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

Development of Shape Parameterization Techniques, a Flow Solver and its Adjoint, for Optimization on GPUs. Turbomachinery and External Aerodynamics Applications

PhD Thesis

Konstantinos T. Tsiakas

Supervisor: Kyriakos C. Giannakoglou
Professor NTUA

Athens, 2019

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

Development of Shape Parameterization Techniques, a Flow Solver and its Adjoint, for Optimization on GPUs. Turbomachinery and External Aerodynamics Applications

PhD Thesis

Konstantinos T. Tsiakas

Athens, 2019

Abstract

This thesis is concerned with the development of methods and numerical tools for fast and cost-efficient aerodynamic/hydrodynamic shape optimization. Developments related to the ensemble of building blocks of a shape optimization framework, namely **a)** the Computational Fluid Dynamics (CFD) analysis software, **b)** the optimization methods and **c)** the shape parameterization techniques are performed. These reduce the overall optimization cost and make them appropriate for use in industrial design processes. Both Gradient-Based (GB) optimization methods assisted by the adjoint technique and Evolutionary Algorithms (EAs) are considered, with emphasis on the development of the former. Most of the proposed methods focus on turbomachinery design/optimization but, their range of applicability is much wider, including also external aerodynamics applications.

The GPU-enabled in-house (U)RANS solver PUMA (Parallel Unstructured Multi-row Adjoint), previously developed only for steady and unsteady compressible flows, is enriched with an incompressible flow solver based on the Artificial Compressibility (AC) approach. The RANS equations for both compressible and incompressible flows are expressed in a relative frame of reference although they are solved for the absolute velocity components, using the Multiple Reference Frame (MRF) approach. To enable the simulation of multi-row turbomachinery configurations, Rotor-Stator Interaction techniques, such as the mixing interface, frozen rotor and sliding interface approaches are employed. GPU-specific programming techniques and numerical schemes, developed in previous works on the PUMA software, are revisited, upgraded and re-evaluated, resulting to a GPU implementation which is up to 40$\times$ faster than the equivalent CPU one. This is achieved by careful code restructuring, delicate GPU memory handling, the use of Mixed-Precision Arithmetics (MPA) and efficient utilization of the NVIDIA CUDA programming environment, in order to optimally exploit the SIMD architecture and hardware characteristics of modern GPUs. Especially, MPA is a technique according to which, when solving for the update of flow quantities, double precision storage is used for the right-hand-side (r.h.s.) terms, while single precision storage is employed for the left-hand-side (l.h.s.) ones, reducing the overall memory usage and increasing GPU memory bandwidth. The PUMA solver is also capable of running in parallel on multiple GPUs, using overlapping domains. Communications among GPUs of the same computational node are carried out through the shared on-node CPU memory, while the MPI protocol is employed for communications among different computational nodes.

A series of test cases are studied for validation and verification purposes of the PUMA compressible and incompressible flow solver. These concern the turbulent flow around two isolated airfoils, the flow through a convergent-divergent transonic diffuser and over a backward-facing step and the flow around a Horizontal Axis Wind Turbine (HAWT) blade, with published results to compare with. Furthermore, the solver is also validated/verified on the prediction of the flow field around the ONERA M6 wing, inside a propeller type water turbine and a high-pressure turbine stator.

Throughout this thesis, the continuous adjoint method is exclusively used and its development for both the compressible and the AC-based incompressible flow solver is proposed. The method allows the computation of the sensitivity derivatives of several objective functions (like the lift and drag forces, the torque, the total pressure losses and the turbomachinery row efficiency) with respect to the aerodynamic shape at a cost independent of the number of design variables controlling the shape, enabling the efficient use of Gradient-Based (GB) optimization methods.

Two different expressions for the sensitivity derivatives are developed, namely the Surface Integral (SI) and Field Integral (FI) adjoint formulations. The former makes use of the Leibniz rule and, usually neglecting surface integral terms containing the residuals of the state equations (referred to as Severed-SI,S-SI), results in a series of surface integrals that constitute the sensitivity derivatives expression. The latter, on the other hand, results to sensitivity derivatives expressed as the sum of surface and volume integrals. Though mathematically equivalent, numerically these expressions result in sensitivity derivatives of potentially different accuracy and computational complexity. The FI approach is more accurate, especially in turbulent flows, compared to the S-SI one but requires a properly differentiated grid displacement model and more arithmetic operations in order to compute the volume integrals involved in the sensitivity derivatives expressions. Conversely, the SI approach results in potentially less accurate but, also, computationally cheaper sensitivity derivatives.

In both formulations, the variations of the eddy viscosity due to shape changes are taken into account by differentiating the Spalart-Allmaras turbulence model for both compressible and incompressible flows. The variations of the distance from the nearest wall are also taken into account, through the differentiation of the Eikonal equation, i.e. a PDE computing distances from the walls. Proper consideration of these variations results in a complete and consistent expression for the sensitivity derivatives enhancing the convergence and robustness of the GB optimization method.

In the field of shape parameterization for aerodynamic optimization, a parametric modeler for turbomachinery blade rows is developed, namely the GMTurbo software. A bottom-up strategy is followed, where the meridional outline is constructed, camber lines of the blade at several spanwise positions are built using metal and other angles, thickness is added normal to them and the blade sections are interpolated to yield the 3D blade shape. With the GMTurbo, a wide range of blade shapes can be generated parametrically, while maintaining a compact CAD-compatible representation of the geometry. The GMTurbo software, together with the PUMA flow solver and the in-house evolutionary algorithm based optimization tool (EASY software), is employed for the optimization of a propeller type water turbine. Both the inlet guide vanes and the runner shape are optimized for maximum efficiency. In this case, the interaction between the stationary and rotating blades is taken into account through the mixing plane technique. The advantage of jointly optimizing the two rows is demonstrated.

Additionally, a Free Form Deformation technique, based on Volumetric NURBS, is developed to support optimization loops. According to this technique a NURBS volume is defined by means of a control point lattice, in which the aerodynamic shape under consideration is embedded together with parts of the CFD domain. Displacements of the control points lead to displacements of any entity embedded in the NURBS volume allowing for a monolithic approach for simultaneously displacing the aerodynamic body and the CFD mesh around it. A new strategy for extending the applicability of this technique to the shape optimization of turbomachinery components, based on intermediate coordinate system transformations, is proposed.

The Volumetric NURBS parameterization is differentiated to provide the grid sensitivities needed by the FI continuous adjoint method. Then, it is used for the GB shape optimization of a 2D transonic airfoil and that of a linear compressor cascade. The method is also applied to the shape optimization of a 3D transonic wing and, finally, that of a peripheral high-pressure turbine nozzle guide vane.

Keywords: Computational Fluid Dynamics, Graphics Processing Units, Continuous Adjoint Method, Shape Optimization, Thermal and Hydraulic Turbomachines, Wind Turbines, Shape Parameterization, NURBS, External and Internal Flows, Evolutionary Algorithms.

Acknowledgements

At this point, I would like to thank all those who have contributed to the completion of this thesis. First of all, I would like to sincerely thank the supervisor of this thesis, Kyriakos Giannakoglou, Professor NTUA, for introducing me as an undergraduate student to the field of turbomachinery and optimization and for giving me the opportunity to work on such a challenging field of research. His constant presence and investment in time and effort throughout the duration of my studies have been of paramount importance for the completion of this work.

I would also like to thank Konstantinos Mathioudakis, Professor NTUA and Ioannis Anagnostopoulos, Professor NTUA for their comments on the thesis presentation and their participation in the advisory committee of my thesis. Additionally, I would like to thank Spyridon Voutsinas, Professor NTUA and Georgios Papadakis, Assistant Professor NTUA, for providing me with data and useful insight in the field of wind turbines.

I owe special thanks to my colleague Dr. Xenofon Trompoukis for being my mentor on the field of GPU programming and CFD and for the countless hours dedicated on the development of the PUMA software. I would also like to thank Dr. Varvara Asouti for introducing me to the field of high performance computing and evolutionary optimization methods. Her contribution on the set-up and maintenance of the HPC CPU and GPU cluster of PCOpt/LTT/NTUA has played an important role in this thesis development. I would also like to specially thank Dr. Evangelos-Papoutsis Kiachagias for his priceless advice on the field of adjoint and deterministic optimization methods. The contribution of Dr. Ioannis Kavvadias and Dr. Dimitrios Papadimitriou is, also, highly appreciated.

I am also deeply grateful to all the other members of the PCOpt/LTT family, namely Konstantinos Samouchos, Dimitrios Kapsoulis, Flavio Gagliardi, James Koch, Konstantinos Gkaragkounis, Ioannis Vryonis and Morteza Monfaredi for the great atmosphere, their friendship and support throughout my PhD research. I would like to thank additionally Dr. Stylianos Kyriacou and, especially, Dr. Evgenia Kontoleontos for their help, by providing useful data and industrial feedback. For the same reason I would like to thank Dr. Georgios Ntanakas and Ilias Vasilopoulos. Finally, I would like to thank Markella Zormpa and Georgios Papageorgiou for their very recent work on the tools developed in this thesis and their feedback, which proved useful for a number of improvements.

Data for the industial applications presented in this thesis were partially provided by ANDRITZ Hydro and Rolls Royce Deutschland. I would also like to acknowledge the General Secretariat of Research and Technology for funding part of the research of the thesis through the project "THALIS-NTUA: Development of know-how for the aeroelastic analysis and the design, optimization of wind turbines".

Last but not least, I would like to thank my family and friends for their moral support and for being there for me whenever I needed them, during this PhD thesis and my whole life up to now.

Contents

Chapter 1

Introduction

During the last decades, the exponential growth of computational power of computers has led to the evolution of methods for the numerical solution of the partial differential equations (PDEs) that govern a variety of physical and social systems, ranging from structural and fluid mechanics, chemistry and nuclear physics, up to the behavior of economic systems.

In particular, the field of fluid dynamics is of paramount importance in a variety of engineering applications, such as weather forecast, automotive design, energy production and aerospace engineering. Despite the ongoing theoretical research on the PDEs governing fluid flows, namely the Navier-Stokes (NS) equations, the analytical solution and even its existence and uniqueness are yet to be found/proven by mathematicians. As a result, methods of Computational Fluid Dynamics (CFD) are the only alternative to experimental processes, providing a cost effective tool that can successfully be used during the design phase of several systems and their components.

The first computer programs modeling fluid flows around aerodynamic bodies (airfoils and aircraft configurations) appeared in the late 1960s, solving potential flow equations. Then, in the 1970s and 1980s, the scientific community proceeded with the development of solvers for the inviscid flow equations (Euler equations) [79, 77] and, then, the first codes for solving the full Navier-Stokes equations appeared [54, 112, 193]. In the turn of the century, numerous commercial codes for solving the Reynolds Averaged Navier-Stokes (RANS) and Unsteady RANS (URANS) equations were already available and widely used by the industry [72, 74, 171, 62, 177]. The next step in the evolution of CFD was the computational cost reduction, allowing for increasingly accurate models to be used (Large Eddy Simulation-LES [170, 35], Direct Numerical Simulation-DNS [132] e.t.c.) as well as incorporating CFD in design optimization processes. For the latter, three aspects are of major importance, namely **a)** high performance computing and evolution of general purpose GPU computing (GPGPU) as well as the development of efficient numerical schemes for the solution of the Navier-Stokes equations, **b)** development of methods for numerical optimization and **c)** the evolution of methods to parameterize and numerically treat geometries of aerodynamic bodies, mainly through Computer Aided (Geometric) Design (CA(G)D) tools.

This thesis addresses all of these three developments and targets at creating methods and computational tools that bring CFD-based design and optimization closer to the industrial reality.

1.1 CFD Tools Running on GPUs

GPUs are parallel co-processors to Central Processing Units (CPUs), originally used to perform computations needed to render graphics on a computer's screen. The chain of tasks that need to be performed in order to render an image on the screen are known as the "graphics pipeline" [2]. The need for increasingly realistic and detailed graphics led to the development of GPUs which offer more Floating Point Operations Per Second (FLOPS) and higher memory bandwidth than CPUs. This is achieved using different hardware architecture, comprising hundreds of specialized cores that can operate in parallel and each one has fast access to its local memory, thus obtaining higher theoretical memory bandwidth. Since all cores process data in parallel, they execute the same instructions, on different data though, stored in their local memory, following the Single Instruction Multiple Data (SIMD) architecture. Such an architecture fits perfectly the needs of graphics processing, since the operations needed to compute each pixel's color on the screen can be carried out independently for each pixel [49, 133]. On the other hand, multi-core CPUs follow the Multiple Instruction Multiple Data (MIMD) architecture, since the instruction chain executed from one core may differ from that executed by another.

Although the SIMD architecture is responsible for the great theoretical performance of GPUs, it also posses serious constraints when it comes to carrying out tasks other than those of the typical graphics pipeline. This is more profound in the earlier generations of GPUs, which had limited amount of cache, limited mechanisms for communication between processes running simultaneously and limited amount of GPU global memory. Another major shortcoming is that programming languages and Application Programming Interfaces (APIs) developed for GPUs were either extremely low-level or oriented exclusively towards graphics processing (e.g. OpenGL, Microsoft DirectX e.t.c.).

Despite these shortcomings, the superior hardware capabilities of GPUs attracted a lot of attention from the scientific community, mainly for exploiting them for general purpose GPU programming (GPUGPU). Numerical solution of PDEs and CFD was one of the first fields of GPGPU to evolve. By achieving noticeable speed-up with respect to (w.r.t.) CPU implementations, GPU-enabled CFD codes can have a great impact on reducing the analysis and design/optimization wall clock time. In addition, this will render higher fidelity and computationally more demanding simulation methods (such as LES, or ultimately DNS) more appealing for industrial use.

The first works related to CFD on GPUs where concerned with linear algebra and solution of basic PDEs on single-block structured meshes [105, 156, 17]. In [59] a GPU implementation of the multigrid technique is presented. The implementations of the algorithm of Stam [176] for computing visually realistic flow fields on the GPU is presented in [65, 118]. One of the first works on a CFD method producing results of useful accuracy for engineers is found in [63] where a GPU capable solver for the Euler equations is presented. In this work, the Cg shader language is used to program the parts of the solver executed on the GPU. Two latter works [20, 42] used the extension of the Brook language, originally for multi-streaming programming, for GPUs named BrookGPU. In both of these works, BrookGPU was chosen instead of any shader language, since it provided a moderate level of abstraction that facilitates general purpose programming. It is jointly pointed out that major reconsideration and restructuring of an existing CFD code is needed in order to achieve high parallel speed-up by the GPU implementation. In [20] the use of NVIDIA Compute Unified Device Architecture (CUDA) is also discussed. It is shown that the capabilities offered by CUDA allow a much more efficient solver in 3D compared to the BrookGPU implementation.

Another application of GPU programming on CFD can be found in Lattice Boltzmann Methods (LBM) [116, 129]. For such methods, there is a large margin for code acceleration, since the interaction necessary between data manipulated by different cores is minimal and mostly local in nature.

The use of GPUs was later extended in CFD problems involving larger meshes by employing more than one GPUs. In [179] a multi-GPU scheme is presented using POSIX threads for inter-GPU communication. In this approach different CPU threads are used to control every GPU. The implementation presented in [20] was also extended to account for multiple GPUs by the same group of authors in [21]. In addition, it is shown that one option to increase the overall memory bandwidth is the extensive use of the shared memory of the GPU, which is normally used for communication between different threads and is also characterized by low latency. The use of multiple GPUs on the same computational node was greatly facilitated by the introduction of Unified Memory Addressing (UVA) in version 4.0 of CUDA. With this version, a single CPU thread can control more than one GPU.

Achieving considerable speed-up by implementing unstructured CFD codes on GPUs posed a whole new challenge to the scientific community. Especially unstructured CFD codes with vertex storage are by far the most exigent case. The reason is that in such a case neither the neighbours of a vertex are know a priori (as in structured meshes) nor the number of neighbours (as in unstructured codes with cell storage). The Parallel CFD & Optimization unit of Lab. of Thermal Turbomachines at National Technical University of Athens (PCOpt/LTT/NTUA), which will be shortly referred to as PCOpt/LTT, tackled this problem in [85] by optimizing the use of available cache memory on the GPU and reconsidering the memory access patterns. A speed-up of approximately 25× for Single Precision Arithmetics (SPA) and 17.5× for Double Precision Arithmetics (DPA) is shown. In the same work, a Mixed Precision Arithmetics (MPA) technique is introduced, which maintains the accuracy of DPA but improves the achieved memory bandwidth reaching a speed-up of 22.5× for 3D inviscid flows. The same group of authors studied several flux computation schemes in [8]. It is there shown that choosing the appropriate flux computation scheme for a certain application is a compromise between parallel speed-up and GPU memory consumption. In [56], an implementation of an unstructured CFD code using the OpenCL programming environment is presented. The importance of the flux and gradient computation schemes is again highlighted. The same group presents an improved version of their implementation in [32], achieving a speed-up of up to 63× compared to the same code on a single CPU core. However, it must be noted that this figure corresponds to a SPA implementation. In addition, the scalability of the solver is also studied when the MPI protocol is used for communication among the GPUs.

Concerning the available APIs for GPGPU, two different options have dominated most current implementations. The first is the NVIDIA CUDA [31] and the second the OpenCL API [73]. OpenCL can be used on heterogeneous platforms consisting of multi-core CPUs, GPUs or even the latest Intel Phi CPU architecture. On the other hand, CUDA is only available for GPU applications running on NVIDIA graphics hardware. Since the computational tools developed in the scope of this thesis make use of the CUDA API, a short description of some basic CUDA terminology follows.

CUDA was officially released by NVIDIA in 2007. The name refers to both the API and the computing platform itself. Three abstractions are at the core of the CUDA programming model, namely a hierarchy of thread groups, shared memories and synchronization barriers. A CUDA thread is the fundamental processing unit that executes a series of instructions. A series of instructions that need to be executed in parallel by numerous threads are organized in a kernel. All threads have read and write access to the global memory of the GPU through the L1 and L2 cache memories. Threads are organized in blocks, which are in turn organized in a grid of thread blocks. Threads of the same block can communicate with each other through shared memory. Parts of global memory can be read with lower memory latency by using texture fetches and the constant memory. A lot of details on the optimal use of each GPU memory space can be found in [182], which the current thesis builds upon.

1.2 Methods for Aerodynamic Shape Optimization

Numerical optimization methods can be categorized based on several criteria, such as computational cost, search strategy, ability to treat constraints or not and many more. The most rational categorization is related to the strategy that each method uses to search for the optimal solution. According to this, optimization methods are classified in stochastic and deterministic ones.

The class of stochastic optimization methods [125, 175] includes, among others, simulated annealing [100], particle swarm optimization (PSO) [97], genetic algorithms (GAs) [48, 33] and evolution strategies (ES) [14]. Evolutionary algorithms (EAs) combine elements of both GAs and ES [124]. EAs, like GAs, are inspired from the theory of evolution of species, proposed by Darwin [34], by mimicking natural processes of parent selection, crossover (recombination) and mutation.

Every candidate solution of the optimization problem is represented by an individual, whose genotype is the set of design variables (also referred to as the design vector), while its phenotype is the objective function value(s). Individuals are typically organized in generations. Parent selection, crossover and mutation operators are applied in each generation, successively producing the best performing (best fit in evolutionary terms) individuals. EAs are advantageous compared to other optimization methods (especially deterministic ones) w.r.t. several aspects. To name a few of them, they can easily be applied to any optimization problem without necessitating specific knowledge of the problem itself, can easily treat constraints, their extension for multi-objective optimization (MOO) problems for computing Pareto fronts of non-dominated solutions is possible [194] and, given an adequate computational budget, they tend to converge to the global optimum even for multi-modal objective functions [200].

However, as any population-based method, they have a major disadvantage, which is the large number of candidate solutions to be generated and evaluated in order to approach the optimum. This can be a significant impediment in case the software needed to evaluate each candidate solution is associated with high computational cost, as is the case for CFD-based aerodynamic optimization. To remedy this, surrogate evaluation models (or metamodels) are employed. Artificial Neural Networks (ANN) or other regression models are used as metamodels, trained on a set of evaluated candidate solutions. Evolutionary algorithms employing metamodels are referred to as Metamodel-Assisted EAs (MAEAs). The use of metamodels can be characterised as either on-line or off-line, based on whether their training takes place during the evolution or not. Another classification of the metamodels used in MAEAs is based on the range of the design space in which their prediction capability is acceptable [80, 22, 91]. Based on this criterion, they are classified in global metamodels used in the entirety of the design space, or local ones that are valid only in the neighborhood of the individual to be evaluated.

Off-line trained metamodels make use of a single (global) surrogate model trained with patterns appropriately selected through a Design of Experiments (DoE) [128, 19]. During the evolution, the off-line trained metamodel is used to inexpensively evaluate each candidate solution. The "optimal" solution(s) is (are) re-evaluated on the problem specific model (PSM), such as the CFD solver in this thesis. Depending on several criteria, the algorithm may terminate or resume the search with a newly trained metamodel (capable of reproducing the re-evaluated solutions).

MAEAs with on-line trained metamodels make combined use of the metamodel and the PSM during the evolution. These tools are employed on the entire population, either periodically or by switching from metamodels to the PSM depending on several criteria. About two decades ago, the PCOpt/LTT has proposed a MAEA based on the Low-Cost Pre-Evaluation (LCPE) as described in [92]. In LCPE, the metamodel is used to pre-evaluate all individuals in a generation and only a few promising ones are evaluated using the PSM.

Another challenging approach for reducing the computational cost of EAs is the use of distributed and hierarchical optimization schemes [37, 93]. In distributed evolutionary algorithms (DEAs or DMAEAs) [84, 199], a number of sub-populations evolve in semi-isolation and exchange information regularly. According to [83], Hierarchical Evolutionary Algorithms (HEAs or HMAEAs) are multilevel schemes categorized in three classes, namely **a)** multilevel evaluation, where evaluation tools of different cost and/or fidelity are used in each level, **b)** multilevel search, where different search methods (e.g. EA and gradient-based, GB, descent methods) are used in each level, and **c)** multilevel parameterization, where different parameterizations are used and each level corresponds to a design space of possibly different dimensionality [198]. Again, populations in each level evolve separately and exchange members or genetic material regularly. The communication among the different levels may be one-way or bidirectional. All of the above variations are available in the general purpose optimization platform EASY (Evolutional Algorithms SYstem) [1] developed by PCOpt/LTT.

When EAs are used for optimization problems with a large number of design variables, the number of evaluations needed to reach the optimal solution may become prohibitively large. Even in the case of MAEAs, the large number of design variables still posses a problem, since the metamodel's computation cost increases non-proportionally and, most importantly, its prediction capability tends to deteriorate (this is the so-called "curse of dimensionality" [13]). To remedy this condition, unsupervised learning techniques have been proposed that selectively reduce the number of design variables. Principal Component Analysis (PCA) [66] has been used successfully in this context, introduced in the EASY platform in [107] and extended later in [89, 87].

Deterministic methods constitute the second class of optimization methods, among which the GB optimization methods are by far the dominant ones. For this reason, in what follows only the latter will be considered. There is a plethora of GB optimization methods ranging from line search [5, 46] to trust region ones [173, 196]. Some of them need information for the objective function gradient (also referred to as sensitivity derivatives, SDs), while others need additional information for the second derivative of the objective function w.r.t. the design variables [18, 46, 26, 25]. GB and, in general, deterministic optimization methods are significantly more computationally efficient compared to stochastic optimization ones [47], since they guarantee that any additional computational effort leads to solution that better approximates the local or global optimum of the problem. A common disadvantage of such methods is that they are easily trapped in local optima (especially in complex optimization problems with high-modality). Especially for line search methods, the convergence rate is greatly affected by the choice of the step-size used to march the solution along the computed objective function gradient direction. An extensive presentation of GB optimization methods can be found in [131].

The most important aspect of GB optimization methods is the capability and cost of computing the objective function gradient. This is more pronounced in problems where the objective function and/or its gradient cannot be computed analytically, as is the case in aerodynamic shape optimization. The common workaround is to employ methods that numerically compute the necessary gradient [145]. The simpler is the Finite Difference (FD) method. In FD, each design variable $b_i, \ i = 1, \ldots, N$ undergoes positive and negative perturbations by a small quantity ϵ_{FD}. The numerical tool that computes the objective function F (i.e. a CFD solver for aerodynamic optimization problems) is, then, used to compute F for each perturbed configuration and the gradient components $\frac{\delta F}{\delta b_i}$ are computed by the quotient

$$\frac{\delta F}{\delta b_i} = \frac{F\left(b_1, \ b_2, \ \ldots, \ b_i + \epsilon_{FD}, \ \ldots, b_N\right) - F\left(b_1, \ b_2, \ \ldots, \ b_i - \epsilon_{FD}, \ \ldots, b_N\right)}{2 \ \epsilon_{FD}}$$

The above equation, approximates the gradient with second-order accuracy w.r.t. to the selected perturbation size ϵ_{FD}. It is obvious that the computation of all the gradient components requires $2N$ calls to the PSM (i.e. the CFD solver). Even for 1$^{\text{st}}$ order accurate computation of the gradient, the cost is half but still proportional to N. Moreover, the gradient accuracy is strongly dependent on the choice to the parameter ϵ_{FD}. Large values result in low accuracy predictions of the gradient, while very small values may lead to numerical errors due to the denominator approaching zero. An alternative to FD is the use of the Complex Variable Method [3, 122, 98], by which the dependency on the value of ϵ_{FD} is circumvented, but the cost of computing the objective function gradient is still proportional to N. Direct Differentiation (DD), in which the PSM is differentiated w.r.t. each design variable is another commonly used method [167, 40, 12]. However, this still has a computational cost which is proportional to N, for the gradient.

Since industrial scale applications typically call for millions, or even tens of millions, of CFD mesh nodes and a large number of design variables are involved in the optimization, the dependency of the cost of computing $\frac{\delta F}{\delta b_i}$ on N may render a GB method prohibitively expensive. The most viable alternative that has attracted a lot of academic and industrial interest in the last three decades, is the adjoint method. The optimization problem can be viewed as a constrained one, where a value-set of the design variables b_i^* is sought that minimizes (or maximizes) F subject to the constraint of satisfying the state equations (the flow equations, in case of aerodynamic optimization) denoted as $R_n = 0$. A dual problem can be formulated by introducing the adjoint (or dual or co-state) variables Ψ into the Lagrangian function $F_{\text{aug}} = F + \Psi_n R_n$. Differentiation of F_{aug} w.r.t. b_i and elimination of derivatives of flow quantities w.r.t. b_i yields a set of adjoint (or co-state) equations. The cost of solving these equations is about the cost of a single run of the PSM, effectively making the cost of computing the objective function gradient independent of N. The adjoint method was first introduced by Lions in 1971 [117], though it was not until 1984 that Pironneau applied the adjoint methods in aerodynamics, for potential flow problems [150]. Later, Jameson mathematically developed [76] and applied [153, 78] the adjoint method for flows governed by the compressible Euler equations. From that point on, the field of adjoint methods for CFD-based optimization flourished, with the introduction of different methods to develop the adjoint equations and different expressions to compute the SDs.

Adjoint methods for CFD are categorized in two main approaches, depending on whether the flow equations are first discretized and then differentiated (discrete adjoint) or vice-versa (continuous adjoint).

Discrete adjoint methods are further distinguished by the approach chosen to differentiate the discretized system of flow equations. The discretized residuals of the flow equations can be differentiated "by hand" or by using code transformation tools and/or operator overloading. The latter approach is widely known as Algorithmic Differentiation (AD) [61], and there are several tools available for transforming a code that computes the flow residuals and solves the flow equations into one that solves the adjoint ones, such as TAPENADE [172], ADIFOR [123], ADOL-C [191] to name a few. Since discrete adjoint is not the focus of this thesis, an extensive description and literature review of related research is omitted. However, it is worth noting the great advantage of discrete adjoint methods of computing $\frac{\delta F}{\delta b_i}$ with extreme accuracy and maintaining the convergence rate of the flow equations [55]. The main drawback of "hand-differentiated" discrete adjoint methods is the large investment in code development, while for the AD approach, the memory consumption for the solution of the adjoint equations is typically 3-5 times that of solving the flow equations [38].

In the continuous adjoint approach, the adjoint equations, resulting by differentiating the flow governing PDEs, are similar to the flow PDEs themselves (though linear) and, consequently, similar techniques can be used for their numerical solution. However, the expressions arising for the computation of $\frac{\delta F}{\delta b_i}$ are not unique. One option is to differentiate F_{aug} by employing the Leibniz rule and obtain expressions for $\frac{\delta F}{\delta b_i}$ solely on the boundaries of the CFD domain, neglecting at the same time boundary terms containing the residuals of the flow equations (referred to as the severed Surface Integral, severed-SI, approach) [137], while an alternative leads to expressions that involve also field integrals on the CFD domain (Field Integral, FI, approach) [140]. An extensive study of the impact that the choice between the SI and FI approach has on the accuracy of the computed gradient is found in [95] and the concept of the adjoint grid displacement is introduced to bridge the gap between the two methods. There, the inability of the severed-SI approach to consistently match the accuracy of the FD method (used as reference) and the strong dependency of the accuracy on the grid density and quality is underlined. Thus, in terms of accuracy the FI approach proved to be more reliable, accompanied, however, by more complex expressions for the objective function gradient. Incorporating the adjoint grid displacement equations led to the introduction of the Enhanced-Surface Integral (E-SI) approach [96], resulting in integrals expressed only along the boundaries of the CFD domains, while matching at the same time the accuracy of the FI approach. Depending on the parameterization method used, the E-SI approach (that involves the solution of an additional adjoint equation) may prove more computationally expensive compared to the FI one. Thus, in most applications presented in this thesis, the FI approach will be employed.

Another aspect of the continuous adjoint approach that has been shown to have a significant impact on the accuracy of the computed objective function gradient, is the differentiation of the turbulence model (if any) used together with the flow equations. Early developments of the continuous adjoint method followed the "frozen turbulence" assumption, where variations in the eddy viscosity are neglected [139, 9]. In [202] by PCOpt/LTT, this assumption is overcome for the first time by differentiating the Low-Reynolds Spalart-Allmaras turbulence model. Later in [203] by the same group, the High-Reynolds k-ε turbulence model was differentiated introducing the notion of adjoint wall functions. Both these developments were proven to improve the accuracy of the computed gradient. A series of works by PCOpt/LTT on the continuous adjoint to different turbulence models followed [142, 94]. In [23], the continuous adjoint approach to the Spalart-Allmaras model for compressible flows was presented. This is the first work on continuous adjoint to turbulence models from a group other than PCOpt/LTT. In this work, variations of the distance from the wall were also taken into account by incorporating the Eikonal equation in the adjoint formulation, avoiding the "frozen distance" assumption. The effects of the "frozen distance" assumption and a systematic approach to circumvent them are also studied in [141]. The aforementioned considerations are taken into account in the continuous adjoint development presented in this thesis.

Even though it not among the interests of the current thesis, it is worth noting that an active field of research on adjoint methods for CFD is the development of adjoint methods for unsteady flows. The major obstacle when dealing with unsteady flows is that the adjoint equations march backwards in time, demanding either an excessive amount of data storage for the flow field instants or, an excessive amount of flow field re-computations. Techniques of flow field reconstruction and data compression such as Singular Value Decomposition (SVD), Proper Orthogonal Decomposition (POD) and many other have been employed [195] and are currently under development [159, 190] for overcoming this problem.

Hybrid optimization methods combining stochastic and deterministic methods also exist [50] and have been employed for aerodynamic shape optimization. In some approaches, GB methods are employed to improve the resulting optimal solution(s) of an EA-based optimization [99]. This is usually referred to as post-hybridization [169]. In other works, EAs are combined with local search algorithms employed during the evolution, to yield the class of memetic algorithms (MAs) [130, 166]. An open issue exists when using hybrid algorithms for MOO problems. This arises from the need to either combine the different optimization objectives in a single function that the GB method can treat [83, 103] or split the GB problem in as many sub-problems as the number of objectives and optimize for each objective with the rest acting as constraints [99]. A different approach making use of a PCA-assisted MAEA with the steepest-descent method assisted by the continuous adjoint is presented in [90]. There, the PCA technique is applied in the objective space in order to combine the optimization objectives into a single function.

1.3 Parameterization Techniques for Aerodynamic Shape Optimization

A component of paramount importance for aerodynamic shape optimization is the shape parameterization technique used. This is responsible for translating the set of design variables in an aerodynamic shape. Since the objective function value for each candidate shape is to be evaluated by means of CFD, the shape is ultimately defined by the set of grid nodes on the boundary of the CFD domain. An extension of the notion of parameterization to include also the internal nodes of a CFD volume mesh is also valid.

For a given number of design variables, the shape parameterization method greatly affects the shapes that arise throughout the optimization and, consequently, affects the convergence rate of the whole process as well as the quality of the optimal solution. A poor parameterization method may lack the ability to produce a shape with optimal performance or may lead to unreasonably complex shapes.

Another important aspect associated with the choice of a parameterization method is the ease of computing the so-called geometric sensitivities. These are the derivatives of the CFD nodal coordinates, describing the shape to be optimized, w.r.t. the values of the design variables. In case of GB optimization methods, this information is necessary. Their accuracy and the cost for obtaining them may greatly affect the optimization convergence rate and the overall optimization cost.

A survey of shape parameterization methods for CFD and Computational Structural Mechanics (CSM) based optimization is presented in [158]. There, parameterization techniques are categorized in eight different approaches, namely the **a)** basis vector **b)** domain element **c)** PDE-based **d)** discrete **e)** polynomial and Spline **f)** CAD-based **g)** analytical and **h)** Free Form Deformation (FFD) approaches.

For a description of the basis vector, PDE-based and analytical approaches the reader is referred to [146, 114], [15] and [68, 41, 64], respectively. In short, the basis vector approach uses proposed shapes (and design vectors) to create a vector basis and, then, each subsequent shape is obtained from the baseline by superimposing a linear combination of the proposed designs. The PDE-based approach approximates the surface to be parameterized as an iso-surface obtained from the solution of a particular PDE. Finally, the analytical approach employs analytical functions that are superimposed on the baseline shape in order to modify it. Among these three methods, the analytical approach has recently received a lot of attention since it is used for modeling manufacturing imperfections for optimization under uncertainty [106]. The reason is that the analytical method can be tuned to create shapes with bumps following predefined probability density functions (obtained by statistical analysis of the manufacturing process results).

The discrete approach is the simplest one since CFD boundary grid nodes can move independently. The complexity of the shape that can be achieved is only limited by the number of CFD mesh nodes on the surface of the object whose shape is to be optimized. Nonetheless, the number of design variables may be extremely large (a major disadvantage for use with population based optimization methods) and the manufacturability of the final shape is questionable. In addition, the resulting CFD surface mesh typically requires additional smoothing before proceeding to the volume mesh deformation.

Polynomial and Spline approaches are the most widely used methods to date, especially for optimization of 2D aerodynamic shapes, such as airfoils, ducts and turbomachinery blade sections. They are mostly based on Bézieror B-Spline polynomials [147, 45], with Non-Uniform Rational B-Splines (NURBS) [149] being the most general form among them. They can provide different levels of shape complexity, based on the polynomial degree used and/or the number of control points, with direct control over the shape smoothness. In addition, geometric sensitivities can be obtained directly by differentiating the underlying polynomial or rational formula.

Free-Form Deformation is an approach inspired by the field of interactive design and soft object animation. FFD techniques can rely on physically based modeling by operations such as translation, scaling, rotation e.t.c. [10] and have been, additionally, extended to apply these operations on the whole space in which the object to be manipulated is embedded [164]. Such an approach is based on trivariate Béziervolumes which can be either linear in each parametric coordinate or of higher degree [30], or on NURBS with multiple blocks to handle more complex shapes. In addition, several FFD techniques have been developed that allow the direct manipulation of the predefined points on the shape to be morphed instead of manipulating the control points [70].

Domain element methods [152] use a set of points forming a polygon (polyhedron in 3D) to control the shape to be deformed. Points' displacement is interpolated in the interior of the domain defined by the control polygon by an inverse mapping. The harmonic coordinates method [81] is an example of this class.

Finally, CAD-based approaches provide the best compromise between complexity and manufacturability but, in the general case, are difficult to incorporate in a fully automated optimization loop. However, for a certain area of applications, specialized CAD tools can be developed and are, in fact routinely used in automatic optimization loops. An example is the area of aircraft wing design, where CAD models based on basic operations such as airfoil positioning and skinning, can be used. Another interesting field of application is the design/optimization of turbomachinery components. Several parametric turbomachinery blade design tools have been developed [101, 75, 168, 60]. Most of them use a bottom-up process starting from the meridional flow path definition, defining airfoil sections at several positions along the blade span and, then, creating the blade surface through skinning [126, 155]. Depending on the choice of design variables for each of these steps, simple up to very complicated blade shapes can be generated.

In the current thesis, several parameterization methods are developed lying on the CAD-based, Spline or FFD approach. The advantages and disadvantages of each approach in combination with the optimization method used are thoroughly discussed throughout the rest of the thesis.

1.4 Thesis Outline

This PhD thesis emphasizes, without though being restricted to, turbomachinery shape optimization applications. In the chapters of this thesis, the development of an efficient GPU-enabled CFD software, the continuous adjoint method and different shape parameterization techniques are presented.

Chapter 2 is concerned with the developments-extensions made on the flow solver PUMA (Parallel Unstructured Adjoint Multi-row) of PCOpt/LTT. PUMA is a GPU-enabled flow and adjoint solver for compressible and incompressible flows, in external and internal aerodynamics as well as flows inside thermal and hydraulic turbomachines [104, 108, 6, 197, 182]. The URANS equations are presented, together with the most commonly used boundary conditions. Then, the discretization and numerical solution of the flow equations is presented. The development of the artificial compressibility (AC) method for solving incompressible flows with the same software, is another focal point of Chapter 2. Attention is also drawn on the GPU implementation of PUMA. The programming and numerical techniques that render PUMA approximately $40\times$ faster than a CPU solver are thoroughly described. At the end of Chapter 2 a series of CFD benchmark cases, used for the purpose of validation and verification, are presented. Among them, the flow field around an Horizontal Axis Wind Turbine (HAWT) runner blade is computed and results are compared to these obtained by other CFD codes and experimental data.

In Chapter 3, the continuous adjoint method is presented. Both the SI and FI approaches are developed for compressible flows, while the FI approach is also developed for incompressible flows, where the adjoint to the AC method is introduced. The adjoint to the Spalart-Allmaras (SA) turbulence model is presented for incompressible and compressible flows. The derivation of the adjoint boundary conditions on different kinds of CFD boundaries is thoroughly examined.

In Chapter 4 the fundamental concepts of shape parameterization are introduced, together with numerical tools typically used for this purpose. NURBS curves and surfaces are the focus, since they are extensively used in the shape parameterization tools presented in the thesis.

Chapter 5 is concerned with the development of the software GMTurbo (Geometry Modeler for Turbomachines). The conformal mapping of surfaces of revolution is presented, which is used extensively in order to treat all types of turbomachinery geometries, namely axial, mixed flow and radial blade rows. Then, the methodology followed to construct turbomachinery blades starting from definition of metal angles, camber surface construction and proceeding with superimposing thickness is presented. The chapter ends up with the ensemble of parameters that define the blade geometry and can be used as design variables in the optimization.

An FFD shape parameterization approach based on volumetric NURBS is presented in Chapter 6. This approach is also used, in order to undertake the parameterization and, consequently, the deformation of the aerodynamic shape and the CFD mesh around it in a monolithic manner. The difficulty of applying such a parameterization approach to the optimization of turbomachinery components, due to axisymmetry and the presence of periodic boundaries, is identified and a novel extension is, then, introduced to overcome this shortcoming. The developed method is differentiated in order to by coupled with the FI continuous adjoint approach for the optimization of wings and turbomachinery blades.

Chapter 7 is concerned with the optimization of single- and multi-row turbomachinery configurations using EAs. The GMTurbo design and parameterization tool, presented in Chapter 5, is used to generate candidate solutions throughout the optimization. A propeller type water turbine consisting of the guide vanes (GV) and runner is optimized for maximum efficiency. The optimization is carried out in three steps, namely **a)** optimizing the GV geometry coupled with the baseline runner, **b)** optimizing the runner geometry with baseline GV one and **c)** concurrent optimization with combined modification of the GV and runner.

In Chapter 8 applications of gradient-based optimization by means of the continuous adjoint method developed in Chapter 3 are presented. The volumetric NURBS method developed in Chapter 6 is used to parameterize and control the aerodynamic shape and the CFD mesh around it. A transonic airfoil is optimized separately for minimum drag and maximum lift. Then, a 2D linear compressor cascade is optimized for minimum total pressure losses between the inlet and outlet. Concerning optimization of 3D geometries, the drag minimization of a transonic wing and, finally, the minimization of the mass averaged total pressure losses through a high-pressure turbine nozzle guide vane (HPT-NGV) are presented.

The conclusions of the thesis are drawn in Chapter 9 together with some ideas and recommendations for future developments associated with the current work.

Finally, supplementary tools supporting mostly the EA-based optimization loops presented in Chapter 7 are presented in the Appendices.

Chapter 2

The Navier–Stokes Equations and Their Numerical Solution

The main focus of this thesis is the aerodynamic optimization using gradient-based techniques assisted by the continuous adjoint method. Before proceeding to the development of the continuous adjoint method and other tools required by the optimization workflow, the flow model (equations, discretization schemes, numerical solution) used to evaluate the performance of each candidate solution resulting during the optimization process is presented. The flow model consists of the RANS equations (or URANS for unsteady problems) for either compressible or incompressible flows and the Spalart–Allmaras turbulence model (to effect closure in turbulent flows). Since, a large number of the problems to be dealt with in this thesis are concerned with turbomachinery applications and rotating blade rows, the equations are expressed w.r.t. a (relative) non–inertial frame of reference rotating at a constant speed. The CFD solver together with its adjoint counterpart (to be presented in Chapter 3) is called PUMA. Code development started about two decades ago in the framework of a number of PhD theses of PCOpt/LTT ([197],[6],[182]) and, during the last years, it has been transferred to GPUs and enriched with new features and capabilities. This PhD thesis relies exclusively upon the GPU-enabled variant of PUMA.

In this chapter, the discretization of the governing flow equations is presented followed by the numerical methods used to solve the discretized system of equations. Finally, issues arising from the implementation of the numerical solution on GPUs are addressed along with GPU specific techniques to speed-up the process of the numerical solution.

2.1 The URANS Equations for Compressible Flows

Let a coordinate system $O(x_1\ x_2\ x_3)$ which rotates at a constant rotation speed $\omega_m\ (m = 1, 2, 3)$, be defined. The Navier–Stokes equations for compressible flows are then expressed as

$$R_n^{\mathrm{MF}} = \underbrace{\frac{\partial U_n}{\partial t}}_{\mathrm{MF}^t} + \underbrace{\frac{\partial f_{nk}^{\mathrm{inv}}}{\partial x_k}}_{\mathrm{MF}^{inv}} - \underbrace{\frac{\partial f_{nk}^{\mathrm{vis}}}{\partial x_k}}_{\mathrm{MF}^{vis}} + \underbrace{S_n}_{\mathrm{MF}^s} = 0 \tag{2.1.1}$$

In Eq. 2.1.1, U_n stands for the conservative flow variables namely $U_n = \left[\rho \;\; \rho v_1^A \;\; \rho v_2^A \;\; \rho v_3^A \;\; \rho E\right]$, with ρ being the fluid density, $v_m^A\,(m = 1,2,3)$ being the velocity components w.r.t. the absolute/inertial frame of reference and E the energy per unit mass. Inviscid fluxes f_{nk}^{inv}, the viscous fluxes f_{nk}^{vis} and the source terms S_n (corresponding to the Coriolis force) are defined as

$$f_{nk}^{\mathrm{inv}} = \begin{bmatrix} \rho v_k^R \\ \rho v_1^A v_k^R + p\delta_{1k} \\ \rho v_2^A v_k^R + p\delta_{2k} \\ \rho v_3^A v_k^R + p\delta_{3k} \\ \rho h_t v_k^R + v_k^F p \end{bmatrix} \quad f_{nk}^{\mathrm{vis}} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \\ v_\ell^A \tau_{\ell k} + q_k \end{bmatrix} \quad S_n = \begin{bmatrix} 0 \\ \rho \varepsilon_{1\ell k}\omega_\ell v_k^A \\ \rho \varepsilon_{2\ell k}\omega_\ell v_k^A \\ \rho \varepsilon_{3\ell k}\omega_\ell v_k^A \\ 0 \end{bmatrix} \tag{2.1.2}$$

where p stands for the static pressure. In Eqs. 2.1.1-2.1.2 and the rest of the thesis, the Einstein convention is employed according to which summation is indicated by repeated indices in the same term. The relative velocity components v_m^R are linked to the absolute ones v_m^A through the equation $v_m^A = v_m^R + v_m^F$, with $v_m^F = \varepsilon_{m\ell k}\omega_k\left(x_k - x_k^C\right)$ being the rotating/non-inertial frame velocity and x_k^C the position vector of the center of rotation. Eqs. 2.1.2 are supplemented by the following definitions:

- (k,m) component of the viscous stress tensor (τ_{km}) for Newtonian fluid:

$$\tau_{km} = \frac{\mu + \mu_t}{\mathrm{Re}_0}\left(\frac{\partial v_k^A}{\partial x_m} + \frac{\partial v_m^A}{\partial x_k} - \frac{2}{3}\delta_{km}\frac{\partial v_\ell^A}{\partial x_\ell}\right) \tag{2.1.3}$$

- Reynolds number (Re_0) arising when the equations are solved in their non-dimensional form. Specifically, let the velocity field be normalized by a reference velocity (v_{ref}), the density by a reference density (ρ_{ref}), all lengths by a reference length (l_{ref}) and the molecular viscosity by a reference viscosity (μ_{ref}). Then,

$$\mathrm{Re}_0 = \frac{\rho_{\mathrm{ref}}\, v_{\mathrm{ref}}\, l_{\mathrm{ref}}}{\mu_{\mathrm{ref}}}$$

This must not be confused with the actual Reynolds number $\mathrm{Re} = \frac{\rho v l}{\mu}$.

- k component of the heat flux (q_k):

$$q_k = \frac{\mathcal{C}_p}{\mathrm{Re}_0}\left(\frac{\mu}{\mathrm{Pr}} + \frac{\mu_t}{\mathrm{Pr}_\mathrm{t}}\right)\frac{\partial T}{\partial x_k} \tag{2.1.4}$$

  where $\mathcal{C}_p$ is the specific heat under constant pressure.

- Static temperature (T). Since the fluid is assumed to be a perfect gas, T is related to pressure and density through the equation of state:

$$p = \rho R_g T, \text{with } R_g \text{ being the specific gas constant} \tag{2.1.5}$$

- Prandtl number (Pr) and turbulent Prandtl number (Pr_t)

- Specific heat ratio (γ):

$$\gamma = \frac{\mathcal{C}_p}{\mathcal{C}_v} \tag{2.1.6}$$

  where $\mathcal{C}_v$ is the specific heat under constant volume.

- Total or stagnation enthalpy (h_t):

$$h_t = E + \frac{p}{\rho} \tag{2.1.7}$$

  For a perfect gas, the total enthalpy is linked to pressure (p), density (ρ) and velocity ($v_\ell^A, \ell = 1, \ldots, 3$) through

$$h_t = \frac{\gamma p}{\rho(\gamma - 1)} + \frac{1}{2}v_\ell^A v_\ell^A \tag{2.1.8}$$

- Absolute Mach number (M):

$$M = \frac{\sqrt{v_\ell^A v_\ell^A}}{c} \tag{2.1.9}$$

  where c is the local speed of sound, which for perfect gases is given as $c = \sqrt{\gamma R_g T}$.

- Total or stagnation temperature (T_t):

$$T_t = T + \frac{v_\ell^A v_\ell^A}{2\mathcal{C}_p} \tag{2.1.10}$$

- Total or stagnation pressure for perfect gases (p_t):

$$p_t = p\left(1 + \frac{\gamma - 1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}} \tag{2.1.11}$$

- Dynamic viscosity (μ) related to temperature through Sutherland's formula [178]

$$\mu = \mu_0 \frac{T_0 + \mathrm{Su}}{T + \mathrm{Su}}\left(\frac{T}{T_0}\right)^{\frac{3}{2}} \tag{2.1.12}$$

  where Su is an effective temperature value usually called Sutherland temperature, T_0, μ_0 are the reference temperature and reference dynamic viscosity, respectively and μ is linked to the kinematic viscosity ν through $\mu = \rho\nu$. The Sutherland's law is used optionally. For air, $\mu_0 = 1.716 \times 10^{-5}\,\mathrm{kg\,m^{-1}\,s^{-1}}$, $T_0 = 273.15\,\mathrm{K}$ and $\mathrm{Su} = 110.4\,\mathrm{K}$.

Turbulent viscosity μ_t is computed by employing the one-equation Spalart–Allmaras turbulence model [174]. According to this model, an additional PDE is solved for the turbulence field $\tilde{\nu}$, namely

$$\begin{aligned} R^{SA} = & \underbrace{\frac{\partial(\rho\tilde{\nu})}{\partial t}}_{\mathrm{SA}^t} + \underbrace{\frac{\partial\left(\rho\tilde{\nu}v_k^R\right)}{\partial x_k}}_{\mathrm{SA}^c} \underbrace{- \frac{\rho}{\mathrm{Re}_0\,\sigma}\left\{\frac{\partial}{\partial x_k}\left[(\nu + \tilde{\nu})\frac{\partial\tilde{\nu}}{\partial x_k}\right] + c_{b_2}\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k}\right\}}_{\mathrm{SA}^d} \\ & \underbrace{- \rho c_{b_1}(1 - f_{t_2})\tilde{S}\tilde{\nu} + \frac{\rho}{\mathrm{Re}_0}\left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\triangle}\right)^2}_{\mathrm{SA}^s} \end{aligned} \tag{2.1.13}$$

where $\triangle$ stands for the distance of each point within the flow domain from the closest wall boundary. Solving Eq. 2.1.13, μ_t is computed from $\tilde{\nu}$ by $\mu_t = \rho\tilde{\nu}f_{v_1}$. Eq. 2.1.13 is supplemented by the following relations and constants [174]:

$$\begin{aligned} & \chi = \frac{\tilde{\nu}}{\nu}, \quad f_{v_1} = \frac{\chi^3}{\chi^3 + c_{v_1}^3}, \quad f_{v_2} = 1 - \frac{\chi}{1 + \chi f_{v_1}}, \quad S = \sqrt{\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_m^A}{\partial x_\ell}\frac{\partial v_r^A}{\partial x_q}}, \\ & \tilde{S} = S + \frac{\tilde{\nu}f_{v_2}}{\mathrm{Re}_0\,\kappa^2\,\triangle^2}, \quad f_w = g\left(\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}^6}\right)^{\frac{1}{6}}, \quad g = r + c_{w_2}\left(r^6 - r\right), \\ & r = \min\left(10, \frac{\tilde{\nu}}{\mathrm{Re}_0\,\tilde{S}\kappa^2\,\triangle^2}\right), \quad \tilde{\mu} = \rho\tilde{\nu}, \quad f_{t_2} = c_{t_3}e^{-c_{t_4}\chi^2}, \\ & c_{v_1} = 7.1, \quad c_{b_1} = 0.1355, \quad c_{b_2} = 0.622, \quad c_{w_1} = \frac{c_{b_1}}{\kappa^2} + \frac{1 + c_{b_2}}{\sigma}, \end{aligned} \tag{2.1.14}$$

$$c_{w_2} = 0.3, \quad c_{w_3} = 2.0, \quad \sigma = \frac{2}{3}, \quad \kappa = 0.41, \quad c_{t_3} = 1.2, \quad c_{t_4} = 0.5$$

2.2 Boundary Conditions for Compressible Flows

In order to fully define the flow problem, Eqs. 2.1.1 and 2.1.13 must be accompanied by a set of appropriate boundary conditions. These conditions vary, depending on the type of each physical boundary and whether they are strongly or weakly imposed. For example, on the symmetry boundary, the symmetry conditions are imposed, i.e. $\frac{\partial U_m}{\partial x_k}\mathsf{n}_k^{\text{sym}} = 0, (m = 1,2,3,4,5)$ and $\mathsf{n}_m^{\text{sym}}, (m = 1,2,3)$ the components of the normal to the symmetry plane.

Along pairs of periodic boundaries, appropriate periodic conditions are imposed. For linear cascades, two points are assumed to be periodically paired if they have two of their coordinates equal and differ on the third by the cascade pitch. In such cases all flow variables (scalar and vector ones) must be equal between periodically paired points. In case of peripheral rows, two points are periodically paired if their projections on the meridional plane coincide and their circumferential position differs by the blade row pitch. Between paired points all scalar quantities are the same, while every vector and tensor quantity (e.g. velocity or spatial derivatives) is rotated by the row pitch.

Wall, inlet and outlet boundaries are analyzed separately in the following subsections. Farfield boundaries are treated as a combination of inlet and outlet boundary, depending on the local velocity field. If flow enters the domain the boundary is locally treated as inlet, otherwise it is treated as outlet.

2.2.1 Wall boundary conditions

For slip walls, the no-penetration condition applies, namely the normal component of the relative to the wall velocity is set to zero. Let the velocity of the wall boundary be denoted by $v_k^W, (k = 1,2,3)$ (i.e. $v_k^W = 0$ for stationary walls and non-zero otherwise), then the no-penetration condition is expressed as

$$v_k^A \mathsf{n}_k = v_k^W \mathsf{n}_k \tag{2.2.1.1}$$

For no-slip wall boundaries, the absolute velocity is set equal to the wall boundary velocity (Eq. 2.2.1.2). In addition, the turbulence variable $\tilde{\nu}$ is set to zero in case the turbulent boundary layer is resolved down to the wall (Low-Reynolds approach, Eq. 2.2.1.3).

$$v_k^A = v_k^W \tag{2.2.1.2}$$
$$\tilde{\nu} = 0 \tag{2.2.1.3}$$

In case the mesh density is not adequate for fully resolving the boundary layer down to the wall, the wall function technique is employed as described by [27].

Considering the thermal conditions for the wall boundaries, depending on the case, these can be **a)** adiabatic, **b)** constant temperature (T^W) or **c)** constant heat flux (q^W), which can respectively be written as

$$\begin{aligned} q_k \mathsf{n}_k &= 0 \\ T &= T^W \\ q_k \mathsf{n}_k &= q^W \end{aligned} \tag{2.2.1.4}$$

2.2.2 Inlet Boundary Conditions

For subsonic inlet boundaries, the total pressure (p_t^{IN}), total temperature (T_t^{IN}), and inlet absolute velocity direction are specified. The inlet velocity direction is given in terms of two angles, namely θ_1^{IN} and θ_2^{IN}. For the sake of simplicity, let x_3 be the axial direction (i.e. the axis of rotation for peripheral turbomachinery cases). For linear cascade cases, the inlet velocity components v_k^A, $(k = 1, 2, 3)$ are computed as

$$\begin{aligned} v_1^A &= |v_\ell^A| \sin\theta_1 \\ v_2^A &= |v_\ell^A| \cos\theta_1 \sin\theta_2 \\ v_3^A &= |v_\ell^A| \cos\theta_1 \cos\theta_2 \end{aligned} \tag{2.2.2.1}$$

Since, within PUMA, the flow equations are always solved w.r.t. the Cartesian coordinate system, for peripheral rows Eqs. 2.2.2.1 provide the means to compute the absolute velocity components in cylindrical coordinates (r, θ, z). Specifically, v_1^A is replaced by the radial velocity component (v_r^A), v_2^A by the peripheral (v_θ^A) and v_3^A by the axial one (v_z^A). Then, the Cartesian velocity components are computed by applying an inverse transformation.

Thus, for the inlet boundaries, four quantities are specified and a fifth one has to be extrapolated from the flow domain. The usual options are to extrapolate **a)** the static pressure, **b)** the absolute velocity magnitude or **c)** the local Mach number. Then, Eqs. 2.1.9-2.1.11 are used to compute all the necessary flow quantities. For supersonic inlet conditions, all five necessary flow quantities are set.

Concerning turbulence, the inlet turbulence level is prescribed either by setting the inlet turbulence variable $\tilde{\nu}^{IN}$ or by setting the viscosity ratio $\left(\frac{\nu_t}{\nu}\right)^{IN}$.

2.2.3 Outlet Boundary Conditions

To maintain a well–posed boundary value problem, no flow quantity is specified at the outlet in case of supersonic flow conditions, or a single flow quantity is specified for subsonic flow conditions. In the latter case, the quantity to be set can be **a)** the outlet static pressure distribution, **b)** the outlet mean static pressure or **c)** the outlet mass flow rate.

For the last two options, since only the value of an integral quantity over the whole outlet boundary is set, values for pressure and normal to the outlet velocity, respectively, are computed iteratively by uniformly correcting the ones extrapolated from the fluid domain, so as to achieve the prescribed integral quantity. The remaining four flow quantities are extrapolated from the fluid domain. For the turbulence model, the outlet is assumed to be convective and a zero Neumann boundary condition is applied ($\tilde{\nu}$ is extrapolated from the fluid domain).

2.3 Discretization of the Governing Equations

Eqs. 2.1.1 and 2.1.13 form a system of PDEs of mixed type in space and time. The temporal term $\frac{\partial U_n}{\partial t}$ together with the convection terms $\frac{\partial f_{nk}^{\text{inv}}}{\partial x_k}$ correspond to a hyperbolic system. However, the diffusive terms $\frac{\partial f_{nk}^{\text{vis}}}{\partial x_k}$ correspond to a system of elliptic PDEs. Hence, the character of the system of equations depends on the term that dominates the phenomenon and is associated with the Reynolds number. Since, in most practical applications, this thesis is dealing with, the flow is turbulent and, thus, associated with high Reynolds number, the hyperbolic character dominates over the elliptic one. Consequently, Eqs. 2.1.1, 2.1.13 are solved using a time–marching technique appropriate for the solution of systems of hyperbolic PDEs.

According to the time–marching technique, when a steady state solution is sought, the time derivative (which would otherwise exist in the unsteady equation) is replaced by a pseudo-time derivative. In order to distinguish between real and pseudo-time, the latter is denoted by τ. At each pseudo–time step τ_j, the system is linearized, discretized and solved for the correction of the field variables for the next pseudo–time step $(U_n^{j+1} - U_n^j)$. Upon convergence of the system, the pseudo-time derivative vanishes and the original system of the steady-state equations is retrieved.

In case of time-dependent problems, the time-derivative remains in the initial equations and the pseudo–time derivative is added on top of that. The system of

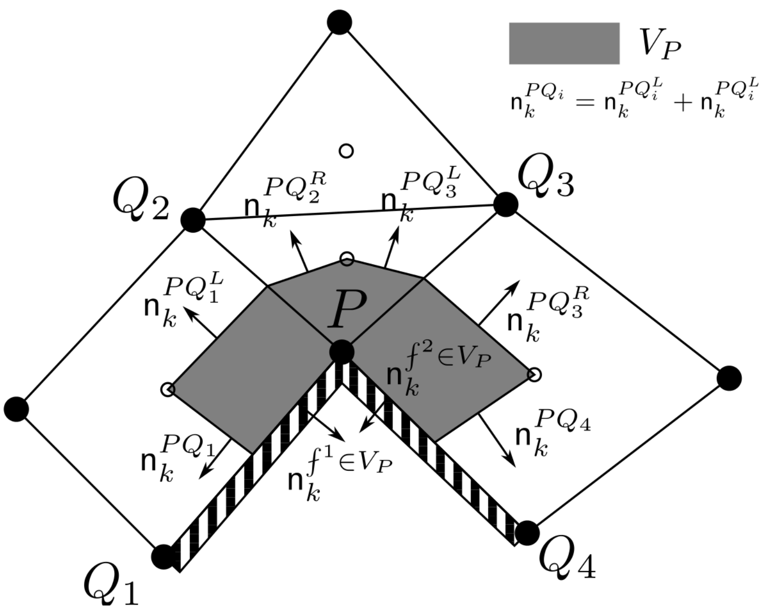


Figure 2.1: A vertex–centered finite volume formed around node P. A boundary node was chosen, so as to cover also the boundary treatment. The normal vectors (n_k) on the finite volume interface are also shown.

equations is linearized and then marched in time, with intermediate pseudo-time steps. This corresponds to the so–called dual time stepping approach. In what follows, the equations are written without the pseudo–time derivative. This term will be added after the equations have been discretized.

For the spatial discretization of Eqs. 2.1.1 and 2.1.13 the vertex–centered variant of the finite volume technique in used on unstructured meshes, consisting of tetrahedra, pyramids, prisms and hexahedra. A finite volume is formed around each mesh node P at real-time step i by connecting the edges midpoints, face centers and element barycenters of the edges, faces and elements attached to this node, respectively. An example of the finite volume formed around node P is shown in Fig. 2.1, for a 2D case. A figure for the 3D case would be very confusing so it is omitted.

For details on the formation of the finite volume and the computation of the geometric quantities, the reader is referred to [197],[6]. It must be noted that the mesh may undergo rigid body motion or a more general deformation in time and so do the finite volumes the discretization is based upon.

2.3.1 Discretization of the Inviscid Terms

Applying the Green–Gauss theorem to the integral of the inviscid terms, for the finite volume of node P, one obtains

$$\int\limits_{V_P^{t_{i+1}}} \frac{\partial f_{nk}^{\text{inv}}}{\partial x_k}\mathrm{d}V = \int\limits_{\partial V_P^{t_{i+1}}} f_{nk}^{\text{inv}}\hat{\mathsf{n}}_k\mathrm{d}\left(\partial V\right) \tag{2.3.1.1}$$

where $\partial V_P^{t_{i+1}}$ is the boundary of the finite volume formed around node P at real–time step t_{i+1} and $\hat{\mathsf{n}}_k\left(t\right), \left(k=1,2,3\right)$ the corresponding unit normal (pointing outwards) components. This is discretized as

$$\int\limits_{\partial V_P^{t_{i+1}}} f_{nk}^{\text{inv}}\hat{\mathsf{n}}_k\mathrm{d}\left(\partial V\right) \simeq \sum_{\forall Q\in\mathcal{N}(P)} \Phi_n^{\text{inv},PQ} + \sum_{\forall f\in\mathcal{B}(P)} \Phi_n^{\text{inv},f\in V_P} \tag{2.3.1.2}$$

where $Q \in \mathcal{N}\left(P\right)$ is a node neighbouring P, $f \in \mathcal{B}\left(P\right)$ is a boundary face (if any) emanating from node P. In case of mesh elements other than tetrahedra, only the nodes connected to P through a mesh edge are assumed as neighbours. For clarity, these are sometimes referred to as direct neighbours while other nodes of the same element are called virtual or indirect neighbours of P. $\Phi_n^{\text{inv},PQ}$ is computed using Roe's approximate Riemann solver [154] as

$$\Phi_n^{\text{inv},PQ} = \frac{1}{2}\left(f_{nk}^{\text{inv},P} + f_{nk}^{\text{inv},Q}\right)\mathsf{n}_k - \frac{1}{2}\left|\tilde{A}_{nmk}^{PQ}\mathsf{n}_k\right|\left(U_m^R - U_m^L\right) \tag{2.3.1.3}$$

where A_{nmk} stands for the flux Jacobian $\frac{\partial f_{nk}^{\text{inv}}}{\partial U_m}$, and n_k is the normal to the finite volume interface between nodes P and Q at time t_{i+1}, with magnitude equal with the area of the interface (see Fig. 2.1). Formally, the normal component should be denoted as $\mathsf{n}_k^{PQ,t_{i+1}}$, but the superscript is omitted for the sake of brevity. $\left|\tilde{A}_{nmk}^{PQ}\mathsf{n}_k\right| = P_{n\ell}\left|\Lambda_{\ell r}\right|P_{rm}^{-1}$ where $\left|\Lambda\right|$ is the diagonal matrix containing the absolute eigenvalues of $A_{nmk}\mathsf{n}_k$ computed using the Roe–averaged quantities between the left (L) and right (R) states. The flow variables at L and R are computed with extrapolation from P and Q using appropriate limiting functions to enforce monotonicity. For more details on these functions the reader is referred to [188, 11, 189]. Note that, on the first term of Eq. 2.3.1.3 instead of the L and R states, the nodal values are used. This, maintains the second–order accuracy of the scheme according to [4] and facilitates the use of a similar scheme for solving the continuous adjoint equations, which will be discussed elsewhere.

The boundary terms, $\Phi_n^{\text{inv},f\in V_P}$ are computed differently for the wall and inlet/outlet boundaries. For the wall boundaries, $f \in \mathcal{B}^{\mathcal{W}}\left(P\right)$,

$$\Phi_n^{\text{inv},f\in V_P} = f_{nk}^{\text{inv},f\in V_P}\mathsf{n}_k \tag{2.3.1.4}$$

where the flux $f_{nk}^{\text{inv},f\in V_P}$ is computed by Eq. 2.1.2 using the flow quantities at node P and taking into account the appropriate slip wall conditions, as described in section 2.2.1. In Eq. 2.3.1.4, n_k are the components of the normal vector on the boundary $\mathcal{f}$, whose magnitude is equal to the area of $\mathcal{f}$ contained in the finite volume formed around node P.

On the other hand, for the inlet/outlet boundaries, namely $\mathcal{f} \in \mathcal{B}^{\mathcal{I}}(P) \cup \mathcal{B}^{\mathcal{O}}(P)$, Eq. 2.3.1.3 is used, where n_k is replaced by $\mathsf{n}_k^{\mathcal{f}\in V_P^{t_{i+1}}}$ and, node Q is a halo (fake) node. The flow variables at node Q are set using the boundary conditions as described in sections 2.2.2 and 2.2.3.

Both Eqs. 2.3.1.3 and 2.3.1.4, contain terms of the form $v_k^R \mathsf{n}_k$ along the finite volume boundaries. These can be computed as $v_k^R \mathsf{n}_k = v_k^A \mathsf{n}_k - v_k^F \mathsf{n}_k$. Term $v_k^F \mathsf{n}_k$ must be computed so as to satisfy the Geometric Conservation Law (GCL) [115]. This ensures, that no unexpected sources or sinks of the conserved quantities are created during the mesh/frame movement. The GCL is stated as

$$\frac{\mathrm{d}}{\mathrm{d}t} \int\limits_{V_P^{t_{i+1}}} \mathrm{d}V = \int\limits_{\partial V_P^{t_{i+1}}} v_k^G \hat{\mathsf{n}}_k \mathrm{d}\left(\partial V\right) \tag{2.3.1.5}$$

where v_k^G is the velocity of the grid. In aeroelastic simulations the finite volume $V_P^{t_{i+1}}$ is computed using a 2nd order backward difference formula (BDF2). When solving the flow equations in a relative frame of reference with a non-deformable grid, one may substitute v_k^F for v_k^G into Eq. 2.3.1.5. Taking into account that, in such a case, the volume of a cell remains constant, Eq. 2.3.1.5 becomes

$$\int\limits_{\partial V_P^{t_{i+1}}} v_k^F \hat{\mathsf{n}}_k \mathrm{d}\left(\partial V\right) \simeq \sum_{\forall Q\in\mathcal{N}(P)} \left(v_k^{F,PQ} \mathsf{n}_k^{PQ}\right)^{t_{i+1}} + \sum_{\forall \mathcal{f}\in\mathcal{B}(P)} \left(v_k^{F,\mathcal{f}\in V_P} \mathsf{n}_k^{\mathcal{f}\in V_P}\right)^{t_{i+1}} = 0 \tag{2.3.1.6}$$

2.3.2 Discretization of the Viscous Terms

Applying the Green-Gauss theorem to the integral of the viscous terms one obtains

$$\int\limits_{V_P^{t_{i+1}}} \frac{\partial f_{nk}^{\text{vis}}}{\partial x_k} \mathrm{d}V = \int\limits_{\partial V_P^{t_{i+1}}} f_{nk}^{\text{vis}} \hat{\mathsf{n}}_k \mathrm{d}\left(\partial V\right) \tag{2.3.2.1}$$

and, by discretization, one obtains

$$\int\limits_{\partial V_P^{t_{i+1}}} f_{nk}^{\text{vis}} \hat{\mathsf{n}}_k \simeq \sum_{\forall Q\in\mathcal{N}(P)} \Phi_n^{\text{vis},PQ} + \sum_{\forall \mathcal{f}\in\mathcal{B}(P)} \Phi_n^{\text{vis},\mathcal{f}\in V_P} \tag{2.3.2.2}$$

where $\Phi_{m+1}^{\text{vis},PQ} = \tau_{mk}^{PQ} \mathsf{n}_k^{PQ,t_{i+1}}, (m = 1, 2, 3)$ is the momentum viscous numerical flux crossing the finite volume interface between nodes P and Q, and $\Phi_5^{\text{vis},PQ} =$

$v_\ell^A\left(\tau_{\ell k}^{PQ}+q_k^{PQ}\right)\mathsf{n}_k^{PQ,t_{i+1}}$ the corresponding energy viscous numerical flux, with τ_{mk}^{PQ} and q_k^{PQ} being the stress tensor and the components of the heat flux, respectively. Computing τ_{mk}^{PQ} and q_k^{PQ} at the finite volume interface between P and Q involves the computation of the velocity and temperature spatial derivatives there. For any quantity ϕ, $\left.\frac{\partial\phi}{\partial x_m}\right|_{PQ}$ is computed using the scheme proposed in [192], that corresponds to a 2nd order central difference scheme involving all neighbours, namely

$$\left.\frac{\partial\phi}{\partial x_m}\right|_{PQ}=\left(\overline{\frac{\partial\phi}{\partial x_m}}\right)_{PQ}-\left[\left(\overline{\frac{\partial\phi}{\partial x_\ell}}\right)_{PQ}\mathsf{t}_\ell-\frac{\phi^Q-\phi^P}{\sqrt{\left(x_\ell^Q-x_\ell^P\right)\left(x_\ell^Q-x_\ell^P\right)}}\right]\mathsf{t}_m \quad (2.3.2.3)$$

where

$$\mathsf{t}_m=\frac{x_m^Q-x_m^P}{\sqrt{\left(x_\ell^Q-x_\ell^P\right)\left(x_\ell^Q-x_\ell^P\right)}} \quad (2.3.2.4)$$

and

$$\left(\overline{\frac{\partial\phi}{\partial x_m}}\right)_{PQ}=\frac{1}{2}\left[\left(\frac{\partial\phi}{\partial x_m}\right)_P+\left(\frac{\partial\phi}{\partial x_m}\right)_Q\right]$$

Along the boundaries, the viscous fluxes are computed as $\Phi_{m+1}^{vis,f\in V_P}=\tau_{mk}^P\mathsf{n}_k^{f\in V_P^{t_{i+1}}}$, $(m=1,2,3)$ for the momentum equations and $\Phi_5^{vis,f\in V_P}=\left[\left(\tau_{\ell k}v_\ell^A\right)^{f\in V_P}+q_k^{f\in V_P}\right]\mathsf{n}_k^{f\in V_P^{t_{i+1}}}$ for the energy equation, by also taking into account the appropriate boundary conditions defined in section 2.2.

2.3.3 Discretization of the Temporal Terms

Applying the Reynolds transport theorem on the integral of the temporal term of Eq. 2.1.1 one obtains

$$\int\limits_{V_P^{t_{i+1}}}\frac{\partial U_n}{\partial t}\mathrm{d}V=\frac{\mathrm{d}}{\mathrm{d}t}\int\limits_{V_P^{t_{i+1}}}U_n\mathrm{d}V-\int\limits_{\partial V_P^{t_{i+1}}}U_nv_k^G\hat{\mathsf{n}}_k\mathrm{d}\left(\partial V\right) \quad (2.3.3.1)$$

The first integral on the right-hand-side (r.h.s.) of Eq. 2.3.3.1 is discretized using a second-order accurate backward difference formula (BDF2), for constant time-step Δt, as

$$\frac{\mathrm{d}}{\mathrm{d}t} \int\limits_{V_P^{t_{i+1}}} U_n \mathrm{d}V \simeq \frac{1}{2\Delta t} \left(3U_n^{P,t_{i+1}} V_P^{t_{i+1}} - 4U_n^{P,t_i} V_P^{t_i} + U_n^{P,t_{i-1}} V_P^{t_{i-1}} \right) \tag{2.3.3.2}$$

The second integral is discretized as

$$\int\limits_{\partial V_P^{t_{i+1}}} U_n v_k^G \hat{\mathsf{n}}_k \mathrm{d}\left(\partial V\right) \simeq \sum_{\forall Q \in \mathcal{N}(P)} \Phi_n^{grid,PQ} + \sum_{\forall f \in \mathcal{B}(P)} \Phi_n^{grid,f \in V_P} \tag{2.3.3.3}$$

with

$$\Phi_n^{grid,PQ} = U_n^{PQ} \left(v_k^G \mathsf{n}_k \right)^{PQ,t_{i+1}} \tag{2.3.3.4}$$

and

$$\Phi_n^{grid,f \in V_P} = U_n^P \left(v_k^G \mathsf{n}_k \right)^{f \in V_P,t_{i+1}} \tag{2.3.3.5}$$

The grid velocities v_k^G are computed so as to satisfy the GCL, Eq. 2.3.1.5. Employing again a BDF2 scheme, this leads to the following constraint

$$\sum_{\forall Q \in \mathcal{N}(P)} \left[v_k^G \mathsf{n}_k \right]^{PQ,t_{i+1}} + \sum_{\forall f \in \mathcal{B}(P)} \left[v_k^G \mathsf{n}_k \right]^{f \in V_P,t_{i+1}} = \frac{1}{2\Delta t} \left(3V_P^{t_{i+1}} - 4V_P^{t_i} + V_P^{t_{i-1}} \right) \tag{2.3.3.6}$$

which must be satisfied by the discretization scheme. Employing the same scheme for the time-integration of the flow quantities as for computing the grid velocities fulfills the GCL. Using a different discretization scheme between these terms may potentially lead to violation of the GCL and harm the time accuracy of the simulation.

2.3.4 Discretization of the Source Terms

The source value on a finite volume S_n^P is assumed to remain constant within the finite volume and the source terms are discretized as

$$\int\limits_{V_P^{t_{i+1}}} S_n \mathrm{d}V \simeq S_n^P V_P^{t_{i+1}} \tag{2.3.4.1}$$

2.3.5 Discretization of the Turbulence Model Terms

Concerning the Spalart–Allmaras turbulence model, for the temporal, diffusion and source terms the same discretization schemes are followed as for the mean flow equations. However, the discretization of the convection term is different. Applying the Green–Gauss theorem to this term leads to the following discretized form

$$\int\limits_{V_P^{t_{i+1}}} \frac{\partial\left(\rho\tilde{\nu}v_k^R\right)}{\partial x_k}\mathrm{d}V = \int\limits_{\partial V_P^{t_{i+1}}} \tilde{\mu}v_k^R\hat{\mathsf{n}}_k\mathrm{d}\left(\partial V\right) = \sum_{\forall Q\in\mathcal{N}(P)} \tilde{\mu}^{PQ}\left(v_k^R\mathsf{n}_k\right)^{PQ,t_{i+1}} + \sum_{\forall f\in\mathcal{B}(P)}\left(\tilde{\mu}v_k^R\right)^P\mathsf{n}_k^{f\in V_P^{t_{i+1}}} \tag{2.3.5.1}$$

where an upwind scheme is used to compute $(\tilde{\mu})^{PQ}$ expressed as

$$\tilde{\mu}^{PQ} = \begin{cases} \tilde{\mu}^P\ , & \text{for } \left(v_k^R\mathsf{n}_k\right)^{PQ,t_{i+1}} > 0 \\ \tilde{\mu}^Q\ , & \text{for } \left(v_k^R\mathsf{n}_k\right)^{PQ,t_{i+1}} < 0 \end{cases} \tag{2.3.5.2}$$

2.4 Numerical Solution of the Discretized Equations

The unsteady residuals of the discretized equations can be expressed as

$$\begin{aligned}\mathcal{R}_n^{\mathrm{GE},P} = &\sum_{\forall Q\in\mathcal{N}(P)}\left(\Phi_n^{\mathrm{GE,conv}} - \Phi_n^{\mathrm{GE,diff}}\right)^{PQ,t_{i+1}} \\ &+ \sum_{\forall f\in\mathcal{B}(P)}\left(\Phi_n^{\mathrm{GE,conv}} - \Phi_n^{\mathrm{GE,diff}}\right)^{f\in V_P,t_{i+1}} \\ &+ S_n^P V_P^{t_{i+1}} + \frac{1}{2\Delta t}\left(3\mathcal{Q}_n^{P,t_{i+1}}V_P^{t_{i+1}} - 4\mathcal{Q}_n^{P,t_i}V_P^{t_i} + \mathcal{Q}_n^{P,t_{i-1}}V_P^{t_{i-1}}\right)\end{aligned} \tag{2.4.1}$$

where $\mathcal{Q}$ stands for either U_n ($n = 1,\ldots,5$) or $\tilde{\mu}$ and GE (Governing Equation) for either MF or SA. As mentioned, Eqs. 2.4.1 are solved using a dual time-stepping technique, with the correction of the flow variables as unknown, namely

$$\frac{V_P}{\Delta\tau_P}\Delta\mathcal{Q}_n^P = -\mathcal{R}_n^{\mathrm{GE},P} \tag{2.4.2}$$

Using a point–implicit scheme and denoting the real-time iteration by i and the pseudo–time one by j, the system of equations reads

$$\left[\frac{V_P}{\Delta\tau_P}\delta_{nm} + \left(\frac{\partial\mathcal{R}_n^{\mathrm{GE},P}}{\partial\mathcal{Q}_m}\right)_{i+1,j}\right]\left(\Delta\mathcal{Q}_m^P\right)_{i+1,j+1} = -\left(\mathcal{R}_n^{\mathrm{GE},P}\right)_{i+1,j} \tag{2.4.3}$$

and by splitting the left-hand-side (l.h.s.) term of these equations in diagonal and off–diagonal terms, these are written as

$$\left(\mathcal{D}_{nm}^{\mathrm{GE},P}\right)_{i+1,j}\left(\Delta\mathcal{Q}_m^P\right)_{i+1,j+1} + \sum_{\forall Q\in\mathcal{N}(P)}\left(\mathcal{Z}_{nm}^{\mathrm{GE},PQ}\right)_{i+1,j}\left(\Delta\mathcal{Q}_m^Q\right)_{i+1,j+1} = -\left(\mathcal{R}_n^{\mathrm{GE},P}\right)_{i+1,j} \tag{2.4.4}$$

where

$$\left(\mathcal{D}_{nm}^{\mathrm{GE},P}\right)_{i+1,j} = \frac{\partial\left(\mathcal{R}_n^{\mathrm{GE},P}\right)_{i+1,j}}{\partial\mathcal{Q}_m^P}, \quad \left(\mathcal{Z}_{nm}^{\mathrm{GE},PQ}\right)_{i+1,j} = \frac{\partial\left(\mathcal{R}_n^{\mathrm{GE},P}\right)_{i+1,j}}{\partial\mathcal{Q}_m^Q}$$

The system of Eq. 2.4.4 is solved using the Jacobi method. The Jacobi method is chosen instead of e.g. the Gauss-Seidel one, because no synchronizations are required in each iteration when the solution of the system is parallelized on the GPU. The flow–chart of the numerical solution process is shown in Fig. 2.2.

2.4.1 Computation of Local Pseudo-Time Step

The time step $\Delta\tau$ used to advance the system of equations in pseudo–time is computed locally for each mesh node. As proposed in [82], this reads

$$\Delta\tau_P = \mathrm{CFL}\frac{V_P}{T^{inv,P} + T^{vis,P}} \tag{2.4.1.1}$$

where

$$\begin{aligned} T^{inv,P} &= \left(\left|v_k^R\right| + c\right)^P S_k^P \\ \text{and}\quad T^{vis,P} &= \frac{2\left(\mu+\mu_t\right)V_P}{\rho^P\left(S_1^P + S_2^P + S_3^P\right)} \\ \text{with}\quad S_k^P &= \frac{1}{2}\sum_{\forall Q\in\mathcal{N}(P)}\left|\mathsf{n}_k^{PQ}\right| \end{aligned}$$

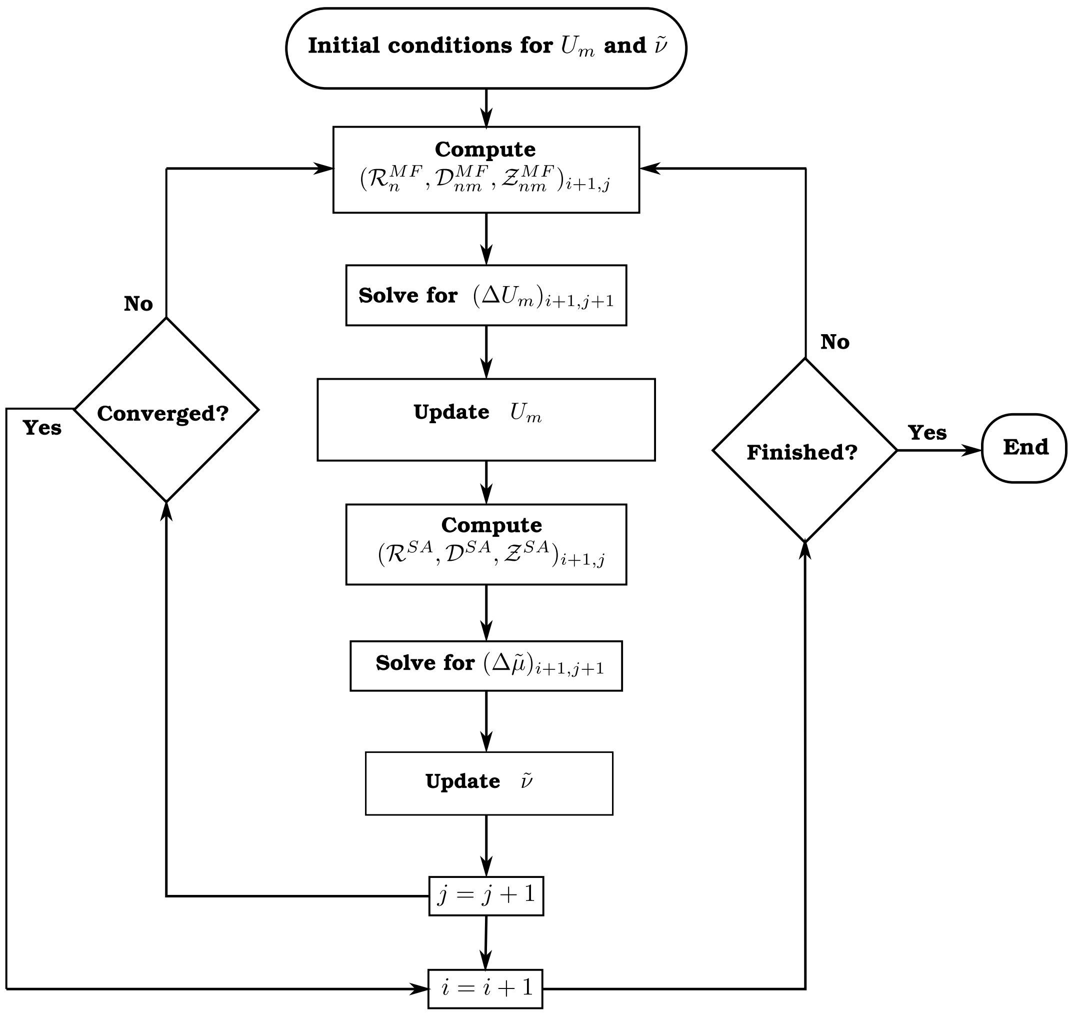

Initial conditions for U_m and $\tilde{\nu}$
Compute $(\mathcal{R}_n^{MF}, \mathcal{D}_{nm}^{MF}, \mathcal{Z}_{nm}^{MF})_{i+1,j}$
Solve for $(\Delta U_m)_{i+1,j+1}$
Update U_m
Compute $(\mathcal{R}^{SA}, \mathcal{D}^{SA}, \mathcal{Z}^{SA})_{i+1,j}$
Solve for $(\Delta\tilde{\mu})_{i+1,j+1}$
Update $\tilde{\nu}$
$j = j + 1$
$i = i + 1$
Converged?
No
Yes
Finished?
No
Yes
End

Figure 2.2: Numerical solution process flow–chart for unsteady flow problems. The real time iteration is i and the pseudo–time one j. In steady flow problems, the outer loop on real time steps is by–passed.

It can be seen that the formula for computing $\Delta\tau$ locally, employs both the maximum (in terms of absolute value) eigenvalue (also referred to as the spectral radii) as well as the local size of the computational mesh. Both measures are known to be closely related to the consistency and the stability of the numerical solution [69].

2.5 The URANS Equations for Incompressible Fluid Flows

This section focuses on the development of the system of equations governing incompressible flows. For incompressible flows, the equation of state, Eq. 2.1.5, no longer holds. The changes in pressure do not cause changes in density and, if, additionally, the flow is assumed to be isothermal, the last of Eqs. 2.1.1, i.e. the energy equation, is identically satisfied. Hence, the density of the fluid (ρ) can be considered constant and, therefore, the kinematic pressure ($\frac{p}{\rho}$) can be used for the pressure p. In what follows, for incompressible flows the symbol p denotes the kinematic pressure. The set of unknown flow variables is now $U_n = \left[p \ \ v_1^A \ \ v_2^A \ \ v_3^A\right]$.

The terms in Eq. 2.1.2 become

$$f_{nk}^{\text{inv}} = \begin{bmatrix} v_k^R \\ v_1^A v_k^R + p\delta_{1k} \\ v_2^A v_k^R + p\delta_{2k} \\ v_3^A v_k^R + p\delta_{3k} \end{bmatrix} \quad f_{nk}^{\text{vis}} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix} S_n = \begin{bmatrix} 0 \\ \varepsilon_{1\ell k}\omega_\ell v_k^A \\ \varepsilon_{2\ell k}\omega_\ell v_k^A \\ \varepsilon_{3\ell k}\omega_\ell v_k^A \end{bmatrix} \tag{2.5.1}$$

The stress tensor is now expressed as $\tau_{km} = \frac{\nu+\nu_t}{\text{Re}_0}\left(\frac{\partial v_k^A}{\partial x_m} + \frac{\partial v_m^A}{\partial x_k}\right)$ since the velocity divergence vanishes due to the continuity equation. In addition, the Mach number is no longer defined since the speed of sound is assumed infinite and p_t is connected to p through $p_t = p + \frac{1}{2}v_\ell^A v_\ell^A$.

The turbulence model equation Eq. 2.1.13 holds and is divided by the constant density, so that the unknown quantity is now $\tilde{\nu}$ instead of $\tilde{\mu}$.

2.6 Boundary Conditions for Incompressible Fluid Flows

The boundary conditions for the symmetry, periodic and wall boundaries are the same as those imposed in case of compressible flows, where the thermal conditions at wall boundaries are redundant due to the isothermal assumption. However, the boundary conditions at inlet and outlet boundaries, although similar, are not identical to the ones used for compressible flows. In the following subsections, the boundary conditions holding along these boundaries for 3D incompressible flow problems are presented.

2.6.1 Inlet Boundary Conditions

For the inlet boundaries, three quantities must be specified and one is extrapolated from the flow domain. The two of the three quantities are the two angles associated with the absolute velocity vector, namely θ_1 and θ_2. The velocity vector is again computed using Eq. 2.2.2.1. The third quantity is either the velocity magnitude $|v_\ell^A|$ or the total pressure. The extrapolated quantity is the static pressure (if the velocity magnitude is imposed), or even the velocity magnitude (if p_t is imposed).

2.6.2 Outlet Boundary Conditions

For the outlet boundaries, one flow quantity must be specified and three are extrapolated from the flow domain. The imposed quantity can be either a fixed value for the static pressure (as a constant along the whole outlet or as radial distribution), a fixed mean value for the static pressure over the whole outlet or the flow rate. For the last two cases, the values of the static pressure or the normal to the outflow boundary velocity, respectively, are computed iteratively by correcting the ones extrapolated from the fluid domain, so as to achieve the prescribed integral quantity value.

In general, care must be taken when specifying inlet and outlet boundary conditions, so that they do not refute each other (e.g. inlet velocity magnitude and outlet flow rate).

2.6.3 The Artificial Compressibility Method

Eqs. 2.1.1 and 2.5.1, although similar to the ones for compressible flows, present a significant difference, when it comes to using the methods developed for the numerical solution of systems of hyperbolic equations. This, is the absence of a temporal term for the pressure in the continuity equation, which makes time-marching techniques not directly applicable to the incompressible equations. Chorin [29] was the first to deal with this problem. He proposed adding an artificial compressibility equation connecting pressure and density, namely

$$\frac{\partial \rho}{\partial p} = \frac{1}{\beta^2} \tag{2.6.3.1}$$

in the continuity equation. Of course, this relation is assumed to hold only in pseudo-time, since applying the same relation in real time would correspond to a violation of the actual conservation laws expressed by Eqs. 2.1.1. By this modification, the system of incompressible flow equations (including also the pseudo-time derivative) is expressed as

$$R_n^{\mathrm{MF}} = \mathcal{M}_{nm}\frac{\partial U_m}{\partial t} + \Gamma_{nm}^{-1}\frac{\partial U_m}{\partial \tau} + \frac{\partial f_{nk}^{\mathrm{inv}}}{\partial x_k} - \frac{\partial f_{nk}^{\mathrm{vis}}}{\partial x_k} + S_n = 0$$

$$\text{with} \quad \Gamma_{nm}^{-1} = \begin{bmatrix} \frac{1}{\beta^2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathcal{M}_{nm} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6.3.2}$$

The parameter β^2 is a constant that corresponds to an artificial speed of sound. This modification changes the mathematical nature of the incompressible flow equations, which are now a system of hyperbolic equations (for moderate to high Reynolds number) with similar properties to the system of compressible flow equations, thus allowing the same numerical techniques to be applied. It must be noted that definitions of Eqs. 2.5.1 still hold.

A more general approach was later introduced by Turkel [187] by using the pseudo-time derivative of $\mathcal{U} = \left[\rho \;\; \rho v_1^A \;\; \rho v_2^A \;\; \rho v_3^A\right]$ and using Eq. 2.6.3.1. This leads to the preconditioned system of equations

$$R_n^{\mathrm{MF}} = \mathcal{M}_{nm}\frac{\partial U_m}{\partial t} + \Gamma_{nm}^{-1}\frac{\partial U_m}{\partial \tau} + \frac{\partial f_{nk}^{\mathrm{inv}}}{\partial x_k} - \frac{\partial f_{nk}^{\mathrm{vis}}}{\partial x_k} + S_n = 0$$

$$\text{with} \quad \Gamma_{nm}^{-1} = \begin{bmatrix} \dfrac{1}{\beta^2} & 0 & 0 & 0 \\ \dfrac{v_1^A + \alpha v_1^R}{\beta^2} & 1 & 0 & 0 \\ \dfrac{v_2^A + \alpha v_2^R}{\beta^2} & 0 & 1 & 0 \\ \dfrac{v_3^A + \alpha v_3^R}{\beta^2} & 0 & 0 & 1 \end{bmatrix} \tag{2.6.3.3}$$

and matrix $\mathcal{M}$ the same as defined in Eq. 2.6.3.2. The parameter β^2 can now vary in space, while α is predefined. The generalized artificial compressibility equations 2.6.3.3 can be viewed as a form of preconditioning, with Γ_{nm}^{-1} being the preconditioning matrix. Hence, different values of the parameter α lead to different preconditioning.

A similar approach can be also employed to the compressible flow equations including the energy equation as well ([43], [7]). For brevity, the thesis refrains from the development of low Mach number preconditioning for compressible flows. For further details on this topic, the reader is referred to [6] and [113].

To ensure a fast convergence rate and at the same time avoid numerical instabilities, β^2 must be selected so as to lead to a stable system for PDEs, with condition number as close as possible to unity. This, requires the minimization of the maximum ratio of the absolute eigenvalues of the preconditioned flux Jacobian. The preconditioned flux Jacobian multiplied by the normal n_k, reads

$$\Gamma_{n\ell}A_{\ell mk}\mathsf{n}_k = A^{\Gamma}_{nmk}\mathsf{n}_k = \begin{bmatrix} 0 & \beta^2\mathsf{n}_1 & \beta^2\mathsf{n}_2 & \beta^2\mathsf{n}_3 \\ \mathsf{n}_1 & v^R_k\mathsf{n}_k - \alpha v^R_1\mathsf{n}_1 & -\alpha v^R_1\mathsf{n}_2 & -\alpha v^R_1\mathsf{n}_3 \\ \mathsf{n}_2 & -\alpha v^R_2\mathsf{n}_1 & v^R_k\mathsf{n}_k - \alpha v^R_2\mathsf{n}_2 & -\alpha v^R_2\mathsf{n}_3 \\ \mathsf{n}_3 & -\alpha v^R_3\mathsf{n}_1 & -\alpha v^R_3\mathsf{n}_2 & v^R_k\mathsf{n}_k - \alpha v^R_3\mathsf{n}_3 \end{bmatrix} \tag{2.6.3.4}$$

The eigenvalues of the matrix with components $A^{\Gamma}_{nmk}\mathsf{n}_k$ are

$$\begin{aligned} \lambda_{1,2} &= v^R_k\mathsf{n}_k \\ \lambda_{3,4} &= \tfrac{1}{2}\left[(1-\alpha)\, v^R_k n_k \pm \sqrt{(1-\alpha)^2\left(v^R_k\mathsf{n}_k\right)^2 + 4\beta^2\mathsf{n}_k\mathsf{n}_k}\right] \end{aligned} \tag{2.6.3.5}$$

and can be proven that $\min\left[\max\left(\frac{|\lambda_i|}{|\lambda_j|}\right)\right]$ is found when

$$\frac{\beta^2\mathsf{n}_m\mathsf{n}_m}{\left(v^R_k\mathsf{n}_k\right)^2} = \begin{cases} 2-\alpha & , \text{ for } \alpha < 1 \\ \alpha & , \text{ for } \alpha \geq 1 \end{cases} \tag{2.6.3.6}$$

Eq. 2.6.3.6 can be used to adjust the value of β^2 locally, based on the flow velocity. The minimum value of β^2 must be safeguarded, since in viscous flows or close to stagnation points it may lead to an infinitely increasing condition number. It is proven that the condition number is always greater than unity (for non-zero velocity) and becomes equal to unity only when $\alpha = 1$. For $\alpha = -1$ the original system Eq. 2.6.3.2 is retrieved.

The incompressible equations are discretized using the same process as for the compressible ones. The only difference lies on the computation of the inviscid terms where the Roe scheme is modified to include the preconditioning matrix in the artificial dissipation part. In this case, Eq. 2.3.1.3 is replaced by

$$\Phi^{inv,PQ}_n = \frac{1}{2}\left(f^{inv,P}_{nk} + f^{inv,Q}_{nk}\right)\mathsf{n}_k - \frac{1}{2}\Gamma^{-1}_{n\ell}\left|\overline{A^{\Gamma}}^{PQ}_{\ell mk}\mathsf{n}_k\right|\left(U^R_m - U^L_m\right) \tag{2.6.3.7}$$

where $\left|\overline{A^{\Gamma}}^{PQ}_{\ell mk}\mathsf{n}_k\right| = P_{\ell n}\,|\Lambda_{nr}|\,P^{-1}_{rm}$ where $|\Lambda|$ is the diagonal matrix containing the absolute eigenvalues of $A_{nmk}\mathsf{n}_k$ computed using the mean-averaged quantities between the left (L) and right (R) states.

2.7 GPU Implementation of the Flow Solver

The different architecture and hardware capabilities of GPUs compared to CPUs raise some issues concerning the discretization and numerical algorithms used for the solution of the flow equations. GPUs are shared memory processors, meaning that all GPU threads which are executed in parallel, access the same RAM. This may lead to thread race conditions which, if not resolved properly, can make the numerical solution process unpredictable. This issue, is more profound in scatter-add algorithms, employed in the computation of the numerical fluxes and the corresponding numerical flux Jacobians. In addition, the amount of cache memory of GPUs (even the latest ones) is still limited compared to CPUs, demanding a more delicate memory handling, so as to minimize the overall memory latency.

In this section, some of the GPU-specific techniques, developed in the scope of the PhD Thesis of X. Trompoukis [182] and, also, used herein, are revisited. The employment of these techniques results in a GPU variant of the flow solver which can be up to 45 times faster compared to the CPU variant of the same flow solver. At this point, it must be noted that the speed-up figures may vary, depending on the actual GPU and CPU software used for the comparison.

2.7.1 Computation of Numerical Fluxes and Flux Jacobians

In a CPU implementation of a flow solver using the vertex-centered variant of the finite volume method, the most efficient method for computing the numerical fluxes and their Jacobians is via an edge-based algorithm. In such an approach, a sweep over all mesh edges is performed, the numerical fluxes and Jacobians are computed at the finite volume interface associated with each edge and, then, these are added to each of the edge's end-nodes, forming r.h.s. (residuals) and l.h.s. (specifically the diagonal part of the Jacobian) terms. By doing so, the flux and Jacobians are computed only once.

The GPU equivalent of such an approach is to associate each GPU thread with a mesh edge, compute the numerical fluxes and Jacobians on this edge and, then, add them to the memory space allocated to each node. However, since the global GPU memory is shared, if two threads, running in parallel, attempt to add the numerical flux simultaneously at the same memory position, the resulting value might be wrong (or, more precisely, the outcome of such an operation is undefined). This can easily be resolved by using atomic operations.

Atomics are operations that block any other thread attempting to modify the memory to which the atomic function operates on, until the data modification by the operation at this specific memory position is completed. Although the use of atomic operations does not explicitly imply a synchronization barrier, threads executed in parallel need to wait for each other when they reach the call to the atomic function, if they need to access the same global memory space. This leads to excessive number of implicit synchronizations among the threads. To overcome this problem, three different techniques have been developed in [182], namely the edge coloring method, the one-kernel and the two-kernel scheme. These techniques are revisited herein.

Computation using Edge Coloring

In the edge coloring technique, mesh edges are colored/grouped so that GPU threads executed simultaneously (i.e. threads belonging to the same warp) do not have to access the same memory positions. The simplest edge coloring technique is to group edges in a way that edges of the same color do not share common mesh nodes. The kernel that computes the r.h.s. $(\mathcal{R})$ and l.h.s. $(\mathcal{D},\ \mathcal{Z})$ terms is called consecutively for each color group. All the kernel calls are associated with the same GPU stream, so as to ensure that threads operating on edges of different groups will not run at the same time.

Historically, this was the first method developed to avoid using atomic operations. However, especially in unstructured meshes, a large number of groups results, thus harming the code's parallel efficiency. The parallel efficiency deteriorates even more when groups with a small number of edges are created. Furthermore, for the vertex-centered approach finite volume approach, the total number of edge groups equals the maximum number of neighbors per node, which neither constant nor known a priori. For this reason, the edge coloring technique is not used in the scope of this thesis.

Computation Using a One-Kernel Scheme

In this technique, each GPU thread is associated with a mesh node. A kernel is launched in which all the numerical fluxes and Jacobians of all mesh edges emanating from the node associated with a certain thread are computed and added to the memory position corresponding to this node, forming the vector $\mathcal{R}$ and the matrices $\mathcal{D}$, $\mathcal{Z}$ of the node. The technique is shown schematically in Fig. 2.3.

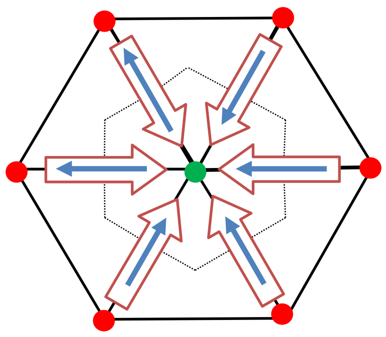


Figure 2.3: The one-kernel technique for the computation of $\mathcal{R}$, $\mathcal{D}$ and $\mathcal{D}$. Each thread associated with a mesh node computes numerical fluxes and Jacobian matrices and adds them directly to the node. Computed fluxes are denoted by thin blue arrows. Thick arrows denote memory write operations.

Using this technique, all problems arising from thread race conditions are resolved at the expense of extra computations. More specifically, the numerical fluxes Φ and their corresponding Jacobian matrices are computed twice for all internal mesh edges. Although this technique involves more computations, it increases the GPU occupancy achieved by the kernel by achieving more efficient memory access and, thus, performs better than the edge coloring technique. In addition, no intermediate data are stored (as is the case with the two-kernel technique to be presented right next) and, hence, there is no additional memory consumption. Since it offers the best compromise between execution time and memory consumption, it is the technique of choice for most scatter-add operations in the flow solver.

Computation using a Two-Kernel Scheme

This technique was proposed in [8] as remedy to the extra computations associated with the one-kernel technique. The scatter-add operation is split into two steps and, consequently, two GPU kernels. Firstly, a kernel is launched associating each GPU thread with a mesh edge. The numerical fluxes and Jacobian matrices at the finite volume interface corresponding to every edge are computed and stored using intermediate memory positions. Then, a second kernel is launched associating every GPU thread with a mesh node. This kernel sweeps all edges emanating from the node and adds flux and Jacobian contributions to $\mathcal{R}$, $\mathcal{D}$ and $\mathcal{Z}$, respectively. The two-kernel technique is described in Fig. 2.4.

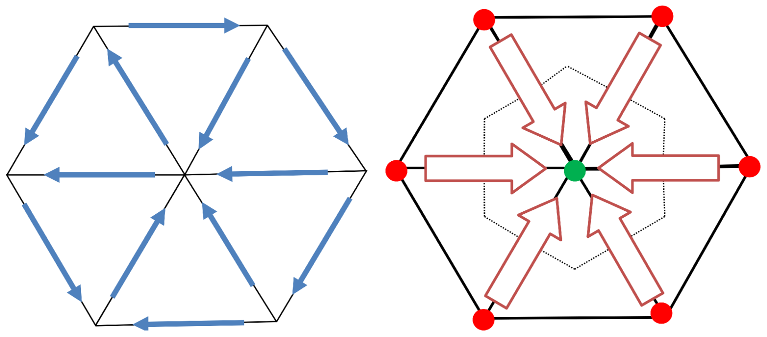


Figure 2.4: The two-kernel technique for the computation of $\mathcal{R}$, $\mathcal{D}$ and $\mathcal{D}$. The 1st kernel associates each thread with a mesh edge and computes the numerical fluxes and Jacobians and stores them on an edge basis. The 2nd kernel associates each thread with a mesh node. It reads the already stored fluxes and Jacobians on the attached edges and adds them to the nodes data. The computed fluxes are denoted by the thin blue arrows. The thick arrows denote the memory write operations [182].

With the two-kernel technique, the fluxes and Jacobian matrices are computed once for each mesh edge, rendering this technique the most efficient in terms of computations and the one providing the higher speed-up. However, the price to pay for the smaller number of arithmetic operations is higher memory requirements. Since GPU memory is usually a limited resource, the two-kernel technique is not used extensively in the flow solver, especially when large 3D meshes are involved. Despite this fact, the two-kernel technique offers a very good alternative for the solution of linear hyperbolic PDEs, where the memory consuming numerical flux Jacobian matrices $\mathcal{D}$, $\mathcal{Z}$ do not change at each pseudo-time iteration and so need not be stored using extra memory. This will become more clear later on, when the adjoint PDEs are introduced.

2.7.2 GPU Memory Handling

Another important aspect affecting the parallel efficiency of the GPU code is the memory handling. GPUs, even the latest ones, offer less cache memory than CPUs and, as a result, the pattern used to store and access data in memory, greatly affects the efficiency of the code. In this section, the patterns used for storing l.h.s. coefficients of Eqs. 2.4.4, are described, since accessing them represents the majority of the memory accesses performed during the numerical solution of the flow equations.

In the CPU implementation of the flow solver, the diagonal terms $\mathcal{D}$ are stored as a list of 5×5 (or 4×4, in the case of incompressible flow) matrices. However, for a GPU implementation this is far from optimal. The reason is that the amount of cached memory is small and, in order to minimize cache miss memory accesses, each memory segment transferred through the bus to each multi-processor must contain as much useful data for the threads of the current warp (executed in parallel) as possible. For this reason, a different pattern is followed. The $(0,0)$ element of matrix $\mathcal{D}$ for the node associated with thread 0 is stored, followed by the $(0,0)$ matrix element of the node associated with thread 1 and so on, up to thread 31, completing the first thread warp (collection of 32 threads which are executed simultaneously). Then, the second element of $\mathcal{D}$ for node of thread 0 follows, and so on. After the whole matrix $\mathcal{D}$ is stored for all the nodes associated with the first warp, storage continues similarly for the next warps. In case the number of nodes on the mesh is not a multiple of 32, a small amount of extra memory is used so as to ensure that even the last warp will follow the same memory access pattern. This method ensures that the memory accesses performed by the threads of a warp are as few as possible and, as a result, the memory bandwidth achieved is close to the hardware nominal value.

The storage in memory of the off-diagonal terms $\mathcal{Z}$ is even more challenging. The number of $\mathcal{Z}$ matrices that need to be stored is equal to two times the number of mesh edges. Consequently, in a serial code, the matrices $\mathcal{Z}$ would be stored as matrices $\mathcal{D}$, one after another, but following the ordering of mesh edges. Of course, using such an approach on a GPU code would not guarantee that the memory accesses needed by the threads of a warp are minimized. To overcome this, the $\mathcal{Z}$ matrices for the first edges emanating from the nodes associated with the warp 0 are first stored, followed by the $\mathcal{Z}$ matrices for the second edges and so on. The storage of each matrix $\mathcal{Z}$ follows the same pattern used for storing matrices $\mathcal{D}$. However, since the number of neighbours per node is not fixed, threads of the same warp may need to access different number of $\mathcal{Z}$ matrices. To eliminate redundant memory accesses, the nodes of the mesh are ordered w.r.t. the number of their neighbours. By doing so, we ensure that the threads of the same warp need to access approximately the same number of $\mathcal{Z}$ matrices. To keep the same memory pattern for all the threads of a warp, even if the number of neighbours differs slightly among them, the memory space allocated for each node is equal to the one needed from the warp thread with the greater number of neighbours.

2.7.3 Mixed Precision Arithmetics

The time spent on accessing memory can be reduced in two ways, namely by improving the memory access patterns (already described in subsection 2.7.2) and by reducing the amount of data that need to be accessed. This is achieved in PUMA by what is called Mixed Precision Arithmetics (MPA) [85].

In MPA, different arithmetics are used to store the l.h.s. and r.h.s. terms of Eqs. 2.4.4. Using a numerical procedure as the one described in section 2.4, the r.h.s. terms ($\mathcal{R}$), contain the residuals of the flow equations represent the physics of the system of equations, while l.h.s. terms ($\mathcal{D}$ and $\mathcal{Z}$ matrices) are linked to the convergence properties of the numerical solution. This means that using inexact values for l.h.s. terms will slightly affect the convergence history of the numerical solution without any impact on the solution accuracy.

For this reason, a mixed precision arithmetics (MPA) technique is proposed. In such an approach, r.h.s. terms are computed and stored using double precision arithmetics (DPA), while single precision arithmetics (SPA) are used to store l.h.s. terms. It must be noted that the flux Jacobians are still computed using double precision operations, but their values are truncated to single precision before these are stored in the global GPU memory.

The MPA could also be used in a CPU code but this would only decrease the memory consumption without any significant effect on its performance, since MPA does not involve any single precision arithmetic operations (only single precision storage). Thus, MPA differs from what is a common practice in several commercial CFD codes, which selectively use SPA (for both arithmetic operations and memory transactions) in order to accelerate the numerical solution. In the case of a GPU code, the advantage of the MPA is twofold. It reduces the number of memory accesses and, consequently, cache-miss operations needed for accessing the l.h.s. terms and, additionally, it reduces memory consumption for storing the demanding l.h.s. terms, which can be crucial in applications involving meshes with a large number of nodes.

2.8 Validation and Verification Test Cases

In this section, a series of validation test cases are presented. They all refer to 2D flow problems widely used for the validation of CFD codes. The code treats 2D flow problems as pseudo–3D ones by using a fake third direction, along which all contributions of numerical fluxes are set to zero. The purpose of these test cases is to check the implementation of the various numerical schemes for correctness and accuracy in comparison with other computational results and experiments. As a consequence, even though these test cases may not represent the current industrial needs, they provide the opportunity of extensively testing the developed software. Validation of the software is also included in the next chapters, in more complex configurations before proceeding to their optimization.

2.8.1 Flow around the RAE2822 Transonic Airfoil

The first test case is an external aerodynamics application and refers to the flow around the RAE2822 airfoil. The freestream flow conditions correspond to a Mach number $\mathrm{M} = 0.725$, a Reynolds number $\mathrm{Re_c} = 6.5 \times 10^6$ based on the chord length and an infinite flow angle $\alpha = 2.92°$.

The computational mesh consists exclusively of quadrilaterals (in practice hexahedra; a one element thick 3D mesh in employed), and is of C-type (Fig. 2.5). Even though the mesh is fully structured, it is treated by PUMA as an unstructured one consisting exclusively of a single element type. The size of the mesh is $\sim 4.8 \times 10^4$ nodes. The Spalart–Allmaras turbulence model is employed. The non–dimensional first wall distance is $y^+ \simeq 1$.

The numerical solution for convergence to machine accuracy takes $\sim 1\,\mathrm{min}$ on a single NVIDIA Tesla K20 GPU. In Fig. 2.6 the Mach number field is presented, where the presence of the shock wave on the suction side is evident. In Fig. 2.7 the pressure coefficient distribution around the airfoil is presented and is also compared with other computational results, as well as experimental measurements. The agreement between computational and experimental results is very satisfactory.

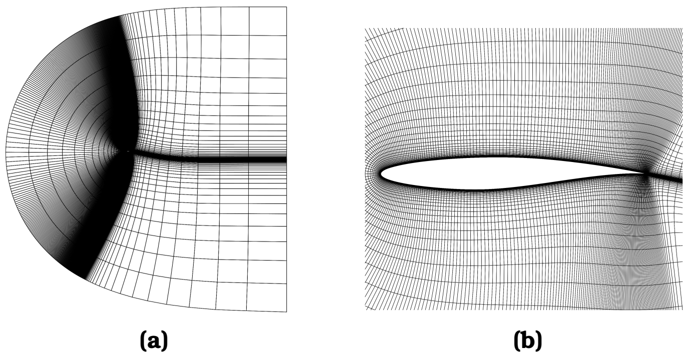


Figure 2.5: *RAE2822 Airfoil*: (a) C–type structured mesh around the airfoil. (b) Mesh view close to the airfoil. Highly stretched cells are used close to the airfoil to provide adequate resolution of the turbulent boundary layer.

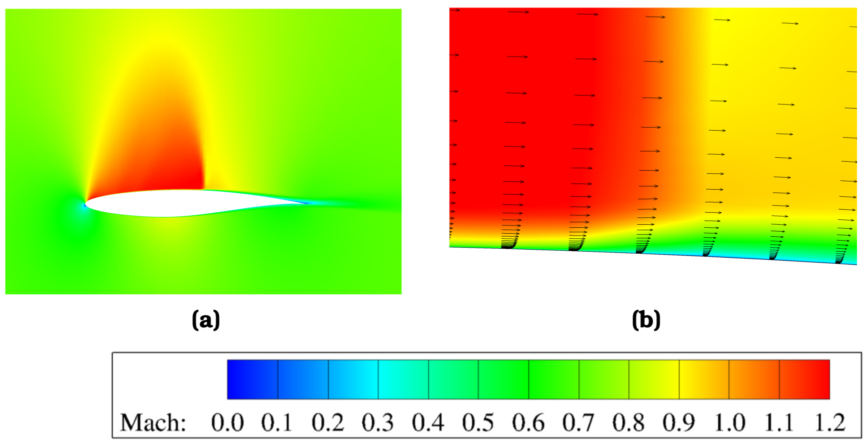


Figure 2.6: *RAE2822 Airfoil*: (a) Mach number field around the airfoil. A shock wave appears over the suction side. (b) View close to the airfoil, at the shock wave position. Velocity vectors are also plotted. The turbulent boundary layer captured by the numerical solution can be seen. The sudden increase of the boundary layer thickness right after the shock wave, a common feature of shock wave/boundary layer interaction, can be seen.

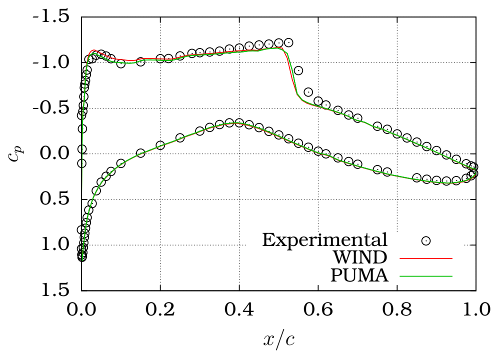


Figure 2.7: *RAE2822 Airfoil*: Pressure coefficient c_p around the airfoil. The abscissa corresponds to normalized chordwise positions (x/c). Comparison with results from the CFD code WIND [58] and experimental results [134]. The position and intensity of the shock wave are properly captured.

2.8.2 Flow through a Transonic Diffuser

The second test case refers to the flow through a transonic diffuser, namely the Sajben test case. The flow conditions are given in Table 2.1. The geometry of the diffuser is presented in Fig. 2.8. Under these flow conditions, a weak shock appears slightly downstream the diffuser's throat.

Quantity	**Symbol**	**Value**
Inlet Total Pressure	p_t^{in}	$134\,999.35\,\text{Pa}$
Inlet Total Temperature	T_t^{in}	$277.78\,\text{K}$
Outlet Static Pressure	p^{out}	$110\,660.85\,\text{Pa}$
Molecular Viscosity (constant)	μ	$1.725 \times 10^{-5}\,\text{kg}\,\text{m}^{-1}\,\text{s}^{-1}$
Inlet Viscosity Ratio	$(\mu_t/\mu)^{\text{in}}$	20.0

Table 2.1: *Sajben Transonic Diffuser*: Flow conditions.

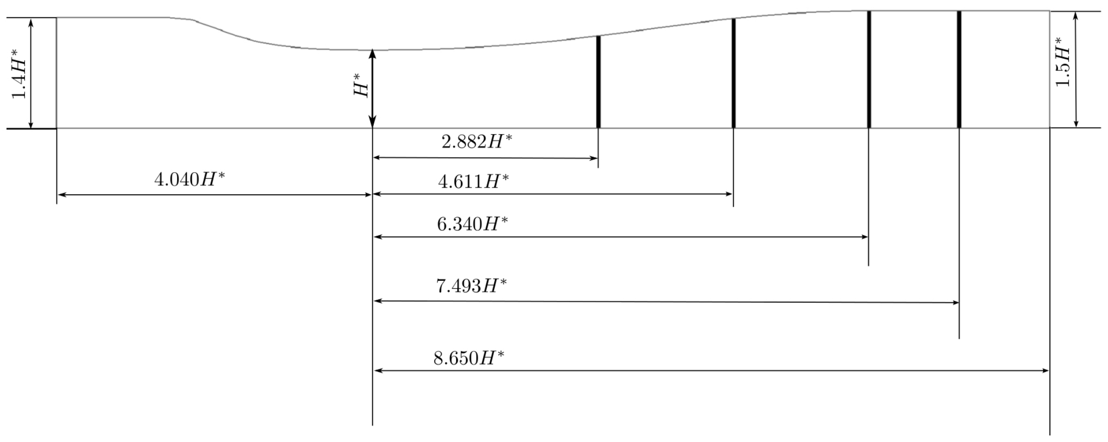


Figure 2.8: *Sajben Transonic Diffuser*: Geometry of the transonic diffuser. All lengths are scaled by the throat diameter (H^*). The positions at which velocity profiles are computed are marked with bold lines.

The mesh consists of $\sim 9 \times 10^3$ nodes and exclusively quadrilateral elements. Again, even though the mesh is structured (H-type) it is treated by PUMA as unstructured. The Spalart-Allmaras turbulence model is used and the turbulent boundary layer is resolved down to the wall. The simulation took $\sim 30\,\text{s}$ on a single NVIDIA Tesla K20 GPU. The pressure distribution along the top and bottom wall of the diffuser is presented in Fig. 2.9. Fig. 2.10 presents the velocity profiles at several positions along the diffuser.

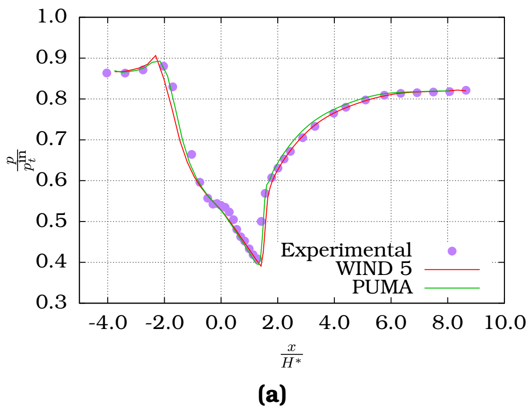


Figure 2.9: *Sajben Transonic Diffuser*: Pressure distribution along (a) the top and (b) bottom wall of the diffuser. The distributions computed are compared to experimental ones [28], [16] and those computed by the CFD code WIND [58].

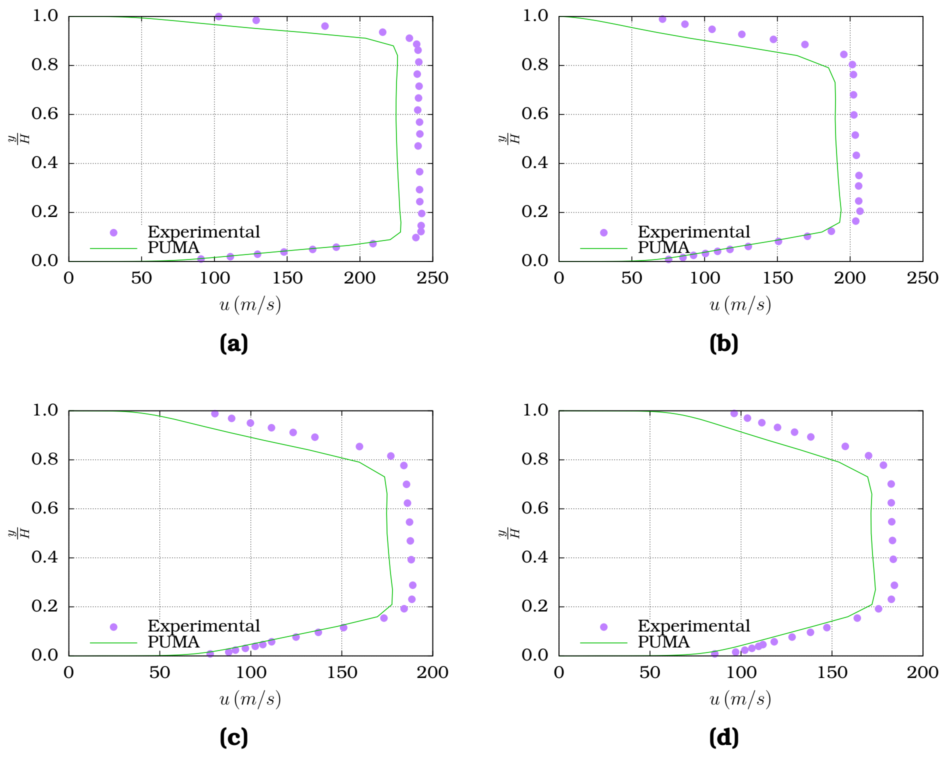


Figure 2.10: *Sajben Transonic Diffuser*: Velocity profiles of the flow through the Sajben transonic diffuser at streamwise positions (a) 2.88, (b) 4.61, (c) 6.34 and (d) 7.49 times the throat diameter, downstream from the throat. Comparison with experimental results at the respective positions.

2.8.3 Flow over a Backward-Facing Step

The third case concerns the flow over a backward-facing step. In this case, an already developed turbulent boundary layer encounters a sudden step-like expansion, which causes immediate flow separation. The flow, then, reattaches downstream. The Reynolds number based on the step height (H) is approximately $\mathrm{Re_H} = 3.6 \times 10^4$. This is a low-speed flow case and, thus, the incompressible flow assumption is valid. Consequently, both the compressible and incompressible flow solvers can be used and validated. In addition, the two variants are compared as far as accuracy of the results and computational cost are of concern.

Quantity	**Symbol**	**Value**
Inlet Total Pressure	p_t^{in}	$101\,325\,\mathrm{Pa}$
Inlet Total Temperature *(only for compressible solver)*	T_t^{in}	$300\,\mathrm{K}$
Outlet Static Pressure	p^{out}	$100\,471.43\,\mathrm{Pa}$
Molecular Viscosity (constant)	μ	$1.846 \times 10^{-5}\,\mathrm{kg\,m^{-1}\,s^{-1}}$
Inlet Viscosity Ratio	$(\mu_t/\mu)^{\mathrm{in}}$	20.0

Table 2.2: *Backward-Facing Step*: Flow conditions.

The flow conditions are summarized in Table 2.2. The computational mesh is multi-block structured consisting of 3 H-type blocks and a total of 6.5×10^5 nodes. Even thought the mesh is multi-block structured, it is, again, treated by PUMA as an unstructured one. The overall computation with the compressible flow solver takes $\sim 20\,\mathrm{min}$ on a single NVIDIA K20 GPU. On the other hand, for the incompressible flow solver the computation on the same hardware and for the same level of convergence takes $\sim 15\,\mathrm{min}$. This was expected since **(a)** the system of equations solved is reduced by one equation (no energy equation for incompressible isothermal flow) and **(b)** the size of the l.h.s. and r.h.s. terms is less and as a result less memory accesses are needed. The increased wall-clock time for this test case, compared to the previous two, is attributed to the significantly larger mesh size which is needed to properly capture the flow separation and reattachment.

The results obtained by the compressible and the incompressible flow solver are compared in Fig. 2.11 along with other computational and experimental results. Also, Fig. 2.12 presents the flow field close to the flow separation and reattachment areas.

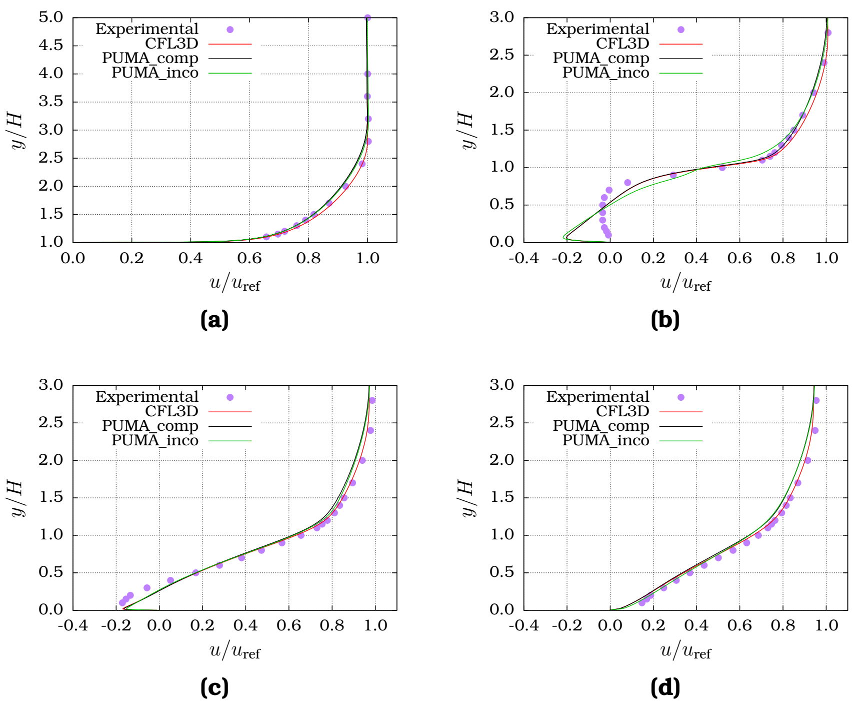


Figure 2.11: *Backward-Facing Step*: Velocity profiles at streamwise positions (x/H) (a) -4, (b) 1, (c) 4 and (d) 6 where H is the step height and $x/H = 0$ corresponds to the step position corresponds to the step position. Comparison with experimental and computational results from the CFD code CFL3D at the same positions. It can be seen that the comparison is good (apart from the velocity profile right after the flow separation) and also both the compressible and incompressible flow solvers compare equally well with the experimental data. However, the incompressible solver has lower computational cost.

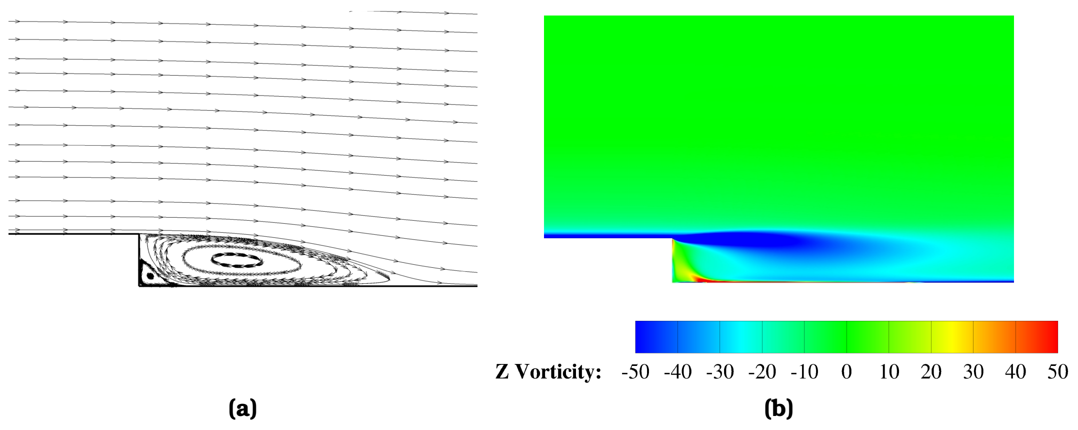


Figure 2.12: *Backward-Facing Step*: (a) Streamlines at the sudden step position. Flow separation and be seen. Flow reattachment is captured right after the recirculation area. (b) Vorticity field at the separation/reattachement area.

2.8.4 Flow around the NACA0012 Isolated Airfoil

This test case is concerned with the 2D flow around the NACA 0012 airfoil. The infinite Mach number is $M = 0.15$ so the incompressible flow solver can be used as well. The Reynolds number based on the airfoil chord is $\mathrm{Re}_c = 6 \times 10^6$. The simulation is performed at different infinite flow angles in the range $\alpha = 0° - 15°$. The computational mesh consists of 4.5×10^5 nodes and is single-block C-type, but is treated by PUMA as unstructured. Both the compressible and incompressible solvers are tested, using the Spalart-Allmaras turbulence model. The compressible solver needs $\sim 10\,\mathrm{min}$ per operating point to reach machine accuracy. The incompressible solver needs $\sim 8\,\mathrm{min}$ per operating point for the same level of convergence. Both solvers run on a single NVIDIA Tesla K40 GPU. The computed values of the lift and drag coefficient at the several infinite flow angles are used to draw the lift and drag polars, which are compared with experimental data and with computational data from a variety of CFD codes (Fig. 2.15). The pressure coefficient distribution, as well as the skin friction coefficient along the suction side of the airfoil are compared with CFD results from the code CFL3D (Fig. 2.14).

The comparison with the experimental and other computational results is very good. Only close to the stall conditions (around $15°$) the lift and drag coefficients measured differ from the computed ones. This is probably due to 3D flow features which appear in the experiment but are not accounted for in the computation. The flow separation close to the trailing edge at $\alpha = 15°$ is shown in Fig. 2.13.

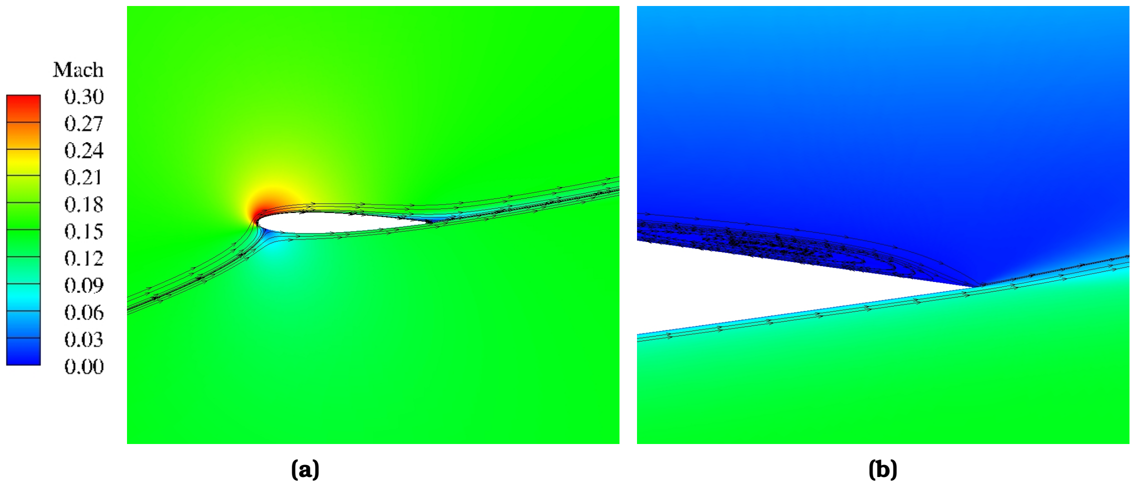
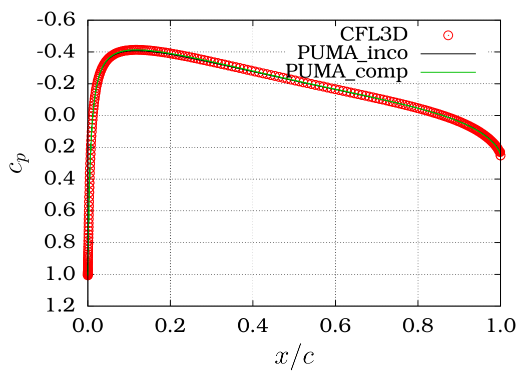
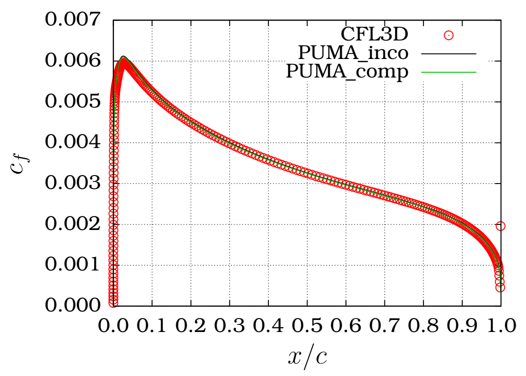


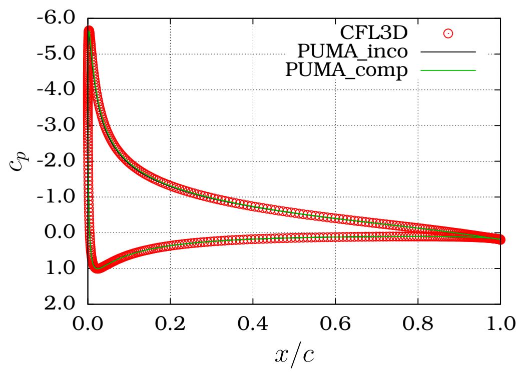
Figure 2.13: *NACA0012 Airfoil*: (a) Mach number field and velocity streamlines for the flow around the airfoil at $15°$ angle of attack. (b) Close-up view at the trailing edge (TE) where flow separation occurs.



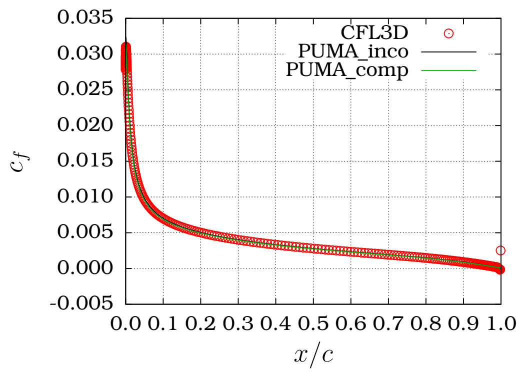
(a) $\alpha = 0°$



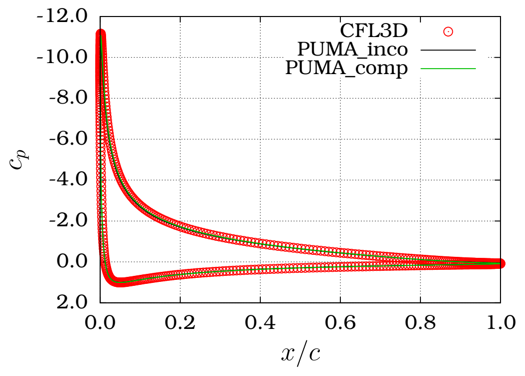
(d) $\alpha = 0°$



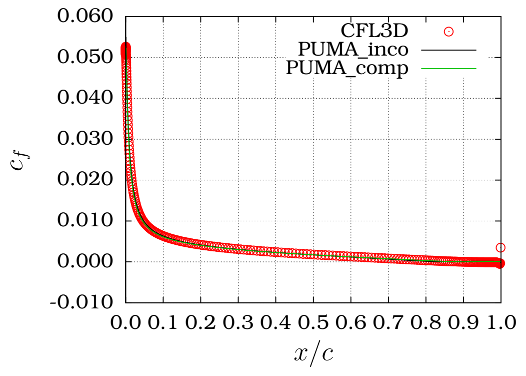
(b) $\alpha = 10°$



(e) $\alpha = 10°$



(c) $\alpha = 15°$



(f) $\alpha = 15°$

Figure 2.14: *NACA0012 Airfoil*: (a)-(c) Comparison of the pressure coefficient distributions around the airfoil contour for different angles of attack. (d)-(f) Comparison of skin friction coefficient distributions along the suction side of the airfoil contour for different angles of attack. All results are compared with the CFD code CFL3D.

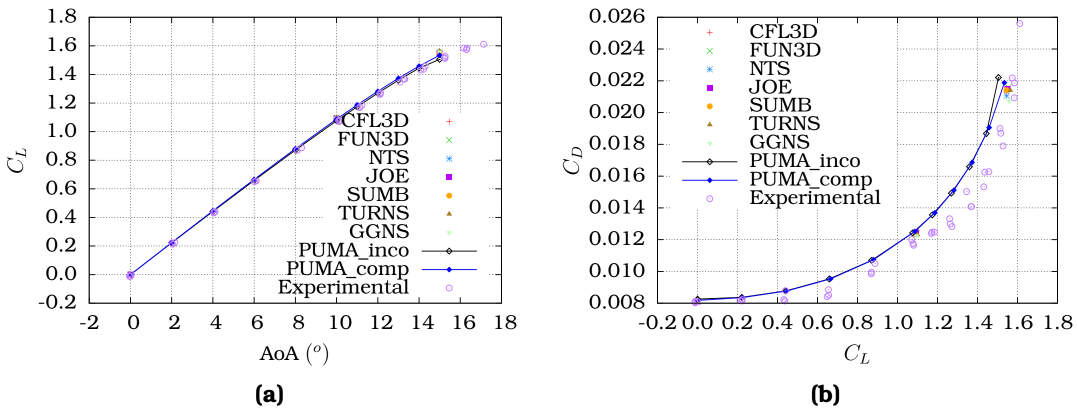


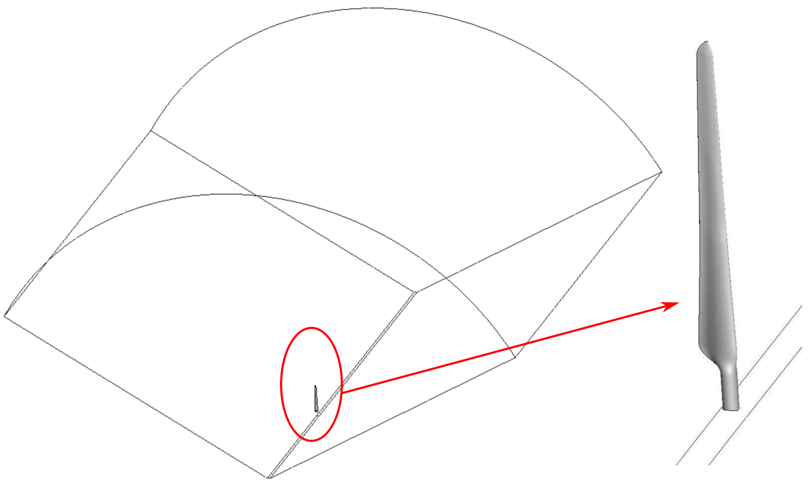
Figure 2.15: *NACA0012 Airfoil*: (a) Comparison of the computed c_L for several angles of attack with other computational and experimental data. (b) Comparison of the computed polar (c_L vs c_D). The comparison with other computational results is excellent. The results compare very well also with the experimental ones. A discrepancy is observed at the higher angles of attack. However, since this appears at near stall conditions the two-dimensionality of the flow as captured by the experiment is questionable.

2.8.5 Flow around a HAWT Blade

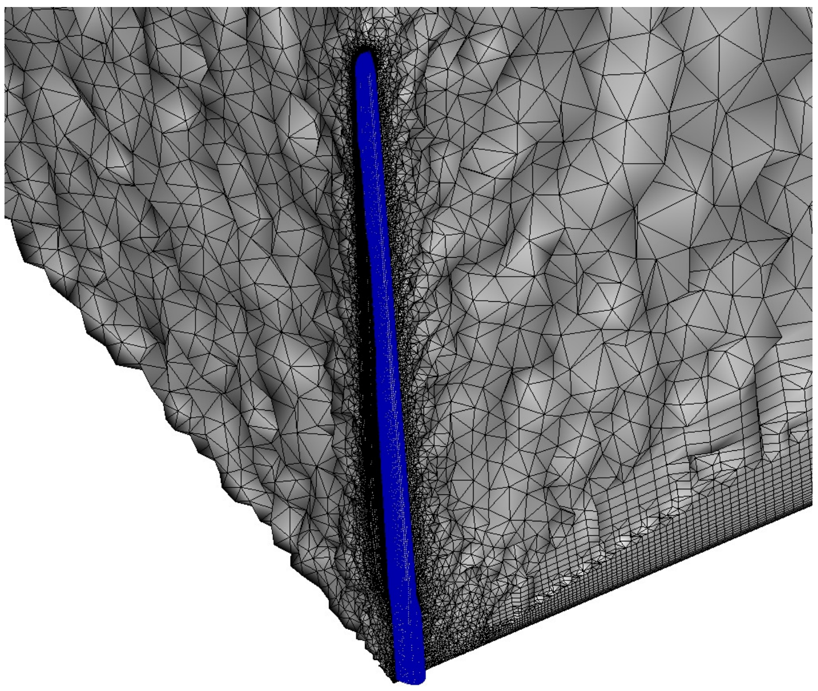
This test case is concerned with the analysis of the flow around a Horizontal Axis Wind Turbine (HAWT) blade, namely that of the MEXICO wind turbine. MEXICO (Model Rotor Experiments In Controlled Conditions) was an EU funded project in which 10 institutes cooperated in experiments carried out on an instrumented 3 bladed wind turbine. The experiments took place in the Large Low-Speed Facility of DNW (German Dutch Wind Tunnels) in the Netherlands [162, 161].

A hybrid mesh of $\sim 2.4 \times 10^6$ nodes and $\sim 4.5 \times 10^6$ elements is built around the blade using the commercial software Pointwise [151]. The CFD domain and the CFD mesh are shown in Fig. 2.16.

The wind speed is $10\,\mathrm{m/s}$ and the wind yaw $0°$. The incompressible flow solver of the PUMA software is used to predict the flow field, running on 4 NVIDIA Tesla K40 GPUs. The overall simulation takes $\sim 70\,\mathrm{min}$. The resulting flow field is presented in Fig. 2.17. The pressure coefficient distribution at different spanwise positions of the blade is shown in Fig. 2.18 and compared with wind tunnel measurements and results CFD results from the code MaPFlow (compressible flow solver) of the Lab. of Aerodynamics, NTUA [135]. The two CFD codes are in full agreement but they do not compare well with the experimental results at sections closer to the blade root. There are many possible explanations for this discrepancy, mainly related to sensitivity of the experimental measurements and turbulence modelling. Nevertheless, the agreement between the two CFD codes



(a)



(b)

Figure 2.16: *MEXICO Wind Turbine Blade*: (a) CFD domain around one blade of MEXICO wind turbine. (b) Hybrid CFD mesh built around the blade.

presented in Fig. 2.18 and other CFD and lifting line codes [160] verify the ability of the PUMA incompressible solver to produce accurate and consistent results.

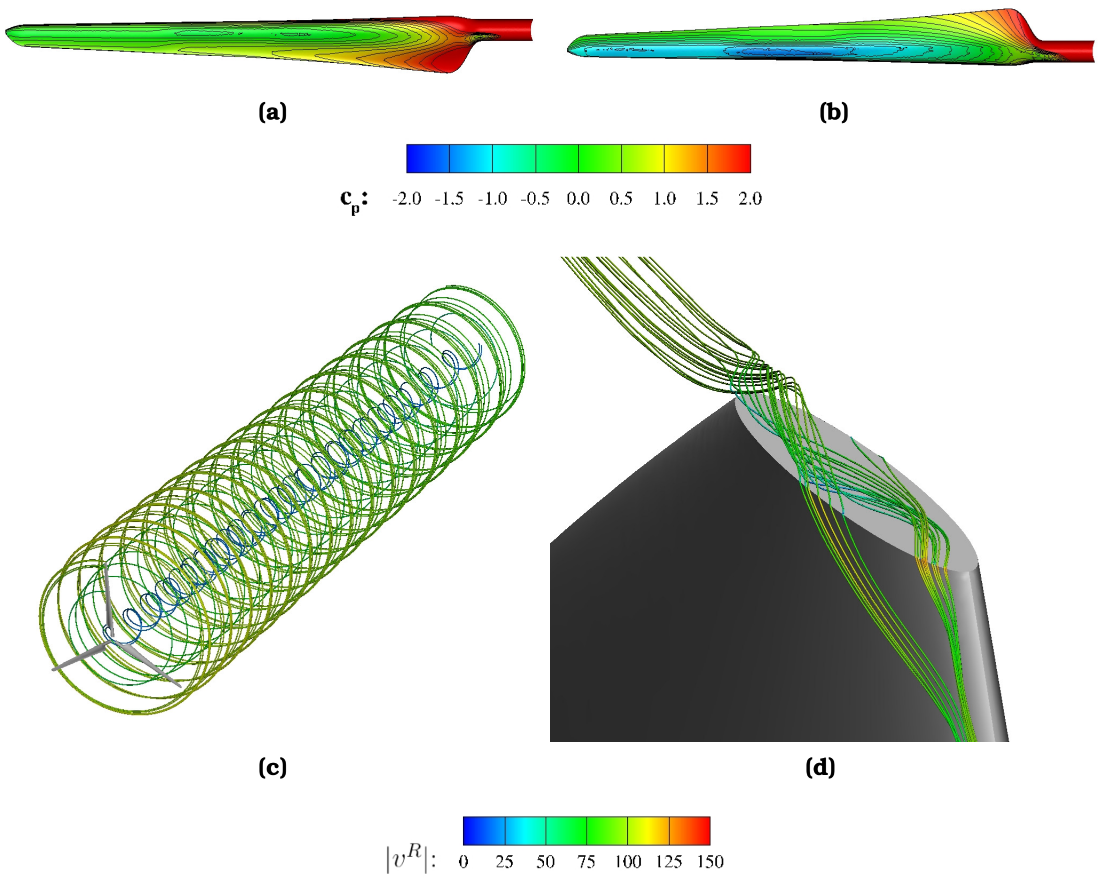
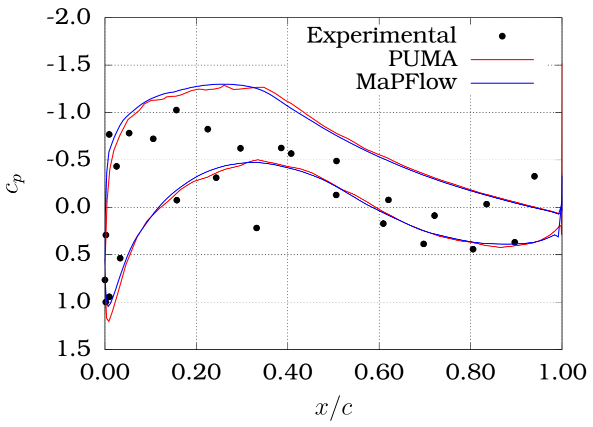
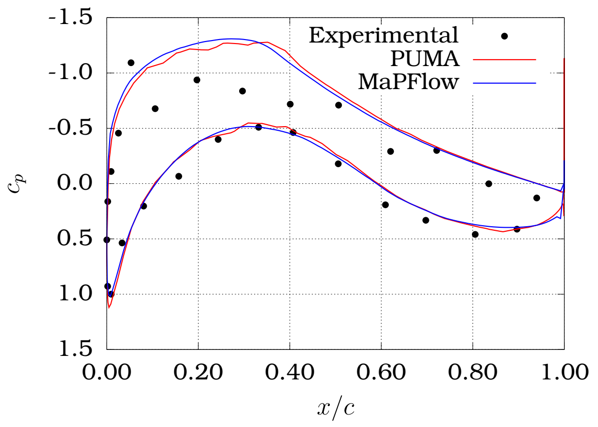

(a)
(b)
c_p: -2.0 -1.5 -1.0 -0.5 0.0 0.5 1.0 1.5 2.0
(c)
(d)
$|v^R|$: 0 25 50 75 100 125 150

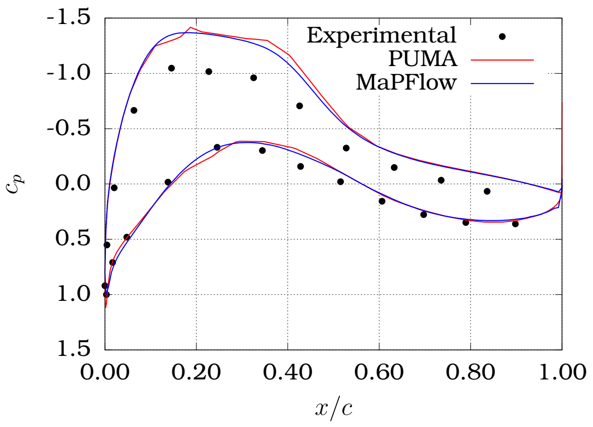
Figure 2.17: *MEXICO Wind Turbine Blade*: Pressure coefficient distribution on the (a) pressure side and (b) of the wind turbine blade. (c) Streamlines computed based on the relative velocity field close to the tip region. (d) Structure of the tip vortex.



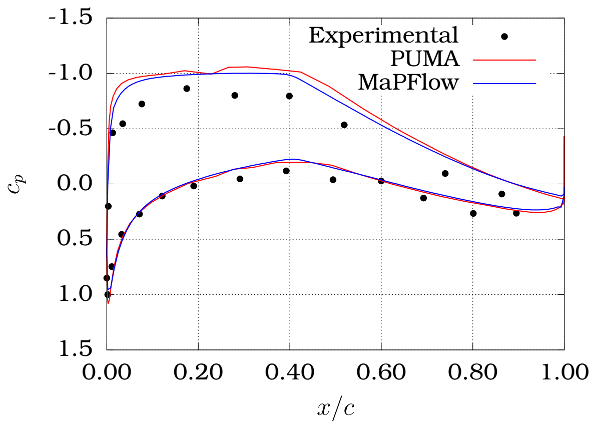
(a) 25% span



(b) 35% span



(c) 60% span



(d) 82% span

Figure 2.18: *MEXICO Wind Turbine Blade*: Comparison between of the pressure coefficient as computed by PUMA and MaPFlow [135] and as measured by experiments [162].

Chapter 3

The Continuous Adjoint Method for Aerodynamic Shape Optimization

This chapter is concerned with the development of the continuous adjoint method for aerodynamic/hydrodynamic shape optimization problems. In gradient-based shape optimization, a main concern is the cost of computing the gradient of an objective function F with respect to a set of design variables $b_i,\ i = 1, \ldots, N$. Hereafter, This gradient $\frac{\delta F}{\delta b_i}$ will be referred to as the sensitivity derivatives, to avoid any confusion with the spatial gradient of flow quantities.

Let F be an integral quantity defined along some surface boundaries of the flow domain S_{Obj} and/or over the fluid volume Ω.

$$F = \int\limits_{\Omega} F_{\Omega} \mathrm{d}\Omega + \int\limits_{S_{\text{Obj}}} F_S \mathrm{d}S \tag{3.1}$$

In what follows, S_{Obj} should not be confused with the whole boundary of the flow domain $\partial\Omega$. Specifically, S_{Obj} is a subset of $\partial\Omega$ along which the objective function F is defined.

Any field quantity Φ can be expressed as a function of the flow variables $\mathcal{Q}_n,\ (n = 1, \ldots, 6$ with $\mathcal{Q}_m \coloneqq U_m$ for $m = 1, \ldots, 5$ and $\mathcal{Q}_6 \coloneqq \tilde{\nu})$ and the position in space $x_k,\ (k = 1, \ldots, 3)$. At this point a distinction must be made between the partial derivative of a field quantity Φ with respect to b_i $\left(\dfrac{\partial \Phi}{\partial b_i}\right)$ and the corresponding total derivative $\left(\dfrac{\delta \Phi}{\delta b_i}\right)$. The partial derivative refers to the change in Φ caused exclusively by changes in the flow variables $\mathcal{Q}_n$ caused by changes in b_i, while the total derivative includes also the change in Φ caused by the change in position x_k of a mesh node. Thus, the partial and total derivative are linked through

$$\frac{\delta\Phi}{\delta b_i}=\frac{\partial\Phi}{\partial b_i}+\frac{\partial\Phi}{\partial x_k}\frac{\delta x_k}{\delta b_i} \tag{3.2}$$

Taking Eq. 3.2 into account and differentiating Eq. 3.1 using also the Leibniz rule, one obtains

$$\begin{aligned}\frac{\delta F}{\delta b_i}=&\int\limits_{\Omega}\frac{\partial F_\Omega}{\partial Q_n}\frac{\partial Q_n}{\partial b_i}d\Omega+\int\limits_{\partial\Omega}F_\Omega\frac{\delta x_k}{\delta b_i}\mathsf{n}_k dS+\int\limits_{S_{\text{Obj}}}\frac{\partial\left(F_S dS\right)}{\partial Q_n}\frac{\delta Q_n}{\delta b_i}\\&+\int\limits_{S_{\text{Obj}}}\frac{\partial\left(F_S dS\right)}{\partial\left(\mathsf{n}_k dS\right)}\frac{\delta\left(\mathsf{n}_k dS\right)}{\delta b_i}+\int\limits_{S_{\text{Obj}}}\frac{\partial\left(F_S dS\right)}{\partial x_k}\frac{\delta x_k}{\delta b_i}\end{aligned} \tag{3.3}$$

The second, fourth and fifth integrals of Eq. 3.3 can be computed by taking the parameterization of the surface under consideration into account. However, the first and third integrals contain variations in the flow variables w.r.t. b_i which are associated with high computational cost. If central finite differences are used, the cost is equal to two equivalent flow solutions (EFS) for each design variable (one around the geometry resulting by altering the design variable by a small number e and one around the geometry resulting by altering the design variable by $-e$.) Even if direct differentiation is employed, the cost is equivalent to one EFS per design variable. The adjoint method, however, makes the cost of computing $\frac{\delta F}{\delta b_i}$ independent of the number of design variables.

Between the two variants of the adjoint method (discrete and continuous), the continuous adjoint method for compressible turbulent flows will be developed in the following sections. The continuous adjoint method will be formulated in two different ways, namely the surface integral (SI) [138, 201], and field integral (FI) [119, 143] formulations. They are proved to be mathematically equivalent but lead to different sensitivity derivative expressions with different accuracy, especially in turbulent flow cases. Upon completion of the continuous adjoint method development for compressible fluid flows, the development of the continuous adjoint method for incompressible flows will be presented based on the FI formulation. The SI formulation for incompressible fluid flows can be found in [102].

3.1 Continuous Adjoint Method - SI Formulation

In order to avoid computing the computationally expensive terms of Eq. 3.3, the augmented objective function F_{aug} is introduced, which is defined as

$$F_{\text{aug}} = F + \int_{\Omega} \Psi_n R_n \mathrm{d}\Omega + \int_{\Omega} \tilde{\nu}_a R_{\tilde{\mu}} \mathrm{d}\Omega, \quad n = 1, \ldots, 5 \tag{3.1.1}$$

In Eq. 3.1.1 , $\Psi_n, (n = 1, \ldots, 5)$ are the mean flow adjoint variables and $\tilde{\nu}_a$ the adjoint turbulence model variable. Both Ψ_n and $\tilde{\nu}_a$ act as Lagrange multipliers since they multiply the equality constraints of the flow equations in the problem of minimizing F_{aug}. Upon convergence of the flow equations (i.e. $R_n = 0$ and $R_{\tilde{\mu}} = 0$) $F_{\text{aug}} = F$ and, consequently the sensitivity derivatives can be computed from $\frac{\delta F_{\text{aug}}}{\delta b_i}$. This is developed as follows,

$$\frac{\delta F_{\text{aug}}}{\delta b_i} = \frac{\delta F}{\delta b_i} + \underbrace{\int_{\Omega} \Psi_n \frac{\partial R_n}{\partial b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF}}} + \underbrace{\int_{\partial\Omega} \Psi_n R_n \frac{\delta x_k}{\delta b_i} \mathsf{n}_k \mathrm{d}S}_{\mathcal{L}^{\text{MF}}} + \underbrace{\int_{\Omega} \tilde{\nu}_a \frac{\partial R_{\tilde{\mu}}}{\partial b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{SA}}} + \underbrace{\int_{\partial\Omega} \tilde{\nu}_a R_{\tilde{\mu}} \frac{\delta x_k}{\delta b_i} \mathsf{n}_k \mathrm{d}S}_{\mathcal{L}^{\text{SA}}} \tag{3.1.2}$$

Terms $\mathcal{L}^{\text{MF}}$ and $\mathcal{L}^{\text{SA}}$ do not contain any variation of flow quantities and, consequently, contribute to the expression of sensitivity derivatives. Term $\frac{\delta F}{\delta b_i}$ can be developed after having defined the objective function F.

During the mathematical development of $\frac{\delta F_{\text{aug}}}{\delta b_i}$, volume integrals containing $\frac{\partial Q_n}{\partial b_i}$ arise. These integrals will be collected to one and the factor multiplying $\frac{\delta Q_n}{\delta b_i}$ will be set equal to zero. By doing so, a new set of PDEs, the so-called field adjoint equations (FAE) arises. Using Eq. 3.2, one may notice that changing $\frac{\delta Q_n}{\delta b_i}$ with $\frac{\partial Q_n}{\partial b_i}$ and eliminating the factors multiplying $\frac{\partial Q_n}{\partial b_i}$ leads to the same FAE. Upon convergence of the FAE, the extra term (arising from the last term of Eq. 3.2) vanishes since $\frac{\partial Q_n}{\partial x_k} \frac{\delta x_k}{\delta b_i}$ is multiplied by the FAE themselves. Satisfaction of the FAE leads to elimination of the volume integrals associated with high computational cost. Similar approach is followed for the surface integrals leading to the introduction of adjoint boundary conditions (ABC). However, for surface integrals the factors multiplying strictly the total derivatives of Q_n must be set to zero.

3.1.1 Differentiation of the Mean Flow Equations

Term $\mathcal{I}^{\text{MF}}$ will be expanded first as

$$\mathcal{I}^{\text{MF}} = \underbrace{\int_{\Omega} \Psi_n \frac{\partial}{\partial b_i} \left(\frac{\partial f^{\text{inv}}_{nk}}{\partial x_k} \right) \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF\_inv}}} \underbrace{- \int_{\Omega} \Psi_n \frac{\partial}{\partial b_i} \left(\frac{\partial f^{\text{vis}}_{nk}}{\partial x_k} \right) \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF\_vis}}} \underbrace{+ \int_{\Omega} \Psi_n \frac{\partial \mathsf{S}_n}{\partial b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF\_src}}} \tag{3.1.1.1}$$

Since the terms $\frac{\partial}{\partial b_i}$ are decoupled from any variation in the physical coordinates x_k, partial and spatial derivatives permute $\left(\text{i.e. } \frac{\partial}{\partial b_i}\left(\frac{\partial}{\partial x_k}\right) = \frac{\partial}{\partial x_k}\left(\frac{\partial}{\partial b_i}\right)\right)$. By doing so and, then, applying the divergence theorem, term $\mathcal{I}^{\text{MF\_inv}}$ becomes

$$\mathcal{I}^{\text{MF\_inv}} = \underbrace{\int\limits_{\partial\Omega} \Psi_n \frac{\partial f^{\text{inv}}_{nk}}{\partial b_i} \mathrm{d}S}_{\mathcal{S}_1} - \underbrace{\int\limits_{\Omega} \frac{\partial \Psi_n}{\partial x_k} A_{nmk} \frac{\partial U_m}{\partial b_i} \mathrm{d}\Omega}_{\text{FAE\_MF}} \tag{3.1.1.2}$$

All volume terms containing $\frac{\partial U_m}{\partial b_i}$ must vanish so as to avoid their expensive computation. So, these volume terms give contribution to the mean flow Field Adjoint Equations (FAE_MF). The surface integral of Eq. 3.1.1.2 must be treated by considering the flow boundary conditions. This is done at a later stage of the formulation, separately for each type of boundary.

Proceeding with the treatment of term $\mathcal{I}^{\text{MF\_vis}}$, this can be split as

$$\begin{aligned}\mathcal{I}^{\text{MF\_vis}} &= \underbrace{-\int\limits_{\partial\Omega} \Psi_n \frac{\partial f^{\text{vis}}_{nk}}{\partial b_i} \mathsf{n}_k \mathrm{d}S}_{\mathcal{S}_2} + \int\limits_{\Omega} \frac{\partial \Psi_n}{\partial x_k} \frac{\partial f^{\text{vis}}_{nk}}{\partial b_i} \mathrm{d}\Omega \\ &= \mathcal{S}_2 + \underbrace{\int\limits_{\Omega} \frac{\partial \Psi_{m+1}}{\partial x_k} \frac{\partial \tau_{km}}{\partial b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MomVis}}} + \underbrace{\int\limits_{\Omega} \frac{\partial \Psi_5}{\partial x_k} \frac{\partial}{\partial b_i} \left(v^A_m \tau_{km}\right) \mathrm{d}\Omega}_{\mathcal{I}^{\text{EnerVis1}}} + \underbrace{\int\limits_{\Omega} \frac{\partial \Psi_5}{\partial x_k} \frac{\partial q_k}{\partial b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{EnerVis2}}}\end{aligned} \tag{3.1.1.3}$$

The first integral ($\mathcal{I}^{\text{MomVis}}$) is expanded using Gauss' theorem and leads to the following terms

$$\begin{aligned}\mathcal{I}^{\text{MomVis}} =& \underbrace{\int\limits_{\partial\Omega} \frac{\mu+\mu_t}{\text{Re}_0} \left(\frac{\partial \Psi_{m+1}}{\partial x_k} + \frac{\partial \Psi_{k+1}}{\partial x_m} - \frac{2}{3}\delta_{km} \frac{\partial \Psi_{\ell+1}}{\partial x_\ell}\right) \mathsf{n}_k \frac{\partial v^A_m}{\partial b_i} \mathrm{d}S}_{\mathcal{S}_3} \\ & \underbrace{- \int\limits_{\Omega} \frac{\partial}{\partial x_k} \left[\frac{\mu+\mu_t}{\text{Re}_0} \left(\frac{\partial \Psi_{m+1}}{\partial x_k} + \frac{\partial \Psi_{k+1}}{\partial x_m} - \frac{2}{3}\delta_{km} \frac{\partial \Psi_{\ell+1}}{\partial x_\ell}\right)\right] \frac{\partial v^A_m}{\partial U_q} \frac{\partial U_q}{\partial b_i} \mathrm{d}\Omega}_{\text{FAE\_MF}} \\ & + \underbrace{\int\limits_{\Omega} \frac{\tau_{km}}{\mu+\mu_t} \frac{\partial \Psi_{m+1}}{\partial x_k} \frac{\partial \mu_t}{\partial \rho} \frac{\partial \rho}{\partial b_i} \mathrm{d}\Omega}_{\text{FAE\_MF}} + \underbrace{\int\limits_{\Omega} \frac{\tau_{km}}{\mu+\mu_t} \frac{\partial \Psi_{m+1}}{\partial x_k} \frac{\partial \mu_t}{\partial \tilde{\nu}} \frac{\partial \tilde{\nu}}{\partial b_i} \mathrm{d}\Omega}_{\text{FAE\_SA}}\end{aligned}$$

$$
+\underbrace{\int_{\Omega}\frac{\tau_{km}}{\mu+\mu_t}\frac{\partial\Psi_{m+1}}{\partial x_k}\left(1+\frac{\partial\mu_t}{\partial\mu}\right)\frac{\partial\mu}{\partial U_q}\frac{\partial U_q}{\partial b_i}\mathrm{d}\Omega}_{Suth} \tag{3.1.1.4}
$$

Here, we can identify surface terms $(\mathcal{S}_2,\mathcal{S}_3)$ and FAE_MF terms. Additionally, terms with variations of the turbulence model variable $\frac{\partial\tilde{\nu}}{\partial b_i}$ arise, whose computation must also be avoided and, so, contribute to the Spalart–Allmaras Field Adjoint Equations (FAE_SA). Finally, the terms denoted as *Suth* are contributions to the FAE_MF, but appear only if the molecular viscosity of the fluid is assumed to obey Eq. 2.1.12. Following the same steps for terms $\mathcal{I}^{\text{EnerVis1}}$ and $\mathcal{I}^{\text{EnerVis2}}$, these lead to

$$
\begin{aligned}
\mathcal{I}^{\text{EnerVis1}} = &\underbrace{\int_{\partial\Omega}\frac{\mu+\mu_t}{\text{Re}_0}\left(\frac{\partial\Psi_5}{\partial x_k}v_m^A+\frac{\partial\Psi_5}{\partial x_m}v_k^A-\frac{2}{3}\delta_{km}\frac{\partial\Psi_5}{\partial x_\ell}v_\ell^A\right)\mathsf{n}_k\frac{\partial v_m^A}{\partial b_i}\mathrm{d}S}_{\mathcal{S}_4}\\
&\underbrace{-\int_{\Omega}\frac{\partial}{\partial x_k}\left[\frac{\mu+\mu_t}{\text{Re}_0}\left(\frac{\partial\Psi_5}{\partial x_k}v_m^A+\frac{\partial\Psi_5}{\partial x_m}v_k^A-\frac{2}{3}\delta_{km}\frac{\partial\Psi_5}{\partial x_\ell}v_\ell^A\right)\right]\frac{\partial v_m^A}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\\
&\underbrace{+\int_{\Omega}\frac{\partial\Psi_5}{\partial x_k}\tau_{km}\left(\frac{\partial v_m^A}{\partial U_n}+\frac{v_m^A}{\mu+\mu_t}\frac{\partial\mu_t}{\partial\rho}\frac{\partial\rho}{\partial U_n}\right)\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\underbrace{+\int_{\Omega}\frac{\partial\Psi_5}{\partial x_k}v_m^A\frac{\tau_{km}}{\mu+\mu_t}\frac{\partial\mu_t}{\partial\tilde{\nu}}\frac{\partial\tilde{\nu}}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}\\
&\underbrace{+\int_{\Omega}\frac{\partial\Psi_5}{\partial x_k}v_m^A\frac{\tau_{km}}{\mu+\mu_t}\left(1+\frac{\partial\mu_t}{\partial\mu}\right)\frac{\partial\mu}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{Suth}
\end{aligned}\tag{3.1.1.5}
$$

$$
\begin{aligned}
\mathcal{I}^{\text{EnerVis2}} = &\underbrace{\int_{\partial\Omega}\frac{\partial\Psi_5}{\partial x_k}\frac{\mathcal{C}_p}{\text{Re}_0}\left(\frac{\mu}{\text{Pr}}+\frac{\mu_t}{\text{Pr}_\text{t}}\right)\mathsf{n}_k\frac{\partial T}{\partial b_i}\mathrm{d}S}_{\mathcal{S}_5}\underbrace{-\int_{\Omega}\frac{\partial}{\partial x_k}\left[\frac{\mathcal{C}_p}{\text{Re}_0}\left(\frac{\mu}{\text{Pr}}+\frac{\mu_t}{\text{Pr}_\text{t}}\right)\frac{\partial\Psi_5}{\partial x_k}\right]\frac{\partial T}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\\
&\underbrace{+\int_{\Omega}\frac{\partial\Psi_5}{\partial x_k}\frac{\mathcal{C}_p}{\text{Re}_0}\frac{\partial T}{\partial x_k}\frac{1}{\text{Pr}_\text{t}}\frac{\partial\mu_t}{\partial\rho}\frac{\partial\rho}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\underbrace{+\int_{\Omega}\frac{\partial\Psi_5}{\partial x_k}\frac{\mathcal{C}_p}{\text{Re}_0}\frac{\partial T}{\partial x_k}\frac{1}{\text{Pr}_\text{t}}\frac{\partial\mu_t}{\partial\tilde{\nu}}\frac{\partial\tilde{\nu}}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}
\end{aligned}
$$

$$\underbrace{+\int\limits_{\Omega}\frac{\partial\Psi_5}{\partial x_k}\frac{C_p}{\mathrm{Re}_0}\frac{\partial T}{\partial x_k}\left(\frac{1}{\mathrm{Pr}}+\frac{1}{\mathrm{Pr_t}}\frac{\partial\mu_t}{\partial\mu}\right)\frac{\partial\mu}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{Suth} \tag{3.1.1.6}$$

Finally, term $\mathcal{I}^{\mathrm{MF\_src}}$, assuming the rotation speed of the frame $\boldsymbol{\omega}$ remains constant, leads to

$$\mathcal{I}^{\mathrm{MF\_src}}=\underbrace{\int\limits_{\Omega}\Psi_{m+1}\rho\varepsilon_{m\ell k}\omega_\ell\frac{\partial v_k^A}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega+\int\limits_{\Omega}\Psi_{m+1}\varepsilon_{m\ell k}\omega_\ell v_k^A\frac{\partial\rho}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_MF}} \tag{3.1.1.7}$$

3.1.2 Differentiation of the Spalart–Allmaras Model Equation

Taking into account Eq. 2.1.13 the term $\mathcal{I}^{\mathrm{SA}}$ is split into three terms for the differentiation of the convection, diffusion and source terms of the Spalart–Allmaras turbulence model PDE.

$$\mathcal{I}^{\mathrm{SA}}=\underbrace{\int\limits_{\Omega}\tilde{\nu}_a\frac{\partial\mathrm{SA}^c}{\partial b_i}\mathrm{d}\Omega}_{\mathcal{I}^{\mathrm{SA\_conv}}}+\underbrace{\int\limits_{\Omega}\tilde{\nu}_a\frac{\partial\mathrm{SA}^d}{\partial b_i}\mathrm{d}\Omega}_{\mathcal{I}^{\mathrm{SA\_diff}}}+\underbrace{\int\limits_{\Omega}\tilde{\nu}_a\frac{\partial\mathrm{SA}^s}{\partial b_i}\mathrm{d}\Omega}_{\mathcal{I}^{\mathrm{SA\_src}}}$$

The convection term reads

$$\begin{aligned}\mathcal{I}^{\mathrm{SA\_conv}}=&\underbrace{\int\limits_{\partial\Omega}\tilde{\nu}_a n_k\frac{\partial\left(\rho\tilde{\nu}v_k^R\right)}{\partial b_i}\mathrm{d}S}_{\mathrm{S}_6}\underbrace{-\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\rho v_k^R\frac{\partial\tilde{\nu}}{\partial b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_SA}}\\&\underbrace{-\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\tilde{\nu}v_k^R\frac{\partial\rho}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_MF}}\underbrace{-\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\rho\tilde{\nu}\frac{\partial v_k^A}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_MF}}\\&+\cancel{\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\rho\tilde{\nu}\frac{\partial v_k^F}{\partial b_i}\mathrm{d}\Omega}\end{aligned} \tag{3.1.2.1}$$

The last term in Eq. 3.1.2.1 vanishes since the frame velocity does not depend on any flow quantity and, thus, its partial derivative w.r.t. b_i must be zero $\left(\frac{\partial v_k^F}{\partial b_i}=0\right)$. Note, however, that since the frame velocity is related to the position of a point in space its total variation w.r.t. b_i is not necessarily equal to zero $\left(\frac{\delta v_k^F}{\delta b_i} \neq 0\right)$.

Term $\mathcal{I}^{\text{SA\_diff}}$ is first expanded as follows

$$\begin{aligned}
\mathcal{I}^{\text{SA\_diff}} = & \underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\tilde{\nu}_a\left\{\frac{\partial}{\partial x_k}\left[(\nu+\tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_k}\right]+c_{b_2}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\right\}\frac{\partial \rho}{\partial U_n}\frac{\partial U_n}{\partial b_i}\text{d}\Omega}_{\text{FAE\_MF}} \\
& \underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\rho\tilde{\nu}_a\frac{\partial}{\partial x_k}\left[\frac{\partial}{\partial b_i}\left\{\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial \tilde{\nu}}{\partial x_k}\right\}\right]\text{d}\Omega}_{\mathcal{I}^{\text{SA\_diff1}}} \\
& \underbrace{+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\rho\tilde{\nu}_a\frac{\partial}{\partial b_i}\left(\tilde{\nu}\frac{\partial^2 \tilde{\nu}}{\partial x_k^2}\right)\text{d}\Omega}_{\mathcal{I}^{\text{SA\_diff2}}}
\end{aligned}$$

After applying the Gauss' theorem, terms $\mathcal{I}^{\text{SA\_diff1}}$ and $\mathcal{I}^{\text{SA\_diff2}}$ result in

$$\begin{aligned}
\mathcal{I}^{\text{SA\_diff1}} = & \underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\rho\tilde{\nu}_a\mathsf{n}_k\frac{\partial}{\partial b_i}\left\{\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial \tilde{\nu}}{\partial x_k}\right\}\text{d}S}_{\mathcal{S}_7} \\
& \underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\frac{\partial(\rho\tilde{\nu}_a)}{\partial x_k}\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\mathsf{n}_k\frac{\partial \tilde{\nu}}{\partial b_i}\text{d}S}_{\mathcal{S}_8} \\
& \underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial(\rho\tilde{\nu}_a)}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}(1+c_{b_2})\frac{\partial \tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}} \\
& \underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial}{\partial x_k}\left\{\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial(\rho\tilde{\nu}_a)}{\partial x_k}\right\}\frac{\partial \tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}} \\
& \underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial(\rho\tilde{\nu}_a)}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\mu}{\rho^2}\frac{\partial \rho}{\partial U_n}\frac{\partial U_n}{\partial b_i}\text{d}\Omega}_{\text{FAE\_MF}}\underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial(\rho\tilde{\nu}_a)}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{1}{\rho}\frac{\partial \mu}{\partial U_n}\frac{\partial U_n}{\partial b_i}\text{d}\Omega}_{Suth}
\end{aligned}$$

$$\mathcal{I}^{\text{SA\_diff2}} = \underbrace{\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\rho\tilde{\nu}_a\tilde{\nu}\mathsf{n}_k\frac{\partial}{\partial b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)\text{d}S}_{\mathcal{S}^9} \underbrace{-\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\frac{\partial\left(\rho\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\mathsf{n}_k\frac{\partial\tilde{\nu}}{\partial b_i}\text{d}S}_{\mathcal{S}^{10}}$$
$$\underbrace{+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\rho\tilde{\nu}_a\frac{\partial^2\tilde{\nu}}{\partial x_k^2}\frac{\partial\tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}} \underbrace{+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial^2\left(\rho\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k^2}\frac{\partial\tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}}$$

Before differentiating the source terms of the Spalart–Allmaras model, the variations of some of the functions defined in Eq. 2.1.14 are calculated. The operator $\mathcal{P}\left(a,c\right)$ denoting the partial derivative of function a w.r.t. quantity c is introduced, by assuming that any other quantity appearing in the expression of a as given in Eq. 2.1.14 is constant. Therefore:

$$\frac{\partial\nu}{\partial b_i} = \mathcal{P}\left(\nu,\rho\right)\frac{\partial\rho}{\partial b_i} + \mathcal{P}\left(\nu,\mu\right)\frac{\partial\mu}{\partial U_n}\frac{\partial U_n}{\partial b_i} \tag{3.1.2.2}$$

$$\frac{\partial\chi}{\partial b_i} = \mathcal{P}\left(\chi,\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial b_i} + \mathcal{P}\left(\chi,\nu\right)\frac{\partial\nu}{\partial b_i} \tag{3.1.2.3}$$

$$\frac{\partial f_{v_1}}{\partial b_i} = \mathcal{P}\left(f_{v_1},\chi\right)\frac{\partial\chi}{\partial b_i} \tag{3.1.2.4}$$

$$\frac{\partial f_{v_2}}{\partial b_i} = \mathcal{P}\left(f_{v_2},\chi\right)\frac{\partial\chi}{\partial b_i} + \mathcal{P}\left(f_{v_2},f_{v_1}\right)\frac{\partial f_{v_1}}{\partial b_i} \tag{3.1.2.5}$$

$$\frac{\partial f_{t_2}}{\partial b_i} = \mathcal{P}\left(f_{t_2},\chi\right)\frac{\partial\chi}{\partial b_i} \tag{3.1.2.6}$$

$$\frac{\partial S}{\partial b_i} = \frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\frac{\partial}{\partial b_i}\left(\frac{\partial v_m^A}{\partial x_\ell}\right) \tag{3.1.2.7}$$

$$\frac{\partial\tilde{S}}{\partial b_i} = \mathcal{P}\left(\tilde{S},S\right)\frac{\partial S}{\partial b_i} + \mathcal{P}\left(\tilde{S},\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial b_i} + \mathcal{P}\left(\tilde{S},f_{v_2}\right)\frac{\partial f_{v_2}}{\partial b_i} + \mathcal{P}\left(\tilde{S},\Delta\right)\frac{\partial\Delta}{\partial b_i} \tag{3.1.2.8}$$

$$\frac{\partial r}{\partial b_i} = \begin{cases} 0 & \text{, for } r > 10 \\ \mathcal{P}\left(r,\tilde{S}\right)\frac{\partial\tilde{S}}{\partial b_i} + \mathcal{P}\left(r,\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial b_i} + \mathcal{P}\left(r,\Delta\right)\frac{\partial\Delta}{\partial b_i} & \text{, else} \end{cases} \tag{3.1.2.9}$$

$$\frac{\partial g}{\partial b_i} = \mathcal{P}\left(g,r\right)\frac{\partial r}{\partial b_i} \tag{3.1.2.10}$$

$$\frac{\partial f_w}{\partial b_i} = \mathcal{P}\left(f_w,g\right)\frac{\partial g}{\partial b_i} \tag{3.1.2.11}$$

Taking into account the Eqs. 3.1.2.2-3.1.2.11 the differentiation of the source terms leads to

$$
\begin{aligned}
\mathcal{I}^{\text{SA\_src}} = & \underbrace{\int_\Omega \tilde{\nu}_a \left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{S}\tilde{\nu} + \frac{1}{\text{Re}_0}\left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\Delta}\right)^2\right]\frac{\partial \rho}{\partial b_i}\text{d}\Omega}_{\text{FAE\_MF}} \\
& + \underbrace{\int_\Omega \rho\tilde{\nu}_a \left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{S} + \frac{2}{\text{Re}_0}\left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\Delta^2}\right)\right]\frac{\partial \tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}} \\
& + \underbrace{\int_\Omega \rho\tilde{\nu}_a \frac{c_{w_1}}{\text{Re}_0}\mathcal{C}_4\frac{\partial \tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}} \underbrace{- \frac{2}{\text{Re}_0}\int_\Omega \rho\tilde{\nu}_a\left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\frac{\tilde{\nu}^2}{\Delta^3}\frac{\partial \Delta}{\partial b_i}\text{d}\Omega + \frac{c_{w_1}}{\text{Re}_0}\int_\Omega \rho\tilde{\nu}_a\mathcal{C}_5\frac{\partial \Delta}{\partial b_i}\text{d}\Omega}_{\text{FAE\_DISTANCE}} \\
& + \underbrace{\int_\Omega \rho\tilde{\nu}_a\left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{\nu} + \frac{c_{w_1}}{\text{Re}_0}\mathcal{C}_3\right]\frac{\partial \tilde{S}}{\partial b_i}\text{d}\Omega}_{\mathcal{I}^{\tilde{S}}} + \underbrace{\int_\Omega \rho\tilde{\nu}_a\left[c_{b_1}\tilde{S}\tilde{\nu} - \frac{c_{b_1}}{\text{Re}_0\,\kappa^2}\left(\frac{\tilde{\nu}}{\Delta}\right)^2\right]\frac{\partial f_{t_2}}{\partial b_i}\text{d}\Omega}_{\mathcal{I}^{f_{t_2}}}
\end{aligned}
\tag{3.1.2.12}
$$

where

$$
\begin{aligned}
&\mathcal{C}_1 = \mathcal{P}\left(\tilde{S}, f_{v_2}\right)\left[\mathcal{P}\left(f_{v_2},\chi\right) + \mathcal{P}\left(f_{v_2},f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\right] \\
&\mathcal{C}_2 = \begin{cases} 0 & \text{, for } r > 10 \\ \mathcal{P}\left(f_w, g\right)\mathcal{P}\left(g, r\right)\left(\frac{\tilde{\nu}}{\Delta}\right)^2 & \text{, else} \end{cases} \\
&\mathcal{C}_3 = \mathcal{C}_2\mathcal{P}\left(r,\tilde{S}\right), \quad \mathcal{C}_4 = \mathcal{C}_2\mathcal{P}\left(r,\tilde{\nu}\right), \quad \mathcal{C}_5 = \mathcal{C}_2\mathcal{P}\left(r,\Delta\right)
\end{aligned}
\tag{3.1.2.13}
$$

Term $\mathcal{I}^{\tilde{S}}$ is further expanded as

$$
\begin{aligned}
\mathcal{I}^{\tilde{S}} = & \underbrace{\int_\Omega \rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{\partial S}{\partial b_i}\text{d}\Omega}_{\mathcal{I}^{\text{VORTICITY}}} + \underbrace{\int_\Omega \rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},\tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}} + \underbrace{\int_\Omega \rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},\Delta\right)\frac{\partial \Delta}{\partial b_i}\text{d}\Omega}_{\text{FAE\_DISTANCE}} \\
& + \underbrace{\int_\Omega \rho\tilde{\nu}_a\mathcal{C}_6\mathcal{C}_1\mathcal{P}\left(\chi,\tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial b_i}\text{d}\Omega}_{\text{FAE\_SA}} + \underbrace{\int_\Omega \rho\tilde{\nu}_a\mathcal{C}_6\mathcal{C}_1\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\rho\right)\frac{\partial \rho}{\partial b_i}\text{d}\Omega}_{\text{FAE\_MF}}
\end{aligned}
$$

$$+\underbrace{\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{C}_1\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\mu\right)\frac{\partial\mu}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{Suth} \qquad (3.1.2.14)$$

where, $\mathcal{C}_6=\left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{\nu}+\frac{c_{w_1}}{\mathrm{Re}_0}\mathcal{C}_3\right]$

Taking Eq. 3.1.2.7 into account and applying the Gauss' theorem, term $\mathcal{I}^{\text{VORTICITY}}$ is expanded as follows

$$\mathcal{I}^{\text{VORTICITY}}=\underbrace{\int\limits_{\partial\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_m^A}{\partial x_\ell}\mathsf{n}_q\frac{\partial v_r^A}{\partial b_i}\mathrm{d}S}_{\mathcal{S}^{11}}$$
$$-\underbrace{\int\limits_{\Omega}\frac{\partial}{\partial x_q}\left[\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_m^A}{\partial x_\ell}\right]\frac{\partial v_r^A}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}} \qquad (3.1.2.15)$$

Finally, term $\mathcal{I}^{f_{t_2}}$ becomes

$$\mathcal{I}^{f_{t_2}}=\underbrace{\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_7\mathcal{P}\left(\chi,\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}+\underbrace{\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_7\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\rho\right)\frac{\partial\rho}{\partial b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}$$
$$+\underbrace{\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_7\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\mu\right)\frac{\partial\mu}{\partial U_n}\frac{\partial U_n}{\partial b_i}\mathrm{d}\Omega}_{Suth} \qquad (3.1.2.16)$$

with $\mathcal{C}_7=\left\{c_{b_1}\left[\tilde{S}\tilde{\nu}-\frac{1}{\mathrm{Re}_0\,\kappa^2}\left(\frac{\tilde{\nu}}{\Delta}\right)^2\right]\right\}\mathcal{P}\left(f_{t_2},\chi\right)$

3.1.3 Field Adjoint Equations

Having differentiated all the terms resulting from the mean flow and the Spalart-Allmaras PDEs, all terms denoted as FAE_MF may vanish by setting them to zero, which gives rise to the mean flow field adjoint PDEs. Similarly, setting the terms denoted as FAE_SA equal to zero gives rise to the adjoint SA PDE. The terms denoted as *Suth* are also grouped with the FAE_MF terms if Sutherland's law is used to account for variations in the fluid's dynamic viscosity. The

FAE_DISTANCE terms (i.e. the ones containing variations in distance) are either computed directly by the parameterization and mesh morphing technique (assuming that the distance of a particular node from the wall is always measured from the same closest to this node point on the wall) or included into the adjoint distance equation (by differentiating the Eikonal equation for the distance) [144].

The mean flow field adjoint equations read

$$-A_{nmk}\frac{\partial \Psi_n}{\partial x_k} - \mathcal{K}_m + \mathcal{K}_m^{\text{SA}} + S_m^{\text{adj}} + \mathcal{B}_m + \mathcal{B}_m^{\text{SA}} + \frac{\partial F_\Omega}{\partial U_m} = 0 \tag{3.1.3.1}$$

where the terms $\mathcal{K}_m$ and $\mathcal{K}_m^{\text{SA}}$ result from the differentiation of the mean-flow viscous terms and the differentiation of the turbulence model, namely

$$\mathcal{K}_m = \frac{\partial \tau_{kq}^{\text{adj}}}{\partial x_k}\frac{\partial v_q^A}{\partial U_m} + \frac{\partial q_k^{\text{adj}}}{\partial x_k}\frac{\partial T}{\partial U_m} - \tau_{kq}\frac{\partial \Psi_5}{\partial x_k}\frac{\partial v_q^A}{\partial U_m} \tag{3.1.3.2}$$

$$\mathcal{K}_m^{\text{SA}} = \mathcal{K}_m^{\text{SA},\rho}\frac{\partial \rho}{\partial U_m} + \mathcal{K}_m^{\text{SA},v_k^A}\frac{\partial v_k^A}{\partial U_m} \tag{3.1.3.3}$$

with

$$\begin{aligned}
\tau_{kq}^{\text{adj}} &= \frac{\mu+\mu_t}{\text{Re}_0}\left(\frac{\partial \Psi_{k+1}}{\partial x_q} + \frac{\partial \Psi_{q+1}}{\partial x_k} - \frac{2}{3}\delta_{kq}\frac{\partial \Psi_{\ell+1}}{\partial x_\ell} + \frac{\partial \Psi_5}{\partial x_k}v_q^A + \frac{\partial \Psi_5}{\partial x_q}v_k^A - \frac{2}{3}\delta_{kq}\frac{\partial \Psi_5}{\partial x_\ell}v_\ell^A\right)\\
q_k^{\text{adj}} &= \frac{\mathcal{C}_p}{\text{Re}_0}\left(\frac{\mu}{\text{Pr}} + \frac{\mu_t}{\text{Pr}_\text{t}}\right)\frac{\partial \Psi_5}{\partial x_k}\\
\mathcal{K}_m^{\text{SA},\rho} &= \frac{\tau_{kq}}{\mu+\mu_t}\left[\frac{\partial \Psi_{q+1}}{\partial x_k} + \frac{\partial \Psi_5}{\partial x_k}v_q^A + \frac{\mathcal{C}_p}{\text{Re}_0\,\text{Pr}_\text{t}}\frac{\partial \Psi_5}{\partial x_k}\frac{\partial T}{\partial x_k}\right]\tilde{\nu} f_{v_1}\\
&\quad - \frac{\partial \tilde{\nu}_a}{\partial x_k}\tilde{\nu} v_k^R - \frac{1}{\text{Re}_0\,\sigma}\tilde{\nu}_a\left\{\frac{\partial}{\partial x_k}\left[(\nu+\tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_k}\right] + c_{b_2}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\right\}\\
&\quad - \frac{1}{\text{Re}_0\,\sigma}\frac{\partial}{\partial x_k}\left(\rho\tilde{\nu}_a\right)\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\mu}{\rho^2}\\
&\quad - \tilde{\nu}_a\left[c_{b_1}\left(1-f_{t_2}\right)\tilde{S}\tilde{\nu} - \frac{1}{\text{Re}_0}\left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\Delta}\right)^2\right]\\
&\quad + \rho\tilde{\nu}_a\left(\mathcal{C}_6\mathcal{C}_1 + \mathcal{C}_7\right)\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\rho\right)\\
\mathcal{K}_m^{\text{SA},v_k^A} &= -\rho\tilde{\nu}\frac{\partial \tilde{\nu}_a}{\partial x_k} - \frac{\partial}{\partial x_q}\left[\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{r\ell n}\varepsilon_{rqk}\frac{\partial v_n^A}{\partial x_k}\right]
\end{aligned} \tag{3.1.3.4}$$

Term S_m^{adj} stands for the adjoint to the Coriolis force expressed as $S_m^{\text{adj}} = \rho\varepsilon_{nk\ell}\Psi_{k+1}\omega_\ell\frac{\partial v_n^A}{\partial U_m} + \varepsilon_{nk\ell}\Psi_{k+1}\omega_\ell v_n^A\frac{\partial \rho}{\partial U_m}$ and, finally terms $\mathcal{B}_m$ and $\mathcal{B}_m^{\text{SA}}$ are present only when Sutherland's law is differentiated. Term $\frac{\partial F_\Omega}{\partial U_m}$ arises from the differentiation of the objective function only if it is defined as a volume integral and is non-zero only at the region of the flow domain that the objective function is defined. Objective functions including exclusively surface integrals do not affect the FAE.

Similarly, the field adjoint equation for the Spalart–Allmaras model reads

$$-\frac{\partial}{\partial x_k}\left(\rho v_k^R \tilde{\nu}_a\right) - \mathcal{D}^{\text{SA,adj}} + \mathcal{G}^{\text{SA,diff}} + \mathcal{G}^{\text{SA,src}} + \mathcal{G}^{\mu_t}\frac{\partial \mu_t}{\partial \tilde{\nu}} + \frac{\partial F_\Omega}{\partial \tilde{\nu}} = 0 \qquad (3.1.3.5)$$

where,

$$\begin{aligned}
\mathcal{D}^{\text{SA,adj}} &= \frac{1}{\text{Re}_0\,\sigma}\frac{\partial}{\partial x_k}\left\{\left[\nu + (1+c_{b_2})\right]\frac{\partial}{\partial x_k}\left(\rho\tilde{\nu}_a\right)\right\} - \frac{c_{b_2}}{\text{Re}_0\,\sigma}\frac{\partial^2}{\partial x_k^2}\left(\rho\tilde{\nu}_a\tilde{\nu}\right)\\
\mathcal{G}^{\text{SA},diff} &= \frac{1}{\text{Re}_0\,\sigma}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\left(1+c_{b_2}\right)\frac{\partial\tilde{\nu}}{\partial x_k} + \frac{c_{b_2}}{\text{Re}_0\,\sigma}\rho\tilde{\nu}_a\frac{\partial^2\tilde{\nu}}{\partial x_k^2}\\
\mathcal{G}^{\text{SA},src} &= \rho\tilde{\nu}_a\Bigg[-c_{b_1}\left(1-f_{t_2}\right)\tilde{S} + \frac{2}{\text{Re}_0}\left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\Delta^2}\right)\\
&\quad + \frac{c_{w_1}}{\text{Re}_0}\mathcal{C}_4 + \mathcal{C}_6\mathcal{P}\left(\tilde{S},\tilde{\nu}\right) + \mathcal{C}_6\mathcal{C}_1\mathcal{P}\left(\chi,\tilde{\nu}\right) + \mathcal{C}_7\mathcal{P}\left(\chi,\tilde{\nu}\right)\Bigg]\\
\mathcal{G}^{\mu_t} &= \frac{\tau_{km}}{\mu+\mu_t}\left(\frac{\partial\Psi_{m+1}}{\partial x_k} + \frac{\partial\Psi_5}{\partial x_k}v_m^A\right) + \frac{\mathcal{C}_p}{\text{Re}_0\,\text{Pr}_\text{t}}\frac{\partial\Psi_5}{\partial x_k}\frac{\partial T}{\partial x_k}
\end{aligned} \qquad (3.1.3.6)$$

Since Eqs. 3.1.3.1 and 3.1.3.5 are a set of linear PDEs, a numerical solution procedure similar to the one described in Section 2.4 can be followed with l.h.s.terms of the system of discretized equations computed a priori and only the r.h.s.recomputed during the numerical solution. Since the r.h.s.consumes much less memory, the two-kernel scheme described in Section 2.7.1 can be employed. As a result, a single pseudo-time iteration for the solution of the adjoint equations in PUMA, is faster compared to one iteration for the solution of the flow equations.

3.1.4 Adjoint Boundary Conditions

After having treated all volume integrals arising from the differentiation of F_{aug}, the surface integrals remain. These are summarized below

$$
\begin{aligned}
\mathcal{S}_1 &= \int\limits_{\partial\Omega} \Psi_n A_{nmk} \mathsf{n}_k \frac{\partial U_m}{\partial b_i} \mathrm{d}S \\
\mathcal{S}_2 &= -\int\limits_{\partial\Omega} \Psi_n \frac{\partial f^{\text{vis}}_{nk}}{\partial b_i} \mathsf{n}_k \mathrm{d}S \\
\mathcal{S}_{3,4} &= \int\limits_{\partial\Omega} \tau^{\text{adj}}_{km} \mathsf{n}_k \frac{\partial v^A_m}{\partial b_i} \mathrm{d}S \\
\mathcal{S}_5 &= \int\limits_{\partial\Omega} q^{\text{adj}}_k \mathsf{n}_k \frac{\partial T}{\partial b_i} \mathrm{d}S \\
\mathcal{S}_6 &= \int\limits_{\partial\Omega} \tilde{\nu}_a \mathsf{n}_k \frac{\partial \left(\rho \tilde{\nu} v^R_k\right)}{\partial b_i} \mathrm{d}S \\
\mathcal{S}_7 &= -\frac{1}{\mathrm{Re}_0\, \sigma} \int\limits_{\partial\Omega} \rho \tilde{\nu}_a \mathsf{n}_k \frac{\partial}{\partial b_i} \left\{ \left[\nu + \left(1 + c_{b_2}\right) \tilde{\nu}\right] \frac{\partial \tilde{\nu}}{\partial x_k} \right\} \mathrm{d}S \\
\mathcal{S}_8 &= \frac{1}{\mathrm{Re}_0\, \sigma} \int\limits_{\partial\Omega} \frac{\partial \left(\rho \tilde{\nu}_a\right)}{\partial x_k} \left[\nu + \left(1 + c_{b_2}\right) \tilde{\nu}\right] \mathsf{n}_k \frac{\partial \tilde{\nu}}{\partial b_i} \mathrm{d}S \\
\mathcal{S}_9 &= \frac{c_{b_2}}{\mathrm{Re}_0\, \sigma} \int\limits_{\partial\Omega} \rho \tilde{\nu}_a \tilde{\nu} \mathsf{n}_k \frac{\partial}{\partial b_i} \left(\frac{\partial \tilde{\nu}}{\partial x_k} \right) \mathrm{d}S \\
\mathcal{S}_{10} &= \frac{c_{b_2}}{\mathrm{Re}_0\, \sigma} \int\limits_{\partial\Omega} \frac{\partial \left(\rho \tilde{\nu}_a \tilde{\nu}\right)}{\partial x_k} \mathsf{n}_k \frac{\partial \tilde{\nu}}{\partial b_i} \mathrm{d}S \\
\mathcal{S}_{11} &= \int\limits_{\partial\Omega} \rho \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqr} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \frac{\partial v^A_r}{\partial b_i} \mathrm{d}S
\end{aligned}
\tag{3.1.4.1}
$$

In order to eliminate any dependency of the sensitivity derivatives computation formula from variations in flow variables along the boundaries, any integral containing such variations must be eliminated. However, the flow boundary conditions must also be taken into account, meaning that variations in the imposed quantities (such as velocity for the wall or static pressure for the outlet boundaries) are zero. Consequently, integrals are developed so that variations in imposed quantities are eliminated. The factors multiplied with the remaining flow quantity variations (like variations in pressure for the wall boundaries) should be set to zero giving rise to the boundary conditions of adjoint mean-flow and Spalart-Allmaras PDEs (ABC).

Upon convergence of the adjoint PDEs (with the appropriate boundary conditions), the expression for computing the sensitivity derivatives includes only surface integrals containing variations in geometric quantities.

Each type of boundary is separately considered in the following sections.

3.1.4.1 Wall Boundaries

Wall boundaries are split to slip or no-slip, stationary or rotating ones. Some of the surface integrals are treated in a similar manner for some of these boundaries, while others require a different treatment. The flow boundary conditions for the wall boundaries are described in Section 2.2.1.

Slip Wall Boundaries $S_{W^{\text{Slip}}}$

Integrals, the treatment of which is common between stationary and rotating slip wall boundaries are presented first:

Term $\mathcal{S}_5$ is expanded as

$$\mathcal{S}_5 = \begin{cases} \underbrace{\int_{S_{W^{\text{Slip}}}} q_k^{\text{adj}} \mathsf{n}_k \frac{\delta T}{\delta b_i} \mathrm{d}S}_{\text{ABC}^T} - \underbrace{\int_{S_{W^{\text{Slip}}}} q_k^{\text{adj}} \mathsf{n}_k \frac{\partial T}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \text{, for adiabatic or constant heat flux wall.} \\ \cancel{\int_{S_{W^{\text{Slip}}}} q_k^{\text{adj}} \mathsf{n}_k \frac{\delta T}{\delta b_i} \mathrm{d}S} - \underbrace{\int_{S_{W^{\text{Slip}}}} q_k^{\text{adj}} \mathsf{n}_k \frac{\partial T}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \text{, for constant temperature wall.} \end{cases} \tag{3.1.4.2}$$

For slip wall boundaries, apart from the no-penetration condition and any thermal condition, if the flow is assumed turbulent, a mirror condition for $\tilde{\nu}$ is imposed, namely $\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k = 0$. This is taken into account when the variation of the normal derivative of $\tilde{\nu}$ appears in the development of the surface integrals. The rest of the terms are developed as follows:

$$\mathcal{S}_7 = \cancel{-\frac{1}{\mathrm{Re}_0\, \sigma} \int_{S_{W^{\text{Slip}}}} \rho \tilde{\nu}_a \frac{\delta}{\delta b_i} \left\{ \left[\nu + (1 + c_{b_2}) \tilde{\nu}\right] \frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \mathrm{d}S \right\}}$$

$$\underbrace{+\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\rho\tilde{\nu}_a\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\mathrm{SD}}$$

$$\underbrace{+\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\rho\tilde{\nu}_a\mathsf{n}_k\frac{\partial}{\partial x_\ell}\left\{\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\frac{\partial x_\ell}{\partial b_i}\mathrm{d}S}_{\mathrm{SD}} \tag{3.1.4.3}$$

$$\mathcal{S}_8=\underbrace{\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\mathsf{n}_k\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}_{\mathrm{ABC}^{\tilde{\nu}}}$$

$$\underbrace{-\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\mathsf{n}_k\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\mathrm{SD}} \tag{3.1.4.4}$$

$$\mathcal{S}_9=\cancel{\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\rho\tilde{\nu}_a\tilde{\nu}\frac{\delta}{\delta b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\mathrm{d}S\right)}\underbrace{-\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\rho\tilde{\nu}_a\tilde{\nu}\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\mathrm{SD}}$$

$$\underbrace{-\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\rho\tilde{\nu}_a\tilde{\nu}\mathsf{n}_k\frac{\partial}{\partial x_\ell}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\mathrm{SD}} \tag{3.1.4.5}$$

$$\mathcal{S}_{10}=\underbrace{-\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\tilde{\nu}\mathsf{n}_k\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}_{\mathrm{ABC}^{\tilde{\nu}}}\underbrace{+\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{Slip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\tilde{\nu}\mathsf{n}_k\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\mathrm{SD}} \tag{3.1.4.6}$$

Terms marked with SD contribute to the sensitivity derivatives expression. In the next two paragraphs, the rest of the integrals are developed separately for stationary and rotating slip walls.

Stationary Slip Wall Boundaries $S_{W^{\mathrm{Slip,St}}}$

For stationary slip wall boundaries, the no-penetration condition applies $\left(v_k^A\mathsf{n}_k=0\right)$. Additionally, in order to simulate stationary walls, when solving the steady state flow equations in a relative reference frame in PUMA, the geometry of these boundaries must be such that $v_k^F\mathsf{n}_k=0$. The combination of the two leads to $v_k^R\mathsf{n}_k=0$. Consequently, $\dfrac{\delta\left(v_k^A\mathsf{n}_k\right)}{\delta b_i}=0$ and $\dfrac{\delta\left(v_k^R\mathsf{n}_k\right)}{\delta b_i}=0$. In case of shape optimization involving parameterized slip stationary wall boundaries, the parameterization of these boundaries must adhere to these last to expressions.

Integral $\mathcal{S}_1$ is, then, expanded as

$$\mathcal{S}_1 = \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_1 \mathsf{n}_k \frac{\partial f_{1k}^{inv}}{\partial b_i} \mathrm{d}S}_{\mathcal{S}_1^{\text{CONTINUITY}}} + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1} \mathsf{n}_k \frac{\partial f_{(m+1)k}^{inv}}{\partial b_i} \mathrm{d}S}_{\mathcal{S}_1^{\text{MOMENTUM}_m}} + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5 \mathsf{n}_k \frac{\partial f_{5k}^{inv}}{\partial b_i} \mathrm{d}S}_{\mathcal{S}_1^{\text{ENERGY}}} \tag{3.1.4.7}$$

where

$$\mathcal{S}_1^{\text{CONTINUITY}} = \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_1 \frac{\delta}{\delta b_i} \left(\rho v_k^R \mathsf{n}_k \mathrm{d}S\right)} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_1 \mathsf{n}_k \frac{\partial \left(\rho v_k^R\right)}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S - \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_1 \rho v_k^R \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.1.4.8}$$

$$\begin{aligned}\mathcal{S}_1^{\text{MOMENTUM}_m} = & \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1} \frac{\delta}{\delta b_i} \left(\rho v_m^A v_k^R \mathsf{n}_k \mathrm{d}S\right)} + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1} \mathsf{n}_m \frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} \\ & \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1} \mathsf{n}_k \frac{\partial}{\partial x_\ell} \left(\rho v_m^A v_k^R + p \delta_{mk}\right) \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S - \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1} \rho v_m^A v_k^R \frac{\delta \mathsf{n}_k \mathrm{d}S}{\delta b_i}}_{\text{SD}}\end{aligned} \tag{3.1.4.9}$$

$$\begin{aligned}\mathcal{S}_1^{\text{ENERGY}} = & \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5 \frac{\delta}{\delta b_i} \left[\left(\rho h_t v_k^R \mathsf{n}_k + v_k^F \mathsf{n}_k p\right) \mathrm{d}S\right]} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5 \mathsf{n}_k \frac{\partial}{\partial x_\ell} \left(\rho h_t v_k^R + v_k^F p\right) \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \\ & \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5 \left(\rho h_t v_k^R + v_k^F p\right) \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}\end{aligned} \tag{3.1.4.10}$$

The $\mathcal{S}_2$ term is similarly expanded as follows

$$\mathcal{S}_2 = \underbrace{- \int\limits_{S_{W^{\text{Slip}}}} \Psi_{m+1} \frac{\partial \tau_{km}}{\partial b_i} \mathsf{n}_k \mathrm{d}S}_{\mathcal{S}_2^{\text{MOMENTUM}_m}} \underbrace{- \int\limits_{S_{W^{\text{Slip}}}} \Psi_5 \frac{\partial}{\partial b_i} \left(v_\ell^A \tau_{\ell k} + q_k\right) \mathsf{n}_k \mathrm{d}S}_{\mathcal{S}_2^{\text{ENERGY}}} \tag{3.1.4.11}$$

where

$$
\begin{aligned}
\mathcal{S}_2^{\text{MOMENTUM}_m} = & -\underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\mathsf{n}_m \frac{\delta}{\delta b_i}\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)\mathrm{d}S}_{\text{ABC}^{\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}} + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell \frac{\delta\left(\mathsf{n}_m\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\
& - \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\frac{\delta}{\delta b_i}\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\mathsf{t}_m\mathrm{d}S\right)} \\
& + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\tau_{km}\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i} + \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\mathsf{n}_k\frac{\partial\tau_{km}}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}}
\end{aligned}
\tag{3.1.4.12}
$$

$$
\begin{aligned}
\mathcal{S}_2^{\text{ENERGY}} = & -\cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5\frac{\delta}{\delta b_i}\left(v_\ell^A\tau_{k\ell}\mathsf{n}_k\mathrm{d}S\right)} + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5\frac{\partial}{\partial x_\ell}\left(v_m^A\tau_{mk}+q_k\right)\mathsf{n}_k\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}} \\
& + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5\left(v_\ell^A\tau_{\ell k}+q_k\right)\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}\mathrm{d}S}_{\text{SD}} - \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_5\frac{\delta}{\delta b_i}\left(q_k\mathsf{n}_k\mathrm{d}S\right)}_{\text{ABC}^{\left(q_k\mathsf{n}_k\right)}}
\end{aligned}
\tag{3.1.4.13}
$$

The last integral in Eq. 3.1.4.13 contributes to the adjoint boundary condition if a constant temperature condition is imposed on the wall. Otherwise (adiabatic or constant heat flux wall), this term vanishes since $\frac{\delta(q_k\mathsf{n}_k)}{\delta b_i}=0$.

The rest of the surface integrals result in

$$
\begin{aligned}
\mathcal{S}_{3,4} = & \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \tau^{\text{adj}}_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\frac{\delta\left(v_m^A\mathsf{t}_m\mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(v_m^A\mathsf{t}_m\right)}} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \tau^{\text{adj}}_{km}\mathsf{n}_k\frac{\partial v_m^A}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S - \int\limits_{S_{W^{\text{Slip,St}}}} \tau^{\text{adj}}_{k\ell}\mathsf{n}_k\mathsf{n}_\ell v_m^A\frac{\delta\left(\mathsf{n}_m\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\
& - \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \tau^{\text{adj}}_{k\ell}\mathsf{n}_k\mathsf{t}_\ell v_m^A\frac{\delta\left(\mathsf{t}_m\mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_1}
\end{aligned}
\tag{3.1.4.14}
$$

$$
\mathcal{S}_6 = -\underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a\mathsf{n}_k\frac{\partial\left(\rho\tilde{\nu}v_k^R\right)}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S - \int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a\rho\tilde{\nu}v_k^R\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a\frac{\delta\left(\rho\tilde{\nu}v_k^R\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}
\tag{3.1.4.15}
$$

$$
\begin{aligned}
\mathcal{S}_{11} = & \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqr} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \frac{\partial v^A_r}{\partial x_n} \frac{\delta x_n}{\delta b_i} \mathrm{d}S}_{\text{SD}} \\
& \cancel{+ \int\limits_{S_{W^{\text{Slip,St}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqs} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \mathsf{n}_s \frac{\delta}{\delta b_i} \left(v^A_r \mathsf{n}_r \mathrm{d}S\right)} \\
& \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqs} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \mathsf{n}_s v^A_r \frac{\delta\left(\mathsf{n}_r \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\
& \underbrace{+ \int\limits_{S_{W^{\text{Slip,St}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqs} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \mathsf{t}_s \frac{\delta}{\delta b_i} \left(v^A_r \mathsf{t}_r \mathrm{d}S\right)}_{\text{ABC}^{\left(v^A_r \mathsf{t}_r\right)}} \\
& \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqs} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \mathsf{t}_s v^A_r \frac{\delta\left(\mathsf{t}_r \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}^2}
\end{aligned}
\qquad (3.1.4.16)
$$

The third integral in Eq. 3.3 for stationary slip wall boundaries can be expressed as

$$
\begin{aligned}
\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\delta F_S}{\delta b_i} \mathrm{d}S = & \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial x_k} \frac{\delta x_k}{\delta b_i} \mathrm{d}S + \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\mathsf{n}_k \mathrm{d}S\right)} \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\mathsf{t}_k \mathrm{d}S\right)} \frac{\delta\left(\mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_3} \\
& \underbrace{+ \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial p} \frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} \underbrace{+ \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial T} \frac{\delta T}{\delta b_i} \mathrm{d}S}_{\text{ABC}^T} \underbrace{+ \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(q_k \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta\left(q_k \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(q_k \mathsf{n}_k\right)}} \\
& \underbrace{+ \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial\left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)} \frac{\delta\left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{\left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_s}{\partial\left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell\right)} \frac{\delta\left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell\right)}{\delta b_i} \mathrm{d}S}
\end{aligned}
$$

$$+\cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v^A_k \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta\left(v^A_k \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v^A_k \mathsf{t}_k \mathrm{d}S\right)} \frac{\delta\left(v^A_k \mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(v^A_k \mathsf{t}_k\right)}}$$

$$+ \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial \tilde{\nu}} \frac{\delta \tilde{\nu}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{\tilde{\nu}}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta}{\delta b_i}\left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \mathrm{d}S\right)} \tag{3.1.4.17}$$

In order to eliminate the terms denoted as ABC^p, the following boundary conditions must be set

$$\Psi_{m+1}\mathsf{n}_m = \begin{cases} -\dfrac{\partial F_S}{\partial p}, \text{ at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ 0, \text{ at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.1.4.18}$$

Similarly, terms denoted as ABC$^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}$ are eliminated

$$\Psi_{m+1}\mathsf{n}_m = \begin{cases} \dfrac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}, \text{ at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ 0, \text{ at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.1.4.19}$$

From Eq. 3.1.4.18 and Eq. 3.1.4.19 it is evident that, in order to obtain a unique adjoint boundary condition along $S^{\text{Obj}}_{W^{\text{Slip,St}}}$, the objective function must be such that $\dfrac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)} = -\dfrac{\partial F_S}{\partial p}$. For common objective functions such as force and torque computed on slip wall boundaries, this condition is satisfied.

Elimination of the rest of the ABC terms leads to

$$\tau^{\text{adj}}_{k\ell}\mathsf{n}_k\mathsf{t}_\ell + \rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S}, S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v^A_m}{\partial x_\ell}\mathsf{n}_q\mathsf{t}_s = \begin{cases} -\dfrac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v^A_k \mathsf{t}_k \mathrm{d}S\right)}, \text{ at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ 0, \text{ at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.1.4.20}$$

$$\frac{1}{\text{Re}_0\,\sigma}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\left(\nu + \tilde{\nu}\right)\mathsf{n}_k = \begin{cases} -\dfrac{\partial F_S}{\partial \tilde{\nu}}, \text{ at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ 0, \text{ at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.1.4.21}$$

If the wall is adiabatic or a constant heat flux is imposed, then the elimination of terms ABC^T leads to

$$q_k^{\text{adj}}\mathsf{n}_k = \begin{cases} -\dfrac{\partial F_S}{\partial T} \text{ , at } S_{W^{\text{Slip,St}}}^{\text{Obj}} \\ 0 \text{ , at } S_{W^{\text{Slip,St}}} \setminus S_{W^{\text{Slip,St}}}^{\text{Obj}} \end{cases} \tag{3.1.4.22}$$

while terms ABC$^{(q_k\mathsf{n}_k)}$ vanish automatically. Similarly, for a constant temperature wall, ABC^{T} terms vanish automatically and the elimination of the ABC$^{(q_k\mathsf{n}_k)}$ terms leads to the following adjoint thermal boundary conditions,

$$\Psi_5 = \begin{cases} \dfrac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(q_k\mathsf{n}_k\mathrm{d}S\right)} \text{ , at } S_{W^{\text{Slip,St}}}^{\text{Obj}} \\ 0 \text{ , at } S_{W^{\text{Slip,St}}} \setminus S_{W^{\text{Slip,St}}}^{\text{Obj}} \end{cases} \tag{3.1.4.23}$$

All of the aforementioned adjoint boundary conditions are taken into account when computing the l.h.s.and r.h.s.terms of the discretized adjoint equations. It must be noted that the application of the adjoint boundary condition defined by Eq. 3.1.4.20 leads to elimination of the terms $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$, when $\dfrac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(\mathsf{t}_k\mathrm{d}S\right)} = \dfrac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(v_k^A t_k\mathrm{d}S\right)}$. For common objective functions defined along slip wall boundaries (such as force and torque) this constraint is satisfied. In any other case, these integrals contribute to the sensitivity derivative formula.

Rotating Slip Wall Boundaries $S_{W^{\text{Slip,Rot}}}$

For rotating slip walls, the no-penetration condition applies which, in this case, is expressed as $v_k^R\mathsf{n}_k = 0$. As a result, $\dfrac{\delta\left(v_k^R\mathsf{n}_k\right)}{\delta b_i} = 0$ and $\dfrac{\delta\left(v_k^A\mathsf{n}_k\right)}{\delta b_i} = \dfrac{\delta\left(v_k^F\mathsf{n}_k\right)}{\delta b_i} = v_k^F\dfrac{\delta\mathsf{n}_k}{\delta b_i} + \mathsf{n}_k\dfrac{\partial v_k^F}{\partial x_\ell}\dfrac{\delta x_\ell}{\delta b_i}$. The integrals arising from the differentiation of the inviscid terms are

$$\begin{aligned}
\mathcal{S}_1^{\text{CONTINUITY}} &= \cancel{\int_{S_{W^{\text{Slip,Rot}}}} \Psi_1 \frac{\delta}{\delta b_i}\left(\rho v_k^R\mathsf{n}_k\mathrm{d}S\right)} \underbrace{- \int_{S_{W^{\text{Slip,Rot}}}} \Psi_1\mathsf{n}_k\frac{\partial\left(\rho v_k^R\right)}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S - \int_{S_{W^{\text{Slip,Rot}}}} \Psi_1\, \rho v_k^R\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\
\mathcal{S}_1^{\text{MOMENTUM}_m} &= \cancel{\int_{S_{W^{\text{Slip,Rot}}}} \Psi_{m+1}\frac{\delta}{\delta b_i}\left(\rho v_m^A v_k^R\mathsf{n}_k\mathrm{d}S\right)} + \underbrace{\int_{S_{W^{\text{Slip,Rot}}}} \Psi_{m+1}\mathsf{n}_m\frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p}
\end{aligned} \tag{3.1.4.24}$$

$$-\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_{m+1}\mathsf{n}_k\frac{\partial}{\partial x_\ell}\left(\rho v_m^A v_k^R+p\delta_{mk}\right)\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S-\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_{m+1}\rho v_m^A v_k^R\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.1.4.25}$$

$$\begin{aligned}\mathcal{S}_1^{\text{ENERGY}}=&\cancel{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5\frac{\delta}{\delta b_i}\left(\rho h_t v_k^R\mathsf{n}_k\mathrm{d}S\right)}+\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5 v_k^F\mathsf{n}_k\frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p}\\&-\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5\rho h_t v_k^R\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}-\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5 p\mathsf{n}_k\frac{\delta v_k^F}{\delta x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}}\end{aligned} \tag{3.1.4.26}$$

Similarly, term $\mathcal{S}_2$ leads to

$$\begin{aligned}\mathcal{S}_2^{\text{MOMENTUM}}=&-\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_{m+1}\mathsf{n}_m\frac{\delta}{\delta b_i}\left(\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell\right)\mathrm{d}S}_{\text{ABC}^{\left(\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell\right)}}+\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_{m+1}\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell\frac{\delta\left(\mathsf{n}_m\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}\\&-\cancel{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_{m+1}\frac{\delta}{\delta b_i}\left(\tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell\mathsf{t}_m\mathrm{d}S\right)}\\&+\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_{m+1}\mathsf{n}_k\frac{\partial\tau_{km}}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S+\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_{m+1}\tau_{mk}\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}\end{aligned} \tag{3.1.4.27}$$

$$\begin{aligned}\mathcal{S}_2^{\text{ENERGY}}=&-\cancel{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5\frac{\delta}{\delta b_i}\left(v_\ell^R\tau_{mk}\mathsf{n}_k\mathsf{n}_m\mathsf{n}_\ell\mathrm{d}S\right)}-\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5 v_\ell^F\mathsf{n}_\ell\frac{\delta}{\delta b_i}\left(\tau_{mk}\mathsf{n}_k\mathsf{n}_m\right)\mathrm{d}S}_{\text{ABC}^{\left(\tau_{mk}\mathsf{n}_k\mathsf{n}_m\right)}}\\&+\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5\mathsf{n}_k\frac{\partial}{\partial x_\ell}\left(v_\ell^A\tau_{\ell k}+q_k\right)\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S+\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5\left(v_\ell^A\tau_{\ell k}+q_k\right)\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}\\&+\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5 v_\ell^F\tau_{mk}\mathsf{n}_k\mathsf{n}_m\frac{\delta\left(\mathsf{n}_\ell\mathrm{d}S\right)}{\delta b_i}-\int\limits_{S_{W^{\text{Slip,Rot}}}}\Psi_5\tau_{mk}\mathsf{n}_k\mathsf{n}_m\mathsf{n}_\ell\frac{\partial v_\ell^F}{\partial x_q}\frac{\delta x_q}{\delta b_i}ddS}_{\text{SD}}\end{aligned}$$

$$\underbