regions and their interface, can be computed using the said field in each optimization cycle. This enables topological changes to occur and circumvents the need of numerically solving a HJE. To create the CC mesh based on the boundary indicator field, Cartesian cells that have part of the interface immersed within them are cut, have their fluid part isolated and in turn, create CCs along the solid walls. However, since no geometry is explicitly defined, the CC mesh generation method presented in Chapter 2 needed to be modified accordingly.

In the context of gradient–based TopO, the algorithm is supported by the flow solver described in Section 3.1 and the continuous adjoint method presented in Chapter 4. TopO dictates the use of constraints; this was realized by the GCMMA algorithm that proved to be very efficient despite the large number of design variables. CCTopO was compared with SPTopO, a method that avoids the reconstruction of solid walls, to assess its performance. To make fair comparisons, a porosity distribution post–processing tool (Appendix D) that extract the solid walls and re–evaluate the optimized solutions on a body–conforming mesh was developed. In the studied examples, the higher flow solution accuracy of CCTopO, compared to SPTopO, translated into finding better performing designs, at the expense of an increased computational cost. For example, in the Single Inlet–Single Outlet Case SPTopO and CCTopO require 130 and 200 minutes, respectively.

Overall, CCTopO provides additional flexibility in terms of considered design criteria and extensions. The solid boundary reconstruction using the CC method can enable design criteria, defined specifically on solid walls, or the inclusion of turbulence, where near–wall effects are of the essence, to be better considered. The crisp fluid–solid interface can also aid to further extend TopO in multidisciplinary applications such as aero–structural, conjugate heat transfer problems or any other problem with two disciplines strongly interacting along their interface.

Chapter 7

Closure

This PhD thesis aimed at the development of a highly automated numerical framework for the analysis, and shape and topology optimization of aero/hydrodynamic bodies utilizing Cartesian meshes, and more precisely, the Cut-Cell (CC) method. Research focused on extending the scope of a baseline in–house CC software to areas in which the application of the CC method is challenging but also rewarding. To this end, the development of numerical methods that allow for the simulation of two–phase and turbulent flows exhibiting cavitation and numerical optimization tools based on the continuous adjoint method has been presented.

Extension to the Reynolds–Averaged Navier–Stokes (RANS) equations was realized using the standard $k-\varepsilon$ model, solved coupled with the mean flow equations, opting for better stability and convergence properties. Wall closure was achieved via a modified wall function technique, defined on linelets protruding towards the interior computational domain due to the irregular wall–normal distances of the first cells near the solid surface. Method's accuracy was assessed on $2D$ single–phase, turbulent flows over a flat plate, wherein comparisons with experimental and numerical data showed excellent agreement and a curved channel, showing its ability to compute a smooth skin friction coefficient along the curved surfaces.

The CC method was also developed to enable the simulation of two–phase flows and flows exhibiting cavitation, using the homogeneous mixture model. This model allows for the simulation of a wide range of two–phase flows with high–density ratios at a relatively low–cost, absence of interface momentum exchange terms that are difficult to quantify numerically. Cavitating phenomena were introduced using a cavitation model that describes the evaporation and condensation processes. The flow model was initially validated for inviscid and laminar flows over isolated hydrofoils exhibiting sheet cavitation. The CC solver was then extended to

turbulent flows, using the implemented linelets technique, and compared with data available in the literature on $2D$ and $3D$ benchmark cases for several cavitation numbers. Overall, a very good agreement was observed regarding the developed cavity lengths and computed surface pressure distributions. In addition, the $3D$ cavitator case showcased the ability of the linelets technique to accurately model the turbulent boundary layer with a reasonable and affordable cell–count. The use of the CC method for these types of flows permitted the a–priori mesh refinement at areas of interest, increasing flow simulation accuracy and simultaneously avoiding wasting computational resources.

Part of this thesis dealt with the development of numerical optimization tools for the design of aero/hydrodynamic bodies. This was achieved using a gradient–based method, and more specifically, the continuous adjoint method that enables the computation of the sought objective function gradient at an independent, to the number of design variables, cost. Based on relative work in the PCOpt, the variations of eddy viscosity are incorporated by differentiating the implemented turbulence model. n addition, the CC–specific wall closure technique, based on linelets, was differentiated to provide the adjoint counterpart, viz. the adjoint linelets. In contrast to the work presented in Papoutsis-Kiachagias [191], Zymaris *et al.* [283], wherein the adjoint counterpart acts directly upon the boundary finite volumes, the adjoint linelets introduce source terms at the forcing points that lie in the interior of the domain and have zero distance variation. With regards to the development of the two–phase adjoint formulation, the homogeneous mixture assumption results in mixture density and molecular viscosity being expressed w.r.t. the liquid volume fraction, introducing additional terms to the Field Adjoint Equations (FAEs) and Adjoint Boundary Conditions (ABCs). The extra unknown flow variable (a_l) gives rise to an additional adjoint variable (ψ_5) and PDE. Furthermore, the cavitation model needed also to be differentiated to incorporate it in the adjoint problem, resulting in new source terms. The development of a two–phase flow model, and its adjoint, provided flexibility and enabled the investigation and optimization using a two–phase specific objective function (J_V). The above components were vital to obtain a consistent adjoint problem, capable of accurately computing objective function gradients.

The optimization problems studied in this thesis are two–fold. First, the developed continuous adjoint method was integrated into a Shape Optimization (ShpO) framework to optimize baseline shapes using the CC method. In the ShpO runs performed, the objective function showed good convergence, optimizing the baseline shape in all cases presented. For example, in the single–phase turbulent flows, the optimized design of the initially straight channel

has 40% reduced volume–averaged total pressure losses, while the NACA 0012 case targeting lift maximization led to a design that induces 10× more lift. These cases also illustrated the main benefit of introducing the CC method to a ShpO framework: the ability to perform large displacements. Regarding two–phase ShpO runs, the cases targeting the minimization of the vapour present in the domain lead to practically cavitation–free designs. Simultaneously, they showed that lift force evolution follows a similar trend with the presence of vapour in each new design. This trend, however, is not observed when the maximization of the lift force is targeted. The second part dealt with the development of a Topology Optimization (TopO) framework, fundamentally built on the ability of the CC method to enforce accurate boundary conditions during the optimization process, without mass and momentum loss. Thus, a TopO problem was formulated based on design variables that can describe the fluid–solid interfaces and allows for topological changes to occur. Several observations were made during the comparisons between the porosity– and CC–based TopO methods. Firstly, it was made clear that porosity–based optimized designs need be re–evaluated on body–fitted meshes to accurately compute the concerned quantity of interest. Therefore, the developed utility tool that can re–evaluate porosity–based optimized solutions and provide an STL description of the solid boundaries can single–handedly be of great value. Secondly, these comparisons showed that CCTopO could provide better–performing designs with reduced, by approximately 8% (Single Inlet–Single Outlet) and 2% (Single Inlet–Two Outlet), objective function values, when compared to the re–evaluated ones at the expense of additional computational cost. In the Single Inlet–Two Outlet case, the obtained design is more inline with results obtained after a ShpO is performed starting from the topologically porosity–optimized design. Furthermore, CCTopO was successfully applied on benchmark TopO problems demonstrating its versatility and a large–scale $3D$ case that featured approximately $2.2M$ design variables. Hence, the CC–based TopO was shown to be a versatile competitor/alternative of porosity–based method that can also further help extend TopO applications to problems where near–walls effects are of paramount importance.

7.1 Novel Contributions

Novel contributions of this PhD thesis are:

- The simulation of two–phase flows that feature cavitation effects is modeled using a TEM with a cavitation model, based on a homogeneous mixture. The derivation and development of the continuous adjoint equations for $2D$ and $3D$ problems is novel and was

presented for the first time on isolated hydrofoils [266]. Furthermore, its implementation using the CC method may also be considered new.

- The development of the continuous adjoint to the $k-\varepsilon$ turbulence model was primarily based on its incompressible formulation, avoiding underlying density or viscosity variations. The presented development includes these variations, that are directly expressed w.r.t. the liquid volume fraction due to their closed–form expression.

- Another novel contribution, arising from the differentiation of the turbulence model, was the development of the *adjoint linelet* technique (the analogous to the adjoint wall–functions concept for body–fitted formulations [191, 283]) that were implemented for the CC method.

- The development of a highly accurate CC–based TopO method that reconstructs the fluid–solid interface without the need of solving a Hamilton–Jacobi Equation (HJE) is novel and was presented for the first time [267].

- A CC–based utility tool that can post–process optimized porosity fields to provide re–evaluations or STL descriptions of the resulting body surface.

Publications and Conference Presentations

- Vrionis, Y.P., Samouchos, K.D. & Giannakoglou, K.C. (2021). Topology optimization in fluid mechanics using continuous adjoint and the cut-cell method. *Computers & Mathematics with Applications*, **97**, 286–297

- Vrionis, Y.P., Samouchos, K.D. & Giannakoglou, K.C. (2021). The continuous adjoint cut-cell method for shape optimization in cavitating flows. *Computers & Fluids*, **224**, 104974

- Vrionis, Y.P., Samouchos, K.D. & Giannakoglou, K.C. (2019). Implementation of a conservative cut–cell method for the simulation of two–phase cavitating flows. In *10th International Conference on Computational Methods (ICCM2019)*, 440–452, Singapore

7.2 Future Work

The CC method is seen to have significant strengths and potency that can be utilized to provide sophisticated, versatile, general–purpose CFD and numerical optimization tools. In the

following, some areas that may be investigated further, based on the experience of the current work, are listed below:

- The extension to different high Reynolds turbulence models, such as the Spalart–Allmaras, that has seen extensive application in PCOpt, or other k–family turbulence models could be pursued. On a related note, the use of more sophisticated wall models, such as the ODE–based one implemented in Berger & Aftosmis [32], that can further alleviate wall functions limitations could be implemented.

- The extension to unsteady cavitating phenomena could exploit the benefits of the CC method (such as cloud cavitation, being an unsteady phenomenon). The CC method utilizing flow adaptation techniques could provide a versatile tool to study such phenomena.

- The extension to unsteady two–phase/turbulent flows using the CC method with moving boundaries, developed in a recently accomplished thesis [214], could be an engaging area of research. The study of such complex applications could further showcase the capabilities of the developed software.

- The extension of TopO methods to turbulent flows is also challenging due to the difficulty of porosity–based methods to capture near–wall effects. The application of CCTopO to turbulent flows could be pursued.

- Building on the last suggestion, the application of CCTopO to two–phase flows, with different objective functions, expressed directly at the solid walls, or multidisciplinary applications such as conjugate heat transfer problems [83] or FSI should be formulated.

- Based on relative work in PCOpt [255], the integration of shape parameterization techniques for $3D$ problems, such as NURBS surfaces, can allow for the ShpO of parameterized $3D$ applications that ensure the smoothness of the optimized design.

- The CC software could also benefit from simulation time speedup. These could arise from geometric multigrid methods that can accelerate flow solution convergence rates or by programming the corresponding solvers on Graphics Processing Units (GPUs). The existing experience at PCOpt could aid in materializing the latter more easily [255].

Appendix A

Inviscid Jacobian and Eigenvectors

A.1 Single–Phase Governing Equations

Considering the system of equation for $3D$ single–phase flows of incompressible fluids in Eq. (3.4), with $\boldsymbol{U} = \left[\breve{p}\ u_1\ u_2\ u_3\right]^T$ and $\boldsymbol{f}_j^I = \left[u_j\ u_1u_j + \delta_j^1\breve{p}\ u_2u_j + \delta_j^2\breve{p}\ u_3u_j + \delta_j^3\breve{p}\right]^T$.

The inviscid Jacobian matrix is obtained as $\boldsymbol{A}_j^{1\phi} = \frac{\partial \boldsymbol{f}_j^I}{\partial \boldsymbol{U}}$, leading to

$$\boldsymbol{A}_j{}^{1\phi} = \begin{bmatrix} 0 & \delta_j^1 & \delta_j^2 & \delta_j^3 \\ \delta_j^1 & u_j + u_1\delta_j^1 & u_1\delta_j^2 & u_1\delta_j^3 \\ \delta_j^2 & u_2\delta_j^1 & u_j + u_2\delta_j^2 & u_2\delta_j^3 \\ \delta_j^3 & u_3\delta_j^1 & u_3\delta_j^2 & u_j + u_3\delta_j^3 \end{bmatrix} \tag{A.1}$$

The right and left eigenvector matrices $\boldsymbol{M}$, $\boldsymbol{M}^{-1}$ that diagonalize the Jacobian matrix $\boldsymbol{A}^{\boldsymbol{\Gamma}}{}_j n_j = \boldsymbol{M}\boldsymbol{\Lambda}\boldsymbol{M}^{-1}$ are given by

$$\boldsymbol{M}^{1\phi} = \begin{bmatrix} 0 & 0 & c & -c \\ t_1^I & t_1^{II} & n_1 + \frac{u_1\lambda_3}{\beta^2} & n_1 + \frac{u_1\lambda_4}{\beta^2} \\ t_2^I & t_2^{II} & n_2 + \frac{u_2\lambda_3}{\beta^2} & n_2 + \frac{u_2\lambda_4}{\beta^2} \\ t_3^I & t_3^{II} & n_3 + \frac{u_3\lambda_3}{\beta^2} & n_3 + \frac{u_3\lambda_4}{\beta^2} \end{bmatrix}, \tag{A.2}$$

$$\left[\boldsymbol{M}^{-1}\right]^{1\phi} = \frac{1}{c^2}\begin{bmatrix} \boldsymbol{w}\cdot\mathbf{n} & \beta^2 t_1^I + u_n w_1 & \beta^2 t_2^I + u_n w_2 & \beta^2 t_3^I + u_n w_3 \\ -\boldsymbol{v}\cdot\mathbf{n} & \beta^2 t_1^{II} - u_n v_1 & \beta^2 t_2^{II} - u_n v_2 & \beta^2 t_3^{II} - u_n v_3 \\ \frac{1}{2}(\ c - u_n) & \frac{\beta^2}{2}n_1 & \frac{\beta^2}{2}n_2 & \frac{\beta^2}{2}n_3 \\ \frac{1}{2}(-c - u_n) & \frac{\beta^2}{2}n_1 & \frac{\beta^2}{2}n_2 & \frac{\beta^2}{2}n_3 \end{bmatrix} \tag{A.3}$$

where $\lambda_{3,4} = u_n \pm c$, $c = \sqrt{u_n^2 + \beta^2 n_j n_j}$ is the artificial speed of sound, $\boldsymbol{v} = \hat{\mathbf{t}}^I \times \boldsymbol{u}$, and $\boldsymbol{w} = \hat{\mathbf{t}}^{II} \times \boldsymbol{u}$.

The unit vectors $\hat{\mathbf{t}}^I$, $\hat{\mathbf{t}}^{II}$ in Eqs. (A.2), (A.3) satisfy

$$\hat{\mathbf{t}}^I \times \hat{\mathbf{t}}^{II} = \hat{\mathbf{n}} \tag{A.4}$$

A.2 Two–Phase Governing Equations

For the system of equation for $3D$ two–phase flows of incompressible constituents in Eq. (3.70), with $\boldsymbol{U} = \begin{bmatrix} p & u_1 & u_2 & u_3 & a_l \end{bmatrix}^T$ and $\boldsymbol{f}_j^I = \begin{bmatrix} u_j & \varrho_m u_1 u_j + \delta_j^1 p & \varrho_m u_2 u_j + \delta_j^2 p & \varrho_m u_3 u_j + \delta_j^3 p & a_l u_j \end{bmatrix}^T$, the inviscid Jacobian matrix $\boldsymbol{A}_j^{2\phi} = \frac{\partial \boldsymbol{f}_j^I}{\partial \boldsymbol{U}}$ is given as

$$\boldsymbol{A}_j{}^{2\phi} = \begin{bmatrix} 0 & \delta_j^1 & \delta_j^2 & \delta_j^3 & 0 \\ \delta_j^1 & \varrho_m \left(u_j + u_1 \delta_j^1\right) & \varrho_m u_1 \delta_j^2 & \varrho_m u_1 \delta_j^3 & u_1 u_j \Delta\varrho \\ \delta_j^2 & \varrho_m u_2 \delta_j^1 & \varrho_m \left(u_j + u_2 \delta_j^2\right) & \varrho_m u_2 \delta_j^3 & u_2 u_j \Delta\varrho \\ \delta_j^3 & \varrho_m u_3 \delta_j^1 & \varrho_m u_3 \delta_j^2 & \varrho_m \left(u_j + u_3 \delta_j^3\right) & u_3 u_j \Delta\varrho \\ 0 & \delta_j^1 a_l & \delta_j^2 a_l & \delta_j^3 a_l & u_j \end{bmatrix} \tag{A.5}$$

where $\Delta\varrho = \rho_l - \rho_v$ and subscripts m, l, v refer to the mixture, liquid and vapour, respectively.

The corresponding right and left eigenvector matrices are obtained using

$$\boldsymbol{M}^{2\phi} = \begin{bmatrix} \left[\boldsymbol{K}\right]^{-1} \left[\boldsymbol{M}^{1\phi}\right] & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \qquad \left[\boldsymbol{M}^{2\phi}\right]^{-1} = \begin{bmatrix} \left[\boldsymbol{M}^{1\phi}\right]^{-1} \left[\boldsymbol{K}\right] & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{A.6}$$

where

$$\boldsymbol{K} = \begin{bmatrix} \frac{1}{\rho_m} & 0 \\ 0 & \boldsymbol{I}_{4\times 4} \end{bmatrix} \tag{A.7}$$

Appendix B

Adjoint Wall Functions and Linelets in the Cut–Cell Method

In this Appendix, the necessary development of the boundary term containing the derivative of the stress tensor w.r.t. the design variables is presented, Eq. (4.38). In the governing equations, this term is computed using the wall function formulation presented in Section 3.2 and, therefore, needs to be differentiated accordingly to formulate the adjoint problem.

To proceed with the development of the said term, several variables, auxiliary projection matrices and total differentials must be defined specifically for the wall functions method, as implemented in the CC method. In detail, the velocity vector at the forcing points is extrapolated using the Taylor expansion from the cell center of the Cartesian cell it lies in, i.e. $u_k^F = u_k^C + \frac{\partial u_k}{\partial x_j}|^C \left(x_j^F - x_j^C\right)$ with superscript C referring to the Cartesian cell and F to the forcing point.

The projection of the forcing point velocity vector $\boldsymbol{u}^F$ along the solid wall tangential direction $\hat{\mathbf{t}}$ is computed by decomposing it into parallel and perpendicular, to the solid wall, vector components $\boldsymbol{u}^F = \boldsymbol{u}_{\parallel}{}^F + \boldsymbol{u}_{\perp}{}^F$ and subtracting its perpendicular vector component, i.e.

$$u_{\parallel}{}_j^F = u_j^F - \left(u_k^F \hat{n}_k\right) \hat{n}_j = \left(\delta_j^k - \hat{n}_k \hat{n}_j\right) u_j^F = \mathbb{P}_{kj} u_j^F \tag{B.1}$$

$$\hat{t}_j = \frac{u_{\parallel}{}_j^F}{\sqrt{u_{\parallel}{}_k^F u_{\parallel}{}_k^F}} \tag{B.2}$$

Thus, the (scalar) component of $\boldsymbol{u}^F$ with respect to $\hat{\mathbf{t}}$ provides the tangential velocity u_t, which

can be equivalently computed using

$$u_t = u_j^F \hat{t}_j \equiv u_j^F \frac{u_{||_j}}{\sqrt{u_{||_k} u_{||_k}}} \equiv \sqrt{u_{||_k} u_{||_k}} \tag{B.3}$$

Aiming to differentiate the law of the wall, the total differentials of Eqs. (B.3), (B.2), and (B.1) are sought. Since $u_t = u_t\left(\boldsymbol{u}^{\boldsymbol{F}}, \hat{\mathbf{t}}\right)$, $\hat{\mathbf{t}} = \hat{\mathbf{t}}\left(\boldsymbol{u}_{\parallel}{}^{\boldsymbol{F}}\right)$ and $\boldsymbol{u}_{\parallel}{}^{\boldsymbol{F}} = \boldsymbol{u}_{\parallel}{}^{\boldsymbol{F}}\left(\boldsymbol{u}^{\boldsymbol{F}}, \hat{\mathbf{n}}\right)$, their total differentials respectively yield

$$\delta u_t = \delta u_j^F \hat{t}_j + u_j^F \delta \hat{t}_j \tag{B.4}$$

$$\delta \hat{t}_i = \frac{1}{\sqrt{u_{\parallel}{}_j^F u_{\parallel}{}_j^F}} \left(\delta_j^i - \hat{t}_i \hat{t}_j\right) \delta u_{\parallel}{}_j^F = \mathbb{M}_{ij} \delta u_{\parallel}{}_j^F \tag{B.5}$$

$$\delta u_{\parallel}{}_i^F = \left(\delta_j^i - \hat{n}_i \hat{n}_j\right) \delta u_j^F - \left(u_k^F \hat{n}_k \delta_j^i + \hat{n}_i u_j^F\right) \delta \hat{n}_j = \mathbb{P}_{ij} \delta u_j^F - \mathbb{Q}_{ij} \delta \hat{n}_j \tag{B.6}$$

The total differential of the tangential velocity can further be expanded by combining the above equations and, after some rearrangement, leads to

$$\delta u_t = \left(\hat{t}_k + u_i^F \mathbb{M}_{ij} \mathbb{P}_{jk}\right) \delta u_k^F - u_i^F \mathbb{M}_{ij} \mathbb{Q}_{jk} \delta \hat{n}_k \tag{B.7}$$

The final step requires the differentiation of the friction velocity expression and, thus, the law of the wall. This is carried out by first identifying its independent variables. Reverting to Eqs. (3.44) and (B.7), the friction velocity can be written as

$$\mathrm{u}_\tau = f\left(\mathrm{u}^+, \mathrm{y}^+\right) = f\left(u_t, \nu_m, d^F\right) = f\left(u_t, \nu_m, d^F\right) = f\left(\boldsymbol{u}^F, \hat{\mathbf{n}}, a_l, d^F\right) \tag{B.8}$$

where $\mathrm{u}^+ = \frac{u_t}{\mathrm{u}_\tau}$, $\mathrm{y}^+ = \frac{d^F \mathrm{u}_\tau}{\nu}$ and $\nu_m = \frac{\mu_m}{\rho_m}$ is introduced to incorporate two–phase cases. However, in the CC method, the forcing points are positioned at a constant distance from the solid wall to create the linelets, axiomatically implying that $\delta d^F = 0$. Thus, the total differential of the friction velocity gives

$$\delta \mathrm{u}_\tau = \frac{\partial f}{\partial u_t} \frac{\partial u_t}{\partial u_j^F} \delta u_j^F + \frac{\partial f}{\partial u_t} \frac{\partial u_t}{\partial \hat{n}_j} \delta \hat{n}_j + \frac{\partial f}{\partial \nu_m} \frac{1}{\varrho_m} \left(\Delta\mu - \nu_m \Delta\varrho\right) \delta a_l \tag{B.9}$$

The partial derivatives of f depend on the formula used to correlate u^+ with y^+. In the implemented method, the SA model is used, Eq. (3.44), solved using Eq. (3.47). The partial derivatives are obtained as

$$\frac{\partial f}{\partial u_t} = \frac{1}{G\mathrm{y}^+ + \mathrm{u}^+} \tag{B.10}$$

$$\frac{\partial f}{\partial \nu_m} = \frac{\left(\mathrm{y}^+\right)^2}{d^F} \frac{G}{G\mathrm{y}^+ + \mathrm{u}^+} \tag{B.11}$$

with

$$G = \frac{\partial \mathrm{u}^+}{\partial \mathrm{y}^+} = \frac{c_3 b_1 + 2c_1\left(\mathrm{y}^+ + a_1\right)}{\left(\mathrm{y}^+ + a_1\right)^2 + b_1^2} + \frac{c_4 b_2 - 2c_2\left(\mathrm{y}^+ + a_2\right)}{\left(\mathrm{y}^+ + a_2\right)^2 + b_2^2} \tag{B.12}$$

The wall viscous flux for the momentum equations is computed using the wall function technique as

$$\hat{t}_i \tau_{ij} \hat{n}_j = -\varrho_m \mathrm{u}_\tau^2 \tag{B.13}$$

The corresponding term appearing at the solid walls (S_W) can be re–written by defining the adjoint velocity $\Psi_k = \psi_{k+1},\ k = 1, 2, 3$ as

$$SI_{WF} \coloneqq -\int_{S_W} \psi_{k+1} \frac{\delta\left(\tau_{kj}\hat{n}_j\right)}{\delta b_i} dS = -\int_{S_W} \left(\Psi_n \hat{n}_k + \Psi_t \hat{t}_k + \Psi_z \hat{z}_k\right) \frac{\delta\left(\tau_{kj}\hat{n}_j\right)}{\delta b_i} dS \tag{B.14}$$

where $\Psi_n = \boldsymbol{\Psi} \cdot \hat{\mathbf{n}}$, $\Psi_t = \boldsymbol{\Psi} \cdot \hat{\mathbf{t}}$, $\Psi_z = \boldsymbol{\Psi} \cdot \hat{\mathbf{z}}$ and $\hat{\mathbf{z}} = \hat{\mathbf{n}} \times \hat{\mathbf{t}}$. Further expanding each term yields

$$\begin{aligned} SI_{WF} = &-\int_{S_W} \Psi_n \left(\frac{\delta\left(\hat{n}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} - \tau_{kj}\hat{n}_j \frac{\delta \hat{n}_k}{\delta b_i} \right) dS - \int_{S_W} \Psi_t \left(\frac{\delta\left(-\varrho_m \mathrm{u}_\tau^2\right)}{\delta b_i} - \tau_{kj}\hat{n}_j \frac{\delta \hat{t}_k}{\delta b_i} \right) dS \\ &- \int_{S_W} \Psi_z \left(\frac{\delta\left(\hat{z}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} - \tau_{kj}\hat{n}_j \frac{\delta \hat{z}_k}{\delta b_i} \right) dS \end{aligned} \tag{B.15}$$

and can further be developed to obtain contributions to the ABCs and SDs as

$$\begin{aligned} SI_{WF} = &-\int_{S_W} \Psi_n \frac{\delta\left(\hat{n}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} dS - \int_{S_W} \Psi_z \frac{\delta\left(\hat{z}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} dS \\ &+ \int_{S_W} \tau_{kj}\hat{n}_j \left(\Psi_n \frac{\delta \hat{n}_k}{\delta b_i} + \Psi_t \frac{\delta \hat{t}_k}{\delta b_i} + \Psi_z \frac{\delta \hat{z}_k}{\delta b_i} \right) dS \\ &+ \int_{S_W} \Psi_t \left(\mathrm{u}_\tau^2 \Delta\varrho \frac{\delta a_l^F}{\delta b_i} + 2\mathrm{u}_\tau \varrho_m \frac{\delta \mathrm{u}_\tau}{\delta b_i} \right) dS \\ = &\underbrace{-\int_{S_W} \Psi_n \frac{\delta\left(\hat{n}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} dS}_{ABCs} \underbrace{- \int_{S_W} \Psi_z \frac{\delta\left(\hat{z}_k \tau_{kj} \hat{n}_j\right)}{\delta b_i} dS}_{ABCs} \\ &\underbrace{+ \int_{S_W} \tau_{kj}\hat{n}_j \left(\Psi_n \frac{\delta \hat{n}_k}{\delta b_i} + \Psi_t \frac{\delta \hat{t}_k}{\delta b_i} + \Psi_z \frac{\delta \hat{z}_k}{\delta b_i} \right) dS}_{SDs} \end{aligned}$$

$$
\begin{aligned}
&\underbrace{+\int_{S_W} \mathrm{u}_\tau \Psi_t \left[\mathrm{u}_\tau \Delta\varrho + 2\mathrm{u}_\tau \frac{\partial \mathrm{u}_\tau}{\partial \nu_m}\left(\Delta\mu - \nu_m \Delta\varrho\right)\right] \frac{\partial a_l^F}{\partial b_i} dS}_{ST} \\
&\underbrace{-\int_{S_W} \mathrm{u}_\tau \Psi_t \left[\mathrm{u}_\tau \Delta\varrho + 2\mathrm{u}_\tau \frac{\partial \mathrm{u}_\tau}{\partial \nu_m}\left(\Delta\mu - \nu_m \Delta\varrho\right)\right] \frac{\partial a_l^F}{\partial x_l}\frac{\delta x_l}{\delta b_i} dS}_{SDs} \\
&\underbrace{+\int_{S_W} \mathcal{B}_{WF}\left(\hat{t}_k + u_i^F \mathbb{M}_{ij}\mathbb{P}_{jk}\right)\frac{\partial u_k^F}{\partial b_i} dS}_{ST} \\
&\underbrace{-\int_{S_W} \mathcal{B}_{WF}\left(\hat{t}_k + u_i^F \mathbb{M}_{ij}\mathbb{P}_{jk}\right)\frac{\partial u_k^F}{\partial x_l}\frac{\delta x_l}{\delta b_i} dS}_{SDs} \underbrace{-\int_{S_W} \mathcal{B}_{WF} u_i^F \mathbb{M}_{ij}\mathbb{Q}_{jk}\frac{\delta \hat{n}_k}{\delta b_i} dS}_{SDs}
\end{aligned}
\tag{B.16}
$$

with $\mathcal{B}_{WF} = 2\mathrm{u}_\tau \varrho_m \frac{\partial \mathrm{u}_\tau}{\partial u_t}\Psi_t$.

Overall, the differentiation of the wall function technique results in additional source terms (ST) at the Cartesian cells that encompass the forcing points and SD contributions that are computed using the geometric derivatives of the boundary face each linelet originates from.

Appendix C

The GCMMA Algorithm

The Method of Moving Asymptotes (MMA) [245], upon which the GCMMA variant is built, is an iterative method used to solve constrained optimization problems by replacing the objective and constraint functions at the current iteration point with strictly convex, explicit approximations and, instead, solves these subproblems. These have more desirable properties, for instance, the existence of a unique, optimal solution, and can be efficiently solved using quadratic programming. However, this requires the computation of the objective function Hessian matrix, namely its second–order derivatives, that requires significant computational effort, especially when a large number of design variables is considered. The Hessian matrix can be approximated by accumulating gradient information from previous iterations points (SQP methods). Contrarily, in MMA the objective function curvature approximation is controlled by introducing lower and upper *asymptotes*, that are based on the feasible solution space (lower and upper bounds).

In Figure C.1, several different convex approximations of a univariant objective function (blue line, $f(x)$) are depicted, created based on the same iteration point, and, thus, objective function value and gradient. As observed, the distance between the asymptotes changes the validity of the approximation near the iteration point and, ultimately, the optimizer behavior. Larger distances between the asymptotes lead to smaller curvature approximation and, thus, larger steps. Conversely, smaller distances lead to more conservative behaviors. During the optimization iterations, the asymptotes position is adjusted to facilitate convergence. For example, if stabilization is needed due to objective function oscillations, the asymptotes converge towards the iteration point, whereas if a slow evolution is observed, more aggressive behavior is opted by moving them away from it. This concept is illustrated in Figure C.2.

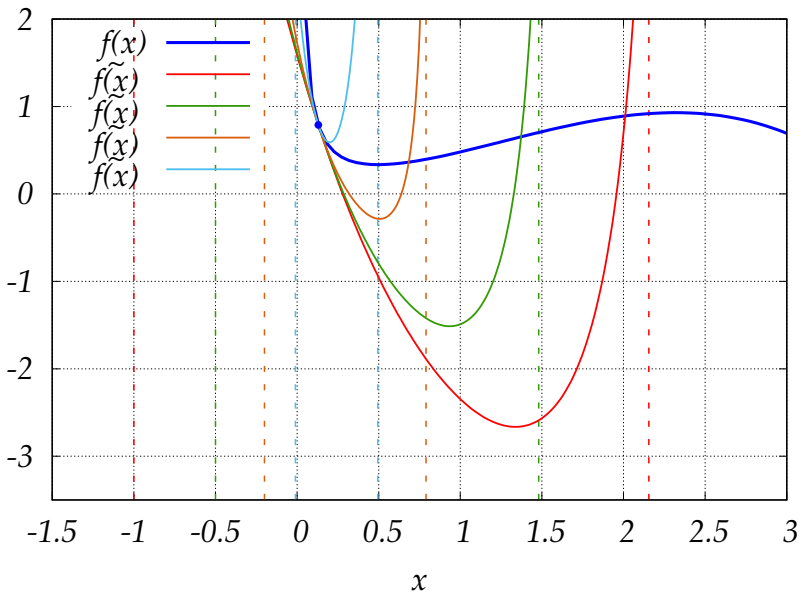


Figure C.1: MMA method - Example of different convex approximations of the objective function (blue line). The vertical dashed lines correspond to the corresponding upper and lower asymptotes' position.

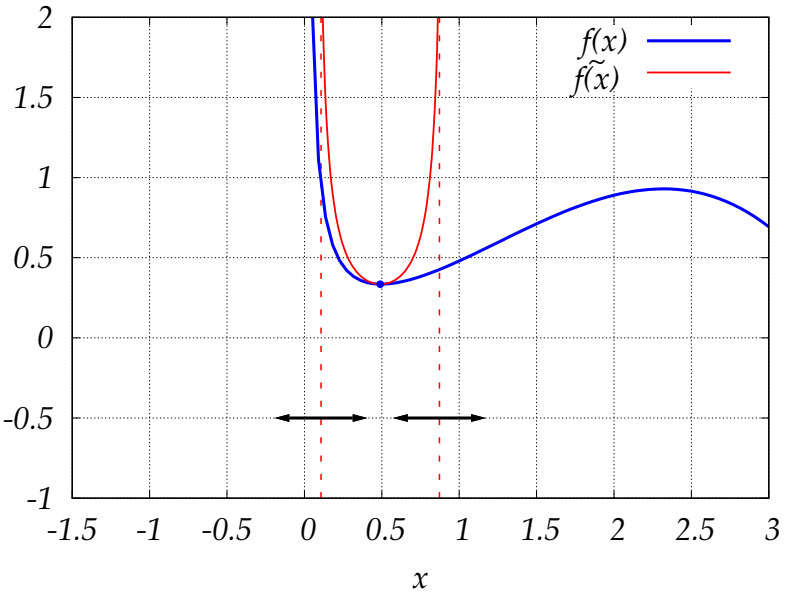


Figure C.2: MMA method - Concept of adapting the asymptotes position, based on the convergence behavior of the objective function.

C.1 (GC)MMA Implementation Details

This section describes in brief the basic implementation details regarding the GCMMA [247] employed to solve constrained optimization problems of the form

$$\begin{aligned} \text{min} \quad & f_0(b_j), \qquad j = 0, ..., N \\ \text{subject to} \quad & f_i(b_j) \leq 0, \qquad i = 1, ..., M \\ & b_j^{min} \leq b_j \leq b_j^{max} \end{aligned} \tag{C.1}$$

where $f_0(\boldsymbol{b})$ is the objective function, $f_i(\boldsymbol{b})$ are the $M-1$ constraint functions, $\boldsymbol{b}$ the design variables of size N, and $\boldsymbol{b}^{max/min}$ their corresponding upper and lower bounds. This form, however, does not restrict its range of applications, since maximization problems can be treated by minimizing $-f_0(\boldsymbol{b})$, *logical* dummy bounds can be the user for unbounded design variables based on the user intuition, and equality constraints can be satisfied by introducing lower and upper limits, i.e. $-\epsilon \leq f_i(\boldsymbol{b}) \leq \epsilon \implies f_i(\boldsymbol{b}) \approx 0$, where the value of ϵ should be carefully chosen to avoid convergence stall.

The GCMMA algorithm is a variant of MMA that employs both inner and outer iterations to solve the constrained optimization problem of Eq. (C.1) by approximating the objective and constraint functions with certain convex functions $\tilde{f}(\boldsymbol{b})$ and introducing several shadow parameters (artificial optimization variables). The additional inner iterations, compared to MMA, are used to further refine approximating functions that can lead to an optimal solution that violates a constraint or leads to an increase in the objective function, by making a more conservative approximation. If no violation occurs, the GCMMA is simply reduced to the MMA algorithm. The GCMMA subproblem reads

$$\begin{aligned} \text{min} \quad & \tilde{f}_0^{(k,v)}(b_j) + a_0 z + \sum_{i=1}^{M} \left(c_i y_i + \frac{1}{2} d_i y_i^2 \right), \qquad j = 0, ..., N \\ \text{subject to} \quad & \tilde{f}_i^{(k,v)}(b_j) - a_i z - yi \leq 0, \qquad i = 1, ..., M \\ & \alpha_j^{(k)} \leq b_j \leq \beta_j^{(k)} \\ & y_i \geq 0 \\ & z \geq 0. \end{aligned} \tag{C.2}$$

where k and v denote the outer and inner iteration, and a_i, c_i, d_i and z the positive artificial parameters.

The approximating functions $f_i^{(k,v)}(b_j)$ are computed as

$$\tilde{f}_i^{(k,v)}(b_j) = \sum_{j=1}^{N} \left(\frac{p_{ij}^{(k,v)}}{u_j^{(k)} - b_j} + \frac{q_{ij}^{(k,v)}}{b_j - l_j^{(k)}} \right) + r_i^{(k,v)}, \qquad i = 0, 1, ..., M \tag{C.3}$$

with

$$p_{ij}^{(k,v)} = \left(u_j^{(k)} - b_j^{(k)} \right)^2 \left[1.001 \ \phi^+ + 0.001 \ \phi^- + \frac{\rho_i^{(k,v)}}{b_j^{max} - b_j^{min}} \right] \tag{C.4}$$

$$q_{ij}^{(k,v)} = \left(b_j^{(k)} - l_j^{(k)} \right)^2 \left[0.001 \ \phi^+ + 1.001 \ \phi^- + \frac{\rho_i^{(k,v)}}{b_j^{max} - b_j^{min}} \right] \tag{C.5}$$

$$r_i^{(k,v)} = f_i \left(\boldsymbol{b}^{(k)} \right) - \sum_{j=1}^{N} \left(\frac{p_{ij}^{(k,v)}}{u_j^{(k)} - b_j} + \frac{q_{ij}^{(k,v)}}{b_j - l_j^{(k)}} \right) \tag{C.6}$$

$$\phi^+ = \max \left(\frac{\partial f_i}{\partial b_j} \left(\boldsymbol{b}^{(k)} \right), \ 0 \right) \tag{C.7}$$

$$\phi^- = \max \left(-\frac{\partial f_i}{\partial b_j} \left(\boldsymbol{b}^{(k)} \right), \ 0 \right) \tag{C.8}$$

The design variables bounds $\alpha_j^{(k)}$ and $\beta_j^{(k)}$ and lower and upper asymptotes $l_j^{(k)}$ and $u_j^{(k)}$ are updated in each outer iteration k as

$$\alpha_j^{(k)} = \max \left\{ b_j^{min}, \ l_j^{(k)} + 0.1 \left(b_j^{(k)} - l_j^{(k)} \right), \ b_j^{(k)} - 0.5 \left(b_j^{max} - b_j^{min} \right) \right\} \tag{C.9}$$

$$\beta_j^{(k)} = \max \left\{ b_j^{max}, \ u_j^{(k)} - 0.1 \left(u_j^{(k)} - b_j^{(k)} \right), \ b_j^{(k)} + 0.5 \left(b_j^{max} - b_j^{min} \right) \right\} \tag{C.10}$$

$$l_j^{(k)} = \begin{cases} b_j^{(k)} - 0.5 \left(b_j^{max} - b_j^{min} \right), & k < 3, \\ b_j^{(k)} - \gamma_j^{(k)} \left(b_j^{k-1} - l_j^{k-1} \right), & otherwise \end{cases} \tag{C.11}$$

$$u_j^{(k)} = \begin{cases} b_j^{(k)} + 0.5 \left(b_j^{max} - b_j^{min} \right), & k < 3 \\ b_j^{(k)} + \gamma_j^{(k)} \left(b_j^{k-1} - l_j^{k-1} \right), & otherwise \end{cases} \tag{C.12}$$

where

$$\gamma_j^{(k)} = \begin{cases} 0.7, & \left(b_j^{(k)} - b_j^{(k-1)} \right) \left(b_j^{(k-1)} - b_j^{(k-2)} \right) < 0, \\ 1.2, & \left(b_j^{(k)} - b_j^{(k-1)} \right) \left(b_j^{(k-1)} - b_j^{(k-2)} \right) > 0, \\ 1, & otherwise \end{cases} \tag{C.13}$$

Finally, term $\rho_i^{(k,v)}$ in Eqs. (C.4) and (C.5) is computed during the first inner iteration as

$$\rho_i^{(k,0)} = \max \left(\frac{0.1}{N} \sum_{j=1}^{N} \left[\left(\left| \frac{\partial f_i}{\partial b_j} \left(\boldsymbol{b}^{(k)} \right) \right| \left(b_j^{max} - b_j^{min} \right) \right) \right], \ 10^{-6} \right) \tag{C.14}$$

and subsequently using

$$\rho_i^{(k,v+1)} = \begin{cases} \min\left\{1.1\left(\rho_i^{(k,v)} + \delta_i^{(k,v)}\right), 10\rho_i^{(k,v)}\right\}, & \delta_i^{(k,v)} > 0 \\ \rho_i^{(k,v)}, & otherwise \end{cases} \tag{C.15}$$

$$\delta_i^{(k,v)} = \frac{f_i\left(\boldsymbol{b}^{*(k,v)}\right) - \tilde{f}_i\left(\boldsymbol{b}^{*(k,v)}\right)}{d^{(k)}\left(\boldsymbol{b}^*\right)^{(k,v)}} \tag{C.16}$$

$$d^{(k)}(\boldsymbol{b}) = \sum_{j=1}^{N}\left[\frac{\left(u_j^{(k)} - l_j^{(k)}\right)\left(b_j - b_j^{(k)}\right)^2}{\left(u_j^{(k)} - b_j\right)\left(b_j - l_j^{(k)}\right)\left(b_j^{max} - b_j^{min}\right)}\right] \tag{C.17}$$

In each inner iteration, a dual interior point method is employed in which the relaxed Karush—Kuhn—Tucker (KKT) conditions are solved using a Newton method to obtain the subproblem optimal solution $\boldsymbol{b}^*$. The condition to terminate inner iterations is $\tilde{f}_i(\boldsymbol{b}^*) \geq f_i(\boldsymbol{b}^*)$. In case this does not hold, $\delta_i^{(k,v)} \leq 0$ holds (Eq. (C.16)), and, thus, more conservative approximate functions are computed for the violated cases using the updated ρ_i value in Eq. (C.3).

Overall, each outer GCMMA iteration requires the computation of a single set of objective and constraint function gradients, whereas each inner iteration requires the re–evaluation of the objective and constraint functions. In most cases, a small number of inner iterations is required, reducing the associated cost. Comparing GCMMA with MMA, the use of inner iterations results in higher computational cost per objective function gradient evaluation. However, it is more robust and allows for better adapatation of the asymptotes that usually leads to a faster convergence.

Appendix D

Re–evaluation of the Porosity–based Optimized Solutions

This appendix is concerned with the method followed to re–evaluate SPTopO optimized solutions on automatically generated CC meshes. Successful termination of an SPTopO run results in an optimized porosity field $\boldsymbol{\alpha}_b^*$ that determines the permeability of each cell and, therefore, the fluid and solid regions. However, it does not define an explicit fluid–solid interface. In order to re–evaluate the solution, must clearly be defined. Here, the same principle, as with the CCTopO, is followed to construct the fluid–solid interfaces. Essentially, the optimized porosity field is mapped onto *nodal* boundary indicator variables $\boldsymbol{\phi}$ of a background mesh, and the CC mesh generation process, Section 6.4, is used to create the CC mesh. Thus, in the following, the mapping of an arbitrary set of N known data points of the form $\left\{\left(\mathbf{x}_1, \alpha_{b\,1}^*\right), ..., \left(\mathbf{x}_N, \alpha_{b\,N}^*\right)\right\}$ onto the M nodal coordinates of a background mesh, denoted as $\boldsymbol{x}_i, i = 1, ..., M$, is presented, assuming non–coinciding, unordered points of meshes ($\{\mathbf{x}_1, ..., \mathbf{x}_N\} \neq \{\boldsymbol{x}, ..., \boldsymbol{x}_M\}$).

The interpolated nodal porosity values are computed using Shepard's Inverse Distance Weighting (IDW) approach [229] with only the K nearest neighbors that lie within an r–*Sphere*, to avoid computing pairwise distances between the N known data points and M nodal coordinates. The porosity value at the nodal coordinate is, thus, computed as

$$\alpha_b^*(\boldsymbol{x}_m) = \begin{cases} \frac{\sum_k^K w_k \alpha_{b\,k}^*}{\sum_k^K w_k} & \mathrm{d}\left(\boldsymbol{x}_m, \mathbf{x}_k\right) \leq R \\ \alpha_{b\,k}^*, & \mathrm{d}\left(\boldsymbol{x}_m, \mathbf{x}_k\right) \leq \epsilon \end{cases}, \qquad w_k = \left(\frac{R - \mathrm{d}\left(\boldsymbol{x}_m, \mathbf{x}_k\right)}{R\,\mathrm{d}\left(\boldsymbol{x}_m, \mathbf{x}_k\right)}\right)^2 \tag{D.1}$$

where ϵ is a user–defined tolerance that assumes the two points coincide and R the r–sphere radius.

Eq. (D.1) requires performing a k–Nearest Neighbors Search (k–NNS) for each node to find the data points that lie within the specified r–sphere. This is efficiently achieved using a $k-d(imensional)$ tree [30] that creates cloud points of data points lying in close proximity*. As such, the nearest neighbors of each node are focused on a small subset of the entire data point leading to very fast searches. Then, Eq. (D.1) can be used to compute the optimized porosity values at each node of the background mesh. What remains is mapping the nodal porosity values to boundary indicator variables (using Eqs. (6.11), (6.12)) to compute the intersection points, create the CCs and solve the governing equations on a body–fitted mesh. This process is identical to the one described in Section 6.3. The interpolation and mapping steps can massively be parallelized to accelerate the entire process. In the current implementation, this is achieved via OpenMP directives.

Algorithms 1 and 2 present the main aspects involved with creating the $k-d$ tree and the subsequent IDW interpolations of the data set to the nodal coordinates of the background mesh. In more detail, the $k-d$ tree is generated using the data points, having been associated with an index list. On each level, the domain splits into two half–spaces in alternate directions based on the average value of the included points, i.e. on level 1, the space splits along the x–axis, on level 2, along the y–axis, etc. On each level, the index list size splits in half and is recursively appointed to a sub–node, until a *limit* is reached. In the current implementation, a limit of 10 points is found to be adequate to avoid excessive tree searches in neighboring leaf nodes. However, this number depends on the resolution of the data points, background mesh and the prescribed radius. With the computed tree data structure, fast proximity searches are performed by traversing the tree nodes and comparing their average positions. Then, the leaf nodes that have part of the r–sphere encompassed within them are retained. At the final step, the distances of the unknown point are computed and stored to be used for the IDW interpolation of the nodal porosity values.

*In Section 6.3 a k–NNS is also performed. However, in that case, an Octree data structure is available to quickly traverse and find the candidate neighboring nodes.

Algorithm 1: $k-d$ tree construction

Input: *list of points* pointList, *list of indexes* idxList, *int* depth
Output: *list of nodes* nodeList
Global: k, limit

```
1  Function kdtree(pointList, idxList, depth)
2      // if the points in the list are less than the limit create a leafNode
3      if idxList.size() < limit then
4          return leafNode(idxList)
5      else
6          // find axis to split half-space
7          axis ← depth mod k
8          // compute average of points to split
9          x̄ ← computeAveragePositionOfPoints(idxList)
10         // find points on left and right half-space
11         leftPointsIdxList ←(indexes of points in pointList < x̄[axis])
12         rightPointsIdxList ←(indexes of points in pointList > x̄[axis])
13         // create splitNode and build subtree
14         splitNode.position ← x̄
15         splitNode.leftChild ← kdtree(pointList, leftPointsIdxList, depth+1)
16         splitNode.rightChild ← kdtree(pointList, rightPointsIdxList, depth+1)
17         return splitNode
18     end
19 end
```

Algorithm 2: Nodal Porosity computation

Input: *list of nodes* treeList, *list of tuples* dataPoints, *list of points* pList, *value* r
Output: *list of tuples* poroList

```
1  Function computeNodalPorosity(treeList, dataPoints, pList, r)
2      do in parallel
3          for x ∈ pList do
4              // traverse the tree to find a leafNodes that include part of the r-sphere
5              leafList ← kdtreeSearch(r, treeList, x)
6              // compute and store distances of dataPoints in leafNodes
7              for n ∈ leafList do  dList ← computeDistances(leafList, dataPoints, x)
8              // find the points of leafNodes that are inside the r-sphere
9              idxInRadiusList ← idxOfPointsInRadius(r, dList, leafList, x)
10             //interpolate porosity values at the unknown locations x
11             // IDW weights are computed based on dList (x is not needed)
12             poroInterpolated ← interpolateIDW(r, idxInRadiusList, dList, dataPoints)
13             // create pairs of (x, a) and add to list
14             poroList ← addToList(x, poroInterpolated)
15         end
16     end
17     return poroList
18 end
```

D. RE–EVALUATION OF THE POROSITY–BASED OPTIMIZED SOLUTIONS

Bibliography

[1] Abassi, W., Aloui, F., Ben Nasrallah, S., Keirsbulck, L. & Legrand, J. (2014). Flow behaviour around square and circular obstacles in 2D and 3D configurations using Lattice Boltzmann Method. In *Fluids Engineering Division Summer Meeting*, vol. 46223, V01BT14A004, American Society of Mechanical Engineers, Chicago, Illinois, U.S.A.

[2] Acosta, A.J. & Hamaguchi, H. (1967). Cavitation inception on the ITTC standard head form. Tech. Rep. E-149.1, California Institute of Technology.

[3] Aftosmis, M.J., Gaitonde, D. & Tavares, T.S. (1994). On the accuracy, stability, and monotonicity of various reconstruction algorithms for unstructured meshes. *AIAA Paper 94-0415*.

[4] Aftosmis, M.J., Berger, M.J. & Melton, J.E. (1998). *Adaptive Cartesian mesh generation*. Handbook of Mesh Generation (Contributed Chapter). CRC Press. ISBN 9780849326875.

[5] Aftosmis, M.J., Berger, M.J. & Melton, J.E. (1998). Robust and efficient Cartesian mesh generation for component-based geometry. *AIAA Journal*, **36**, 952–960.

[6] Ahuja, V., Hosangadi, A. & Arunajatesan, S. (2001). Simulations of cavitating flows using hybrid unstructured meshes. *Journal of Fluids Engineering*, **123**, 331–340.

[7] Alexias, P. & Giannakoglou, K.C. (2020). Optimization of a static mixing device using the continuous adjoint to a two-phase mixing model. *Optimization and Engineering*, **21**, 631–650.

[8] Allmaras, S.R. & Johnson, F.T. (2012). Modifications and clarifications for the implementation of the Spalart–Allmaras turbulence model. In *7th International Conference on Computational Fluid Dynamics (ICCFD7)*, 1–11, Big Island, Hawai, U.S.A.

[9] Aluru, S. & Sevilgen, F. (1997). Parallel domain decomposition and load balancing using space-filling curves. In *Fourth International Conference on High-Performance Computing*, 230–235, Bangalore, India.

[10] Amstutz, S. & Andrä, H. (2006). A new algorithm for topology optimization using a level-set method. *Journal of Computational Physics*, **216**, 573–588.

[11] Anagnostopoulos, J. (2007). A Cartesian grid method for the simulation of flows in complex geometries. In *3rd International Conference on Adaptive Modeling and Simulation, ADMOS*, Göteborg, Sweden.

[12] Anagnostopoulos, J.S. (2003). Discretization of transport equations on 2D Cartesian unstructured grids using data from remote cells for the convection terms. *International Journal for Numerical Methods in Fluids*, **42**, 297–321.

[13] Anderson, W.K. & Bonhaus, D.L. (1999). Airfoil design on unstructured grids for turbulent flows. *AIAA Journal*, **37**, 185–191.

[14] Anderson, W.K. & Venkatakrishnan, V. (1999). Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, **28**, 443–480.

[15] Anderson, W.K., Newman, J.C., Whitfield, D.L. & Nielsen, E.J. (2001). Sensitivity analysis for Navier-Stokes equations on unstructured meshes using complex variables. *AIAA Journal*, **39**, 56–63.

[16] Andreasen, C.S., Elingaard, M.O. & Aage, N. (2020). Level set topology and shape optimization by density methods using cut elements with length scale control. *Structural and Multidisciplinary Optimization*, **62**, 685–707.

[17] Anevlavi, D. & Belibassakis, K. (2021). An adjoint optimization prediction method for partially cavitating hydrofoils. *Journal of Marine Science and Engineering*, **9**.

[18] Angot, P., Bruneau, C.H. & Fabrie, P. (1999). A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, **81**, 497–520.

[19] Argonne National Laboratory Computer Science Mathematics Division and Rice University Center for Research on Parallel Computation (ADIFOR Homepage). `https://www.mcs.anl.gov/research/projects/adifor/`.

[20] Asnaghi, A., Feymark, A. & Bensow, R. (2017). Improvement of cavitation mass transfer modeling based on local flow properties. *International Journal of Multiphase Flow*, **93**, 142–157.

[21] Asouti, V., Zymaris, A., Papadimitriou, D. & Giannakoglou, K. (2008). Continuous and discrete adjoint approaches for aerodynamic shape optimization with low Mach number preconditioning. *International Journal for Numerical Methods in Fluids*, **57**, 1485–1504.

[22] Asouti, V.G. (2009). *Aerodynamic analysis and design methods at high and low speed flows, on multiprocessor platforms*. Ph.D. Thesis, National Technical University of Athens.

[23] Bai, W., Mingham, C., Causon, D. & Qian, L. (2016). Detached eddy simulation of turbulent flow around square and circular cylinders on Cartesian cut cells. *Ocean Engineering*, **117**, 1–14.

[24] Baker, T.J. (2005). Mesh generation: Art or science? *Progress in Aerospace Sciences*, **41**, 29–63.

[25] Barth, T. (1993). Recent developments in high order k-exact reconstruction on unstructured meshes. In *31st Aerospace Sciences Meeting*, 668, Reno, Nevada, U.S.A.

[26] Barth, T. (2003). Numerical methods and error estimation for conservation laws on structured and unstructured meshes. Tech. Rep., VKI Computational Fluid Dynamics Lecture Series.

[27] Bendsøe, M.P. & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, **71**, 197–224.

[28] Bendsøe, M.P. & Sigmund, O. (2011). *Topology optimization: Theory, methods, and applications*. Springer, Berlin, Heidelberg. ISBN 9783642076985.

[29] Bennett, W., Nikiforakis, N. & Klein, R. (2018). A moving boundary flux stabilization method for Cartesian cut-cell grids using directional operator splitting. *Journal of Computational Physics*, **368**, 333–358.

[30] Bentley, J.L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**, 509–517.

[31] Berger, M. & Aftosmis, M. (2012). Progress towards a Cartesian cut-cell method for viscous compressible flow. In *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 1301, Nashville, Tennessee, U.S.A.

[32] Berger, M.J. & Aftosmis, M.J. (2018). An ODE–based wall model for turbulent flow simulations. *AIAA Journal*, **56**, 700–714.

[33] Berger, M.J., Aftosmis, M.J. & Muman, S. (2005). Analysis of slope limiters on irregular grids. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 490, Reno, Nevada, U.S.A.

[34] Bertsekas, D.P. (1996). *Constrained optimization and Lagrange multiplier methods*. Athena Scientific. ISBN 9781886529045.

[35] Boger, D. (2013). *A continuous adjoint approach to design optimization in multiphase flow*. Ph.D. Thesis, The Pennsylvania State University.

[36] Boger, D.A. & Paterson, E.G. (2014). A continuous adjoint approach to design optimization in cavitating flow using a barotropic model. *Computers & Fluids*, **101**, 155–169.

[37] Borrvall, T. & Petersson, J. (2003). Topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Fluids*, **41**, 77–107.

[38] Brennen, C.E. (2013). *Cavitation and bubble dynamics*. Cambridge University Press. ISBN 9781107644762.

[39] Breugem, W.P. (2012). A second-order accurate immersed boundary method for fully resolved simulations of particle-laden flows. *Journal of Computational Physics*, **231**, 4469–4498.

[40] Brockett, T. & David Taylor Model Basin Hydromechanics Laboratory (1966). Minimum pressure envelopes for modified NACA-66 sections with NACA a = 0.8 camber and buships type I and type II sections. Tech. Rep. AD0629379, Defense Technical Information Center.

[41] Bruns, T.E. & Tortorelli, D.A. (2001). Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, **190**, 3443–3459.

[42] Bueno-Orovio, A., Castro, C., Palacios, F. & Zuazua, E. (2012). Continuous adjoint approach for the Spalart–Allmaras model in aerodynamic optimization. *AIAA Journal*, **50**, 631–646.

[43] BURGER, M. & OSHER, S.J. (2005). A survey on level set methods for inverse problems and optimal design. *European Journal of Applied Mathematics*, **16**, 263–301.

[44] Burman, E., Claus, S., Hansbo, P., Larson, M.G. & Massing, A. (2015). CutFEM: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, **104**, 472–501.

[45] Burman, E., Elfverson, D., Hansbo, P., Larson, M.G. & Larsson, K. (2019). Cut topology optimization for linear elasticity with coupling to parametric nondesign domain regions. *Computer Methods in Applied Mechanics and Engineering*, **350**, 462–479.

[46] Capizzano, F. (2011). Turbulent wall model for immersed boundary methods. *AIAA Journal*, **49**, 2367–2381.

[47] Capizzano, F. (2018). Automatic generation of locally refined Cartesian meshes: Data management and algorithms. *International Journal for Numerical Methods in Engineering*, **113**, 789–813.

[48] Carnarius, A., Thiele, F., Oezkaya, E. & Gauger, N.R. (2010). Adjoint approaches for optimal flow control. In *5th Flow Control Conference*, 5088, Chicago, Illinois, U.S.A.

[49] Challis, V.J. & Guest, J.K. (2009). Level set topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Engineering*, **79**, 1284–1308.

[50] Chan, W.M. (2009). Overset grid technology development at NASA AMES Research Center. *Computers & Fluids*, **38**, 496–503.

[51] Charlton, E.F. (1997). *An octree solution to conservation laws over arbitrary regions (OSCAR) with applications to aircraft aerodynamics*. Ph.D. Thesis, University of Michigan.

[52] Chen, Z.L., Hickel, S., Devesa, A., Berland, J. & Adams, N.A. (2014). Wall modeling for implicit large-eddy simulation and immersed-interface methods. *Theoretical and Computational Fluid Dynamics*, **28**, 1–21.

[53] Chien, K.Y. (1982). Predictions of channel and boundary–layer flows with a low–Reynolds–number turbulence model. *AIAA Journal*, **20**, 33–38.

[54] Chorin, A.J. (1967). A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, **2**, 12–26.

[55] Clarke, D.K., Salas, M. & Hassan, H. (1986). Euler calculations for multielement airfoils using Cartesian grids. *AIAA Journal*, **24**, 353–358.

[56] Coirier, W.J. (1994). *An adaptively-refined, Cartesian, cell-based scheme for the Euler and Navier-Stokes equations*. Ph.D. Thesis, University of Michigan.

[57] Coirier, W.J. & Powell, K.G. (1996). Solution-adaptive Cartesian cell approach for viscous and inviscid flows. *AIAA Journal*, **34**, 938–945.

[58] Colella, P., Graves, D.T., Keen, B.J. & Modiano, D. (2006). A Cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics*, **211**, 347–366.

[59] Courant, R., Friedrichs, K. & Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, **100**, 32–74.

[60] De Ruiter, M. & Van Keulen, F. (2004). Topology optimization using a topology description function. *Structural and Multidisciplinary Optimization*, **26**, 406–416.

[61] de Tullio, M., De Palma, P., Iaccarino, G., Pascazio, G. & Napolitano, M. (2007). An immersed boundary method for compressible flows using local grid refinement. *Journal of Computational Physics*, **225**, 2098–2117.

[62] Delannoy, Y. (1990). Two phase flow approach in unsteady cavitation modelling. In *Proceedings of Cavitation and Multiphase Flow Forum, ASME-FED*, vol. 98, 153–158, Toronto, Canada.

[63] Deng, Y., Liu, Z., Zhang, P., Liu, Y. & Wu, Y. (2011). Topology optimization of unsteady incompressible Navier–Stokes flows. *Journal of Computational Physics*, **230**, 6688–6708.

[64] Deshpande, M., Feng, J. & Merkle, C.L. (1994). Cavity flow predictions based on the Euler equations. *Journal of Fluids Engineering*, **116**, 36–44.

[65] Desjardins, O., McCaslin, J., Owkes, M. & Brady, P. (2013). Direct numerical and large-eddy simulation of primary atomization in complex geometries. *Atomization and Sprays*, **23**, 11.

[66] Dhert, T., Ashuri, T. & Martins, J.R. (2017). Aerodynamic shape optimization of wind turbine blades using a Reynolds-Averaged Navier–Stokes model and an adjoint method. *Wind Energy*, **20**, 909–926.

[67] Dick, E. & Steelant, J. (1997). Coupled solution of the steady compressible Navier-Stokes equations and the k-ε turbulence equations with a multigrid method. *Applied Numerical Mathematics*, **23**, 49–61.

[68] Dilgen, C.B., Dilgen, S.B., Fuhrman, D.R., Sigmund, O. & Lazarov, B.S. (2018). Topology optimization of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, **331**, 363–393.

[69] Duan, L., Wang, X. & Zhong, X. (2010). A high-order cut-cell method for numerical simulation of hypersonic boundary-layer instability with surface roughness. *Journal of Computational Physics*, **229**, 7207–7237.

[70] Duan, X., Li, F. & Qin, X. (2016). Topology optimization of incompressible Navier–Stokes problem by level set based adaptive mesh method. *Computers & Mathematics with Applications*, **72**, 1131–1141.

[71] Dwight, R.P. & Brezillon, J. (2006). Effect of approximations of the discrete adjoint on gradient-based optimization. *AIAA Journal*, **44**, 3022–3031.

[72] Eggleton, C.D. & Popel, A.S. (1998). Large deformation of red blood cell ghosts in a simple shear flow. *Physics of Fluids*, **10**, 1834–1845.

[73] Erney, R.W. (2009). *Verification and validation of single phase and cavitating flows using an open source CFD tool*. Master's thesis, The Pennsylvania State University.

[74] Fadlun, E., Verzicco, R., Orlandi, P. & Mohd-Yusof, J. (2000). Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, **161**, 35–60.

[75] Falcovitz, J., Alfandary, G. & Hanoch, G. (1997). A two–dimensional conservation laws scheme for compressible flows with moving boundaries. *Journal of Computational Physics*, **138**, 83–102.

[76] Ferziger, J.H., Perić, M. & Street, R.L. (2002). *Computational methods for fluid dynamics*. Springer, Berlin, Heidelberg. ISBN 9783319996912.

[77] Foley, J.D., Van, F.D., Van Dam, A., Feiner, S.K., Hughes, J.F. & Hughes, J. (1996). *Computer graphics: Principles and practice*, vol. 12110. Pearson Education Limited. ISBN 9780201121100.

[78] Foster, N.F. & Dulikravich, G.S. (1997). Three-dimensional aerodynamic shape optimization using genetic and gradient search algorithms. *Journal of Spacecraft and Rockets*, **34**, 36–42.

[79] Franc, J.P. & Michel, J.M. (2005). *Fundamentals of cavitation*. Springer, Dordrecht. ISBN 9781402022326.

[80] Gaffney JR, R. & Hassan, H. (1987). Euler calculations for wings using Cartesian grids. In *25th AIAA Aerospace Sciences Meeting*, 356, Hampton, Virginia, U.S.A.

[81] Geiss, M.J., Barrera, J.L., Boddeti, N. & Maute, K. (2019). A regularization scheme for explicit level–set XFEM topology optimization. *Frontiers of Mechanical Engineering*, **14**, 153–170.

[82] Gersborg-Hansen, A., Sigmund, O. & Haber, R.B. (2005). Topology optimization of channel flow problems. *Structural and Multidisciplinary Optimization*, **30**, 181–192.

[83] Gkaragkounis, K.T. (2020). *The continuous adjoint method in aerodynamic and conjugate heat transfer shape optimization, for turbulent flows*. Ph.D. Thesis, National Technical University of Athens.

[84] Glowinski, R., Pan, T.W. & Periaux, J. (1994). A fictitious domain method for Dirichlet problem and applications. *Computer Methods in Applied Mechanics and Engineering*, **111**, 283–303.

[85] Gokhale, N.B. (2019). *A dimensionally split Cartesian cut cell method for computational fluid dynamics*. Ph.D. Thesis, University of Cambridge.

[86] Gopalan, S. & Katz, J. (2000). Flow structure and modeling issues in the closure region of attached cavitation. *Physics of Fluids*, **12**, 895–911.

[87] Griewank, A. & Walther, A. (2008). *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. SIAM. ISBN 9780898716597.

[88] Griffith, B.E., Hornung, R.D., McQueen, D.M. & Peskin, C.S. (2007). An adaptive, formally second order accurate version of the immersed boundary method. *Journal of Computational Physics*, **223**, 10–49.

[89] Grogger, H. & Alajbegovic, A. (1998). Calculation of the cavitating flow in venturi geometries using two fluid model. *ASME Paper, FEDSM99-7364*.

[90] Harada, M., Tamaki, Y., Takahashi, Y. & Imamura, T. (2016). A novel simple cut-cell method for robust flow simulation on Cartesian grids. In *54th AIAA Aerospace Sciences Meeting*, 0601, San Diego, California, U.S.A.

[91] Hartmann, D., Meinke, M. & Schröder, W. (2008). An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods. *Computers & Fluids*, **37**, 1103–1125.

[92] Hartmann, D., Meinke, M. & Schröder, W. (2011). A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids. *Computer Methods in Applied Mechanics and Engineering*, **200**, 1038–1052.

[93] Hejranfar, K., Ezzatneshan, E. & Hesary, K. (2009). A dual-time implicit preconditioned Navier-Stokes method for solving 2D steady/unsteady laminar cavitating/noncavitating flows using a barotropic model. In *7th International Symposium on Cavitation*, Ann Arbor, Michigan, U.S.A.

[94] Hejranfar, K., Ezzatneshan, E. & Fattah-Hesari, K. (2015). A comparative study of two cavitation modeling strategies for simulation of inviscid cavitating flows. *Ocean Engineering*, **108**, 257–275.

[95] Helzel, C., Berger, M.J. & LeVeque, R.J. (2005). A high-resolution rotated grid method for conservation laws with embedded geometries. *SIAM Journal on Scientific Computing*, **26**, 785–809.

[96] Hereth, E.A. (2016). *Automatic parallel octree grid generation software with an extensible solver framework and a focus on urban simulation*. Ph.D. Thesis, University of Tennessee at Chattanooga.

[97] Hilbert, D. (1935). Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes*, 1–2, Springer.

[98] Hirt, C.W. & Nichols, B.D. (1981). Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, **39**, 201–225.

[99] Hohenberg, P.C. & Halperin, B.I. (1977). Theory of dynamic critical phenomena. *Reviews of Modern Physics*, **49**, 435.

[100] INRIA Sophia-Antipolis. TAPENADE. (TAPENADE Homepage). `https://www-sop.inria.fr/tropics/tapenade.html`.

[101] Jakobsson, S. & Amoignon, O. (2007). Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, **36**, 1119–1136.

[102] Jameson, A. (1988). Aerodynamic design via control theory. *Journal of Scientific Computing*, **3**.

[103] Jameson, A. (1995). Optimum aerodynamic design using CFD and control theory. In *12th Computational Fluid Dynamics Conference*, 1729, San Diego, California, U.S.A.

[104] Jameson, A. (2001). A perspective on computational algorithms for aerodynamic analysis and design. *Progress in Aerospace Sciences*, **37**, 197–243.

[105] Jameson, A. & Reuther, J. (1994). Control theory based airfoil design using the Euler equations. In *5th Symposium on Multidisciplinary Analysis and Optimization*, 4272, Panama City Beach, Florida, U.S.A.

[106] Jameson, A. & Vassberg, J.C. (2000). Studies of alternative numerical optimization methods applied to the brachistochrone problem. *Computational Fluid Dynamics Journal*, **9**, 281–296.

[107] Jenkins, N. & Maute, K. (2015). Level set topology optimization of stationary fluid–structure interaction problems. *Structural and Multidisciplinary Optimization*, **52**, 179–195.

[108] Jenkins, N. & Maute, K. (2016). An immersed boundary approach for shape and topology optimization of stationary fluid-structure interaction problems. *Structural and Multidisciplinary Optimization*, **54**, 1191–1208.

[109] Ji, H., Lien, F.S. & Yee, E. (2006). An efficient second-order accurate cut-cell method for solving the variable coefficient Poisson equation with jump conditions on irregular domains. *International Journal for Numerical Methods in Fluids*, **52**, 723–748.

[110] Kang, S., Iaccarino, G. & Moin, P. (2009). Accurate immersed-boundary reconstructions for viscous flow simulations. *AIAA Journal*, **47**, 1750–1760.

[111] Kapsoulis, D. (2019). *Low-cost metamodel-assisted evolutionary algorithms with application in shape optimization in fluid dynamics*. Ph.D. Thesis, National Technical University of Athens.

[112] Kapsoulis, D., Tsiakas, K., Trompoukis, X., Asouti, V. & Giannakoglou, K. (2018). Evolutionary multi-objective optimization assisted by metamodels, kernel PCA and multi-criteria decision making techniques with applications in aerodynamics. *Applied Soft Computing*, **64**, 1–13.

[113] Karakasis, M.K. & Giannakoglou, K.C. (2006). On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, **38**, 941–957.

[114] Karpouzas, G. & De Villiers, E. (2014). Level-set based topology optimization using the continuous adjoint method. In *OPT-I International Conference on Engineering and Applied Sciences Optimization*, Kos, Greece.

[115] Katz, A., Wissink, A.M., Sankaran, V., Meakin, R.L. & Chan, W.M. (2011). Application of strand meshes to complex aerodynamic flow fields. *Journal of Computational Physics*, **230**, 6512–6530.

[116] Kavvadias, I. (2016). *Continuous adjoint methods for steady and unsteady turbulent flows with emphasis on the accuracy of sensitivity derivatives*. Ph.D. Thesis, National Technical University of Athens.

[117] Kavvadias, I., Papoutsis-Kiachagias, E., Dimitrakopoulos, G. & Giannakoglou, K. (2015). The continuous adjoint approach to the k–ω SST turbulence model with applications in shape optimization. *Engineering Optimization*, **47**, 1523–1542.

[118] Kavvadias, I., Papoutsis-Kiachagias, E.M. & Giannakoglou, K.C. (2015). On the proper treatment of grid sensitivities in continuous adjoint methods for shape optimization. *Journal of Computational Physics*, **301**, 1–18.

[119] Kenway, G.K., Mader, C.A., He, P. & Martins, J.R. (2019). Effective adjoint approaches for computational fluid dynamics. *Progress in Aerospace Sciences*, **110**, 100542.

[120] Kim, C.S., Kim, C. & Rho, O.H. (2002). Effects of constant eddy viscosity assumption on gradient-based design optimization. *AIAA 2002-262. 40th AIAA Aerospace Sciences Meeting & Exhibit*.

[121] Kim, C.S., Kim, C. & Rho, O.H. (2003). Feasibility study of constant eddy-viscosity assumption in gradient-based design optimization. *Journal of Aircraft*, **40**, 1168–1176.

[122] Kim, J., Kim, D. & Choi, H. (2001). An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, **171**, 132–150.

[123] Kinzel, M.P. (2008). *Computational techniques and analysis of cavitating-fluid flows*. Ph.D. Thesis, The Pennsylvania State University.

[124] Kirkpatrick, M., Armfield, S. & Kent, J. (2003). A representation of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid. *Journal of Computational Physics*, **184**, 1–36.

[125] Knapp, R.T., Daily, J.W. & Hammit, F.G. (1970). *Cavitation*. McGraw-Hill. ISBN 9780070350809.

[126] Knopp, T., Alrutz, T. & Schwamborn, D. (2006). A grid and flow adaptive wall–function method for RANS turbulence modelling. *Journal of Computational Physics*, **220**, 19–40.

[127] Koch, J., Papoutsis-Kiachagias, E. & Giannakoglou, K. (2017). Transition from adjoint level set topology to shape optimization for 2D fluid mechanics. *Computers & Fluids*, **150**, 123–138.

[128] Kontoleontos, E., Papoutsis-Kiachagias, E., Zymaris, A., Papadimitriou, D. & Giannakoglou, K. (2013). Adjoint–based constrained topology optimization for viscous flows, including heat transfer. *Engineering Optimization*, **45**, 941–961.

[129] Koop, A. (2008). *Numerical simulation of unsteady three-dimensional sheet cavitation*. Ph.D. Thesis, University of Twente.

[130] Koukouvinis, F. & Gavaises, M., eds. (2021). *Cavitation and Bubble Dynamics: Fundamentals and Applications*. Academic Press. ISBN 9780128233887.

[131] Kreissl, S. & Maute, K. (2012). Levelset based fluid topology optimization using the extended finite element method. *Structural and Multidisciplinary Optimization*, **46**, 311–326.

[132] Kreissl, S., Pingen, G. & Maute, K. (2011). An explicit level set approach for generalized shape optimization of fluids with the Lattice Boltzmann method. *International Journal for Numerical Methods in Fluids*, **65**, 496–519.

[133] Kreissl, S., Pingen, G. & Maute, K. (2011). Topology optimization for unsteady flow. *International Journal for Numerical Methods in Engineering*, **87**, 1229–1253.

[134] Kröger, J., Kühl, N. & Rung, T. (2018). Adjoint volume-of-fluid approaches for the hydrodynamic optimisation of ships. *Ship Technology Research*, **65**, 47–68.

[135] Kubota, A., Kato, H., Yamaguchi, H. *et al.* (1992). A new modelling of cavitating flows: A numerical study of unsteady cavitation on a hydrofoil section. *Journal of Fluid Mechanics*, **240**, 59–96.

[136] Kühl, N. (2021). *Adjoint-based shape optimization constraint by turbulent two-phase Navier-Stokes systems*. Ph.D. Thesis, Technische Universität Hamburg.

[137] Kunz, R.F., Boger, D.A., Chyczewski, T.S., Stinebring, D., Gibeling, H. & Govindan, T. (1999). Multi–phase CFD analysis of natural and ventilated cavitation about submerged bodies. In *Proceedings of the 3rd ASME-JSME Joint Fluids Engineering Conference*, San Francisco, California, U.S.A.

[138] Kunz, R.F., Boger, D.A., Stinebring, D.R., Chyczewski, T.S., Lindau, J.W., Gibeling, H.J., Venkateswaran, S. & Govindan, T. (2000). A preconditioned Navier–Stokes method for two–phase flows with application to cavitation prediction. *Computers & Fluids*, **29**, 849–875.

[139] Kyriakou, S. (2013). *Evolutionary algorithm-based design-optimization methods in turbomachinery*. Ph.D. Thesis, National Technical University of Athens.

[140] Lai, M.C. & Peskin, C.S. (2000). An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics*, **160**, 705–719.

[141] Langer, S. (2013). Application of a line implicit method to fully coupled system of equations for turbulent flow problems. *International Journal of Computational Fluid Dynamics*, **27**, 131–150.

[142] Langer, S. & Suarez, G. (2022). Loosely coupled and coupled solution methods for the RANS equations and a one-equation turbulence model. *Computers & Fluids*, **232**, 105186.

[143] Launder, B. & Spalding, D. (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, **3**, 269–289.

[144] Lazarov, B.S. & Sigmund, O. (2011). Filters in topology optimization based on Helmholtz-type differential equations. *International Journal for Numerical Methods in Engineering*, **86**, 765–781.

[145] Lindau, J.W., Pena, C., Baker, W.J., Dreyer, J.J., Moody, W.L., Kunz, R.F. & Paterson, E.G. (2012). Modeling of cavitating flow through waterjet propulsors. *International Journal of Rotating Machinery*, **2012**.

[146] Lions, J. (1971). *Optimal Control of Systems Governed by Partial Differential Equations*. Springer, Berlin, Heidelberg. ISBN 9783540051152.

[147] Liu, F. & Zheng, X. (1996). A strongly coupled time-marching method for solving the Navier–Stokes and k-ω turbulence model equations with multigrid. *Journal of Computational Physics*, **128**, 289–300.

[148] Liu, G., Geier, M., Liu, Z., Krafczyk, M. & Chen, T. (2014). Discrete adjoint sensitivity analysis for fluid flow topology optimization based on the generalized Lattice Boltzmann method. *Computers & Mathematics with Applications*, **68**, 1374–1392.

[149] Löhner, R. (1989). Adaptive remeshing for transient problems. *Computer Methods in Applied Mechanics and Engineering*, **75**, 195–214.

[150] Lorensen, W.E. & Cline, H.E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, **21**, 163–169.

[151] Lundgaard, C., Alexandersen, J., Zhou, M., Andreasen, C.S. & Sigmund, O. (2018). Revisiting density-based topology optimization for fluid-structure-interaction problems. *Structural and Multidisciplinary Optimization*, **58**, 969–995.

[152] Malan, A., Lewis, R. & Nithiarasu, P. (2002). An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part i. Theory and implementation. *International Journal for Numerical Methods in Engineering*, **54**, 695–714.

[153] Manninen, M., Taivassalo, V. & Kallio, S. (1996). On the mixture model for multiphase flow. *VTT PUBLICATIONS 288*.

[154] Marta, A.C. & Shankaran, S. (2013). On the handling of turbulence equations in RANS adjoint solvers. *Computers & Fluids*, **74**, 102–113.

[155] Mavriplis, D. (2003). Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. In *16th AIAA Computational Fluid Dynamics Conference*, 3986, Orlando, Florida, U.S.A.

[156] May, S. & Berger, M.J. (2017). An explicit implicit scheme for cut cells in embedded boundary meshes. *Journal of Scientific Computing*, **71**, 919–943.

[157] Meinke, M., Schneiders, L., Günther, C. & Schröder, W. (2013). A cut-cell method for sharp moving boundaries in Cartesian grids. *Computers & Fluids*, **85**, 135–142.

[158] Melton, J.E., Enomoto, F. & Berger, M.J. (1993). 3D automatic Cartesian grid generation for Euler flows. In *11th Computational Fluid Dynamics Conference*, 3386, Orlando, Florida, U.S.A.

[159] Merkle, C.L. (1998). Computational modelling of the dynamics of sheet cavitation. In *Proceedings of 3rd International Symposium on Cavitation*.

[160] Merlin, C., Domingo, P. & Vervisch, L. (2013). Immersed boundaries in large eddy simulation of compressible flows. *Flow, turbulence and combustion*, **90**, 29–68.

[161] Meyer, M., Devesa, A., Hickel, S., Hu, X. & Adams, N.A. (2010). A conservative immersed interface method for large-eddy simulation of incompressible flows. *Journal of Computational Physics*, **229**, 6300–6317.

[162] Meyer, M., Hickel, S. & Adams, N. (2010). Assessment of implicit large-eddy simulation with a conservative immersed interface method for turbulent cylinder flow. *International Journal of Heat and Fluid Flow*, **31**, 368–377.

[163] Mithun, M.G., Koukouvinis, P., Karathanassis, I.K. & Gavaises, M. (2019). Numerical simulation of three-phase flow in an external gear pump using immersed boundary approach. *Applied Mathematical Modelling*, **72**, 682–699.

[164] Mittal, R. & Iaccarino, G. (2005). Immersed boundary methods. *Annual Review of Fluid Mechanics*, **37**, 239–261.

[165] Mittal, R., Dong, H., Bozkurttas, M., Najjar, F., Vargas, A. & Von Loebbecke, A. (2008). A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of Computational Physics*, **227**, 4825–4852.

[166] Mohd-Yusof, J. (1997). Combined immersed-boundary/b-spline methods for simulations of flow in complex geometries. *Center for Turbulence Research Annual Research Briefs*, **161**, 317–327.

[167] Morgut, M. & Nobile, E. (2012). Numerical predictions of cavitating flow around model scale propellers by CFD and advanced model calibration. *International Journal of Rotating Machinery*, 1–11.

[168] Muralidharan, B. & Menon, S. (2016). A high-order adaptive Cartesian cut-cell method for simulation of compressible viscous flow over immersed bodies. *Journal of Computational Physics*, **321**, 342–368.

[169] Murman, S., Aftosmis, M. & Berger, M. (2003). Implicit approaches for moving boundaries in a 3-d Cartesian method. In *41st Aerospace Sciences Meeting and Exhibit*, 1119, Reno, Nevada, U.S.A.

[170] Nadarajah, S. & Jameson, A. (2000). A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace Sciences Meeting and Exhibit*, 667, Reno, Nevada, U.S.A.

[171] Nam, J. & Lien, F. (2015). Assessment of ghost-cell based cut-cell method for large-eddy simulations of compressible flows at high Reynolds number. *International Journal of Heat and Fluid Flow*, **53**, 1–14.

[172] Nemec, M. & Aftosmis, M. (2005). Adjoint algorithm for CAD–based shape optimization using a Cartesian method. In *17th AIAA Computational Fluid Dynamics Conference*, 4987, Toronto, Ontario, Canada.

[173] Nemec, M. & Aftosmis, M. (2006). Aerodynamic shape optimization using a Cartesian adjoint method and CAD geometry. In *24th AIAA Applied Aerodynamics Conference*, 3456, San Francisco, California, U.S.A.

[174] Nemec, M. & Aftosmis, M.J. (2008). Adjoint sensitivity computations for an embedded-boundary Cartesian mesh method. *Journal of Computational Physics*, **227**, 2724–2742.

[175] Nemec, M., Aftosmis, M., Murman, S. & Pulliam, T. (2005). Adjoint formulation for an embedded-boundary Cartesian method. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 877, Reno, Nevada, U.S.A.

[176] Nguyen, T., Isakari, H., Takahashi, T., Yaji, K., Yoshino, M. & Matsumoto, T. (2020). Level-set based topology optimization of transient flow using Lattice Boltzmann method considering an oscillating flow condition. *Computers & Mathematics with Applications*, **80**, 82–108.

[177] Nielsen, E.J. & Anderson, W.K. (1999). Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. *AIAA Journal*, **37**, 1411–1419.

[178] Nocedal, J. & Wright, S. (2006). *Numerical optimization*. Springer, New York, NY. ISBN 9780387303031.

[179] Olesen, L.H., Okkels, F. & Bruus, H. (2006). A high-level programming-language implementation of topology optimization applied to steady-state Navier–Stokes flow. *International Journal for Numerical Methods in Engineering*, **65**, 975–1001.

[180] Örley, F., Pasquariello, V., Hickel, S. & Adams, N.A. (2015). Cut-element based immersed boundary method for moving geometries in compressible liquid flows with cavitation. *Journal of Computational Physics*, **283**, 1–22.

[181] Othmer, C. (2008). A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International Journal for Numerical Methods in Fluids*, **58**, 861–877.

[182] Othmer, C. (2014). Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, **4**, 1–23.

[183] Palacios, F., Alonso, J. & Jameson, A. (2012). Shape sensitivity of free-surface interfaces using a level set methodology. In *42nd AIAA Fluid Dynamics Conference and Exhibit*, 3341, New Orleans, Louisiana, U.S.A.

[184] Palacios, F., Alonso, J.J. & Jameson, A. (2013). Design of free-surface interfaces using RANS equations. In *43rd AIAA Fluid Dynamics Conference*, 3210, San Diego, California, U.S.A.

[185] Pan, D. & Shen, T.T. (2009). Computation of incompressible flows with immersed bodies by a simple ghost cell method. *International Journal for Numerical Methods in Fluids*, **60**, 1378–1401.

[186] Papadimitriou, D. (2007). *Adjoint formulations for the analysis and design of turbomachinery cascades and optimal grid adaptation using aposteriori error analysis*. Ph.D. Thesis, National Technical University of Athens.

[187] Papadimitriou, D. & Giannakoglou, K. (2007). A continuous adjoint method with objective function derivatives based on boundary integrals, for inviscid and viscous flows. *Computers & Fluids*, **36**, 325–341.

[188] Papoutsis-Kiachagias, E., Kyriacou, S. & Giannakoglou, K. (2014). The continuous adjoint method for the design of hydraulic turbomachines. *Computer Methods in Applied Mechanics and Engineering*, **278**, 621–639.

[189] Papoutsis-Kiachagias, E., Zymaris, A., Kavvadias, I., Papadimitriou, D. & Giannakoglou, K. (2015). The continuous adjoint approach to the k–ε turbulence model for shape optimization and optimal active control of turbulent flows. *Engineering Optimization*, **47**, 370–389.

[190] Papoutsis-Kiachagias, E., Asouti, V., Giannakoglou, K., Gkagkas, K., Shimokawa, S. & Itakura, E. (2019). Multi-point aerodynamic shape optimization of cars based on continuous adjoint. *Structural and Multidisciplinary Optimization*, **59**, 675–694.

[191] Papoutsis-Kiachagias, E.M. (2013). *Adjoint methods for turbulent flows, applied to shape or topology optimization and robust design*. Ph.D. Thesis, National Technical University of Athens.

[192] Papoutsis-Kiachagias, E.M. & Giannakoglou, K.C. (2016). Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Archives of Computational Methods in Engineering*, **23**, 255–299.

[193] Patel, V.C., Rodi, W. & Scheuerer, G. (1985). Turbulence models for near–wall and low Reynolds number flows–A review. *AIAA Journal*, **23**, 1308–1319.

[194] Pember, R.B., Bell, J.B., Colella, P., Curtchfield, W.Y. & Welcome, M.L. (1995). An adaptive Cartesian grid method for unsteady compressible flow in irregular regions. *Journal of Computational Physics*, **120**, 278–304.

[195] Pendar, M.R. & Roohi, E. (2018). Cavitation characteristics around a sphere: An LES investigation. *International Journal of Multiphase Flow*, **98**, 1–23.

[196] Peskin, C.S. (1972). Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, **10**, 252–271.

[197] Peskin, C.S. (2002). The immersed boundary method. *Acta Numerica*, **11**, 479–517.

[198] Peter, J.E. & Dwight, R.P. (2010). Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, **39**, 373–391.

[199] Piegl, L. & Tiller, W. (1997). *The NURBS book*. Springer, Berlin, Heidelberg. ISBN 9783540615453.

[200] Pingen, G., Evgrafov, A. & Maute, K. (2007). Topology optimization of flow domains using the Lattice Boltzmann method. *Structural and Multidisciplinary Optimization*, **34**, 507–524.

[201] Pingen, G., Waidmann, M., Evgrafov, A. & Maute, K. (2010). A parametric level-set approach for topology optimization of flow domains. *Structural and Multidisciplinary Optimization*, **41**, 117–131.

[202] Piomelli, U. & Balaras, E. (2002). Wall-layer models for large-eddy simulations. *Annual Review of Fluid Mechanics*, **34**, 349–374.

[203] Pironneau, O. (1982). *Optimal Shape Design for Elliptic Systems*. Springer, Berlin, Heidelberg. ISBN 9783642877247.

[204] Plesset, M.S. (1949). The dynamics of cavitation bubbles. *Journal of Applied Mechanics*, **16**, 277–282.

[205] Quirk, J.J. (1994). An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies. *Computers & Fluids*, **23**, 125–142.

[206] Ragab, S. (2001). Shape optimization in free surface potential flow using an adjoint formulation-surface ships. In *15th AIAA Computational Fluid Dynamics Conference*, 3042, Anaheim, California, U.S.A.

[207] Reuther, J. & Jameson, A. (1995). Aerodynamic shape optimization of wing and wing-body configurations using control theory. In *33rd Aerospace Sciences Meeting and Exhibit*, 123, Reno, Nevada, U.S.A.

[208] Reuther, J., Jameson, A., Farmer, J., Martinelli, L. & Saunders, D. (1996). Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. In *34th Aerospace Sciences Meeting and Exhibit*, 94, Reno, Nevada, U.S.A.

[209] Roe, P.L. (1981). Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, **43**, 357–372.

[210] Roman, F., Armenio, V. & Fröhlich, J. (2009). A simple wall-layer model for large eddy simulation with immersed boundary method. *Physics of Fluids*, **21**, 101701.

[211] Rouse, H. & McNown, J.S. (1948). *Cavitation and pressure distribution: head forms at zero angle of yaw*. State University of Iowa Studies in Engineering, State University of Iowa, Iowa City.

[212] Ruffin, S.M. & Lee, J. (2009). Adaptation of a k-epsilon model to a Cartesian grid based methodology. *International Journal of Mathematical Models and Methods in Applied Sciences*, **3**, 238–245.

[213] Sá, L., Amigo, R., Novotny, A. & Silva, E. (2016). Topological derivatives applied to fluid flow channel design optimization problems. *Structural and Multidisciplinary Optimization*, **54**, 249–264.

[214] Samouchos, K. (2022). *The cut-cell method for the prediction of 2D/3D flows in complex geometries and the adjoint-based shape optimization*. Ph.D. Thesis, National Technical University of Athens.

[215] Sauer, J. & Schnerr, G.H. (2000). Unsteady cavitating flow-a new cavitation model based on a modified front capturing method and bubble dynamics. In *Proceedings of 2000 ASME Fluid Engineering Summer Conference*, vol. 251, 1073–1079, Boston, Massachusetts, U.S.A.

[216] Saurel, R., Cocchi, J.P. & Butler, P.B. (1999). Numerical study of cavitation in the wake of a hypervelocity underwater projectile. *Journal of Propulsion and Power*, **15**, 513–522.

[217] Schneiders, L., Hartmann, D., Meinke, M. & Schröder, W. (2013). An accurate moving boundary formulation in cut-cell methods. *Journal of Computational Physics*, **235**, 786–809.

[218] Schneiders, L., Günther, C., Meinke, M. & Schröder, W. (2016). An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows. *Journal of Computational Physics*, **311**, 62–86.

[219] Semenenko, V.N. & Ukrainian Academy of Sciences Kiev Institute of Hydromechanics (2001). Artificial supercavitation. Physics and calculation. Tech. Rep. ADP012080, Defense Technical Information Center.

[220] Senocak, I. (2002). *A computational methodology for the simulation of turbulent cavitating flows*. Ph.D. Thesis, University of Florida.

[221] Senocak, I. & Shyy, W. (2001). Numerical simulation of turbulent flows with sheet cavitation. In *CAV 2001: Fourth International Symposium on Cavitation*, Pasadena, California, U.S.A.

[222] Senocak, I. & Shyy, W. (2002). A pressure-based method for turbulent cavitating flow computations. *Journal of Computational Physics*, **176**, 363–383.

[223] Senocak, I. & Shyy, W. (2004). Interfacial dynamics-based modelling of turbulent cavitating flows, part-1: Model development and steady-state computations. *International Journal for Numerical Methods in Fluids*, **44**, 975–995.

[224] Sethian, J.A. (1999). Fast marching methods. *SIAM Reviews*, **41**, 199–235.

[225] Shankaran, S. & Jameson, A. (2003). Numerical analysis and design of upwind sails. In *21st AIAA Applied Aerodynamics Conference*, 3501, Orlando, Florida, U.S.A.

[226] Sharma, A., Villanueva, H. & Maute, K. (2017). On shape sensitivities with heaviside-enriched XFEM. *Structural and Multidisciplinary Optimization*, **55**, 385–408.

[227] Sharp, H.T. & Sirovich, L. (1989). Constructing a continuous parameter range of computational flows. *AIAA Journal*, **27**, 1326–1331.

[228] Shen, Y. & Dimotakis, P.E. (1989). The influence of surface cavitation on hydrodynamic forces. In *22nd American Towing Tank Conference (ATTC'89)*, 45–53, St. John's, Newfoundland, Canada.

[229] Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, 517–524, New York, USA.

[230] Shih, T.H., Liou, W.W., Shabbir, A., Yang, Z. & Zhu, J. (1995). A new k-ε eddy viscosity model for high Reynolds number turbulent flows. *Computers & Fluids*, **24**, 227–238.

[231] Shinn, A., Goodwin, M. & Vanka, S. (2009). Immersed boundary computations of shear-and buoyancy-driven flows in complex enclosures. *International Journal of Heat and Mass Transfer*, **52**, 4082–4089.

[232] Sigmund, O. (2007). Morphology–based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, **33**, 401–424.

[233] Sigmund, O. & Maute, K. (2013). Topology optimization approaches. *Structural and Multidisciplinary Optimization*, **48**, 1031–1055.

[234] Singhal, A.K., Athavale, M.M., Li, H. & Jiang, Y. (2002). Mathematical basis and validation of the full cavitation model. *Journal of Fluids Engineering*, **124**, 617–624.

[235] Sinha, K. & Candler, G. (1998). Convergence improvement of two-equation turbulence model calculations. In *29th AIAA Computational Fluid Dynamics Conference*, 2649, Albuquerque, New Mexico, U.S.A.

[236] Slotnick, J.P., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E. & Mavriplis, D.J. (2014). CFD vision 2030 study: A path to revolutionary computational aerosciences. Tech. Rep. CR-2014-218178, NASA.

[237] Sondak, D.L. (1992). *Wall functions for the k–[epsilon] turbulence model in generalized nonorthogonal curvilinear coordinates*. Ph.D. Thesis, Iowa State University.

[238] Sotiropoulos, F. & Yang, X. (2014). Immersed boundary methods for simulating fluid–structure interaction. *Progress in Aerospace Sciences*, **65**, 1–21.

[239] Spalding, D.B. (1961). A single formula for the "law of the wall". *Journal of Applied Mechanics*, **28**, 455.

[240] Spall, J.C. (2003). *Introduction to stochastic search and optimization: Estimation, simulation, and control*, vol. 65. John Wiley & Sons. ISBN 9780471330523.

[241] Stride, E. & Coussios, C. (2019). Nucleation, mapping and control of cavitation for drug delivery. *Nature Reviews Physics*, **1**, 495–509.

[242] Stück, A. (2012). *Adjoint Navier-Stokes methods for hydrodynamic shape optimisation*. Ph.D. Thesis, Technische Universität Hamburg.

[243] Stutz, B. & Reboud, J.L. (1997). Two-phase flow structure of sheet cavitation. *Physics of Fluids*, **9**, 3678–3686.

[244] Sutherland, I.E. & Hodgman, G.W. (1974). Reentrant polygon clipping. *Communications of the ACM*, **17**, 32–42.

[245] Svanberg, K. (1987). The method of moving asymptotes—A new method for structural optimization. *International Journal for Numerical Methods in Engineering*, **24**, 359–373.

[246] Svanberg, K. (2002). A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, **12**, 555–573.

[247] Svanberg, K. (2007). MMA and GCMMA-two methods for nonlinear optimization. Tech. Rep., Optimization and Systems Theory, KTH, Stockholm, Sweden.

[248] Sweby, P.K. (1984). High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, **21**, 995–1011.

[249] Takahashi, Y. & Imamura, T. (2014). High Reynolds number steady state flow simulation using immersed boundary method. In *52nd Aerospace Sciences Meeting*, National Harbor, Maryland, U.S.A.

[250] Tamaki, Y., Harada, M. & Imamura, T. (2017). Near-wall modification of Spalart–Allmaras turbulence model for immersed boundary method. *AIAA Journal*, **55**, 3027–3039.

[251] Tamamidis, P., Zhang, G. & Assanis, D.N. (1996). Comparison of pressure-based and artificial compressibility methods for solving 3D steady incompressible viscous flows. *Journal of Computational Physics*, **124**, 1–13.

[252] Tessicini, F., Iaccarino, G., Fatica, M., Wang, M. & Verzicco, R. (2002). Wall modeling for large-eddy simulation using an immersed boundary method. *Center for Turbulence Research Annual Research Briefs*, **2002**, 181–187.

[253] Toro, E.F. (2010). *Riemann solvers and numerical methods for fluid dynamics: A practical introduction*. Springer, Berlin, Heidelberg. ISBN 9783642064388.

[254] Tseng, Y.H. & Ferziger, J.H. (2003). A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics*, **192**, 593–623.

[255] Tsiakas, K.T. (2019). *Development of shape parameterization techniques, a flow solver and its adjoint, for optimization on GPUs. Turbomachinery and external aerodynamics applications*. Ph.D. Thesis, National Technical University of Athens.

[256] Turbulence Modeling Resource (Langley Research Center). 2DZP: 2D zero pressure gradient flat plate validation case. `https://turbmodels.larc.nasa.gov/flatplate_val.html`.

[257] Turkel, E. (1987). Preconditioned methods for solving the incompressible and low speed compressible equations. *Journal of Computational Physics*, **72**, 277–298.

[258] Uhlmann, M. (2005). An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, **209**, 448–476.

[259] Van Dijk, N.P., Maute, K., Langelaar, M. & Van Keulen, F. (2013). Level-set methods for structural topology optimization: A review. *Structural and Multidisciplinary Optimization*, **48**, 437–472.

[260] Van Leer, B. (1979). Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, **32**, 101–136.

[261] Vandromme, D. (1993). Turbulence modeling for compressible flows and implementation in Navier-Stokes solvers. Tech. Rep., VKI An Introduction to Modeling Turbulence.

[262] Vankateswaran, S., Deshpande, M. & Merkle, C. (1995). The application of preconditioning to reacting flow computations. In *12th Computational Fluid Dynamics Conference*, 1673, San Diego, California, U.S.A.

[263] Venkatakrishnan, V. (1995). Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, **118**, 120–130.

[264] Villanueva, C.H. & Maute, K. (2017). CutFEM topology optimization of 3D laminar incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, **320**, 444–473.

[265] Vrionis, Y.P., Samouchos, K.D. & Giannakoglou, K.C. (2019). Implementation of a conservative cut–cell method for the simulation of two–phase cavitating flows. In *10th International Conference on Computational Methods (ICCM2019)*, 440–452, Singapore.

[266] Vrionis, Y.P., Samouchos, K.D. & Giannakoglou, K.C. (2021). The continuous adjoint cut-cell method for shape optimization in cavitating flows. *Computers & Fluids*, **224**, 104974.

[267] Vrionis, Y.P., Samouchos, K.D. & Giannakoglou, K.C. (2021). Topology optimization in fluid mechanics using continuous adjoint and the cut-cell method. *Computers & Mathematics with Applications*, **97**, 286–297.

[268] Vu-Huu, T., Phung-Van, P., Nguyen-Xuan, H. & Wahab, M.A. (2018). A polytree-based adaptive polygonal finite element method for topology optimization of fluid-submerged breakwater interaction. *Computers & Mathematics with Applications*, **76**, 1198–1218.

[269] WALL-MODELED LARGE EDDY SIMULATION RESOURCE (WMLES Homepage). `https://wmles.umd.edu/`.

[270] Walther, A. (2009). Getting Started with ADOL-C. *Combinatorial Scientific Computing*.

[271] Wang, F., Lazarov, B.S. & Sigmund, O. (2011). On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, **43**, 767–784.

[272] Wang, M. & Moin, P. (2002). Dynamic wall modeling for large–eddy simulation of complex turbulent flows. *Physics of Fluids*, **14**, 2043–2051.

[273] Wang, M.Y. & Wang, S. (2006). Parametric shape and topology optimization with radial basis functions. In *IUTAM symposium on topological design optimization of structures, machines and materials*, 13–22, Springer, Dordrecht.

[274] Wang, Z. (1998). A Quadtree–based adaptive Cartesian/Quad grid flow solver for Navier–Stokes equations. *Computers & Fluids*, **27**, 529–549.

[275] Wang, Z. & Chen, R. (2002). Anisotropic solution–adaptive viscous Cartesian grid method for turbulent flow simulation. *AIAA Journal*, **40**, 1969–1978.

[276] Weiss, J.M., Maruszewski, J.P. & Smith, W.A. (1999). Implicit solution of preconditioned Navier-Stokes equations using algebraic multigrid. *AIAA Journal*, **37**, 29–36.

[277] Wieghardt, K. & Tillmann, W. (1951). On the turbulent friction layer for rising pressure. Tech. Rep. TM-1314, NACA.

[278] Yakhot, V., Orszag, S., Thangam, S., Gatski, T. & Speziale, C. (1992). Development of turbulence models for shear flows by a double expansion technique. *Physics of Fluids A: Fluid Dynamics*, **4**, 1510–1520.

[279] Ye, T., Mittal, R., Udaykumar, H. & Shyy, W. (1999). An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, **156**, 209–240.

[280] Zhou, H., Xiang, M., Zhao, S. & Zhang, W. (2019). Development of a multiphase cavitation solver and its application for ventilated cavitating flows with natural cavitation. *International Journal of Multiphase Flow*, **115**, 62–74.

[281] Zwart, P., Gerber, A. & Belamri, T. (2004). A two-phase flow model for predicting cavitation dynamics. In *5th International Conference on Multiphase Flow*, vol. 152, Yokohama, Japan.

[282] Zymaris, A., Papadimitriou, D., Giannakoglou, K. & Othmer, C. (2009). Continuous adjoint approach to the Spalart–Allmaras turbulence model for incompressible flows. *Computers & Fluids*, **38**, 1528–1538.

[283] Zymaris, A., Papadimitriou, D., Giannakoglou, K.C. & Othmer, C. (2010). Adjoint wall functions: A new concept for use in aerodynamic shape optimization. *Journal of Computational Physics*, **229**, 5228–5245.

[284] Zymaris, A.S. (2010). *Adjoint methods for the design of shapes with optimal aerodynamic performance in laminar and turbulent flows*. Ph.D. Thesis, National Technical University of Athens.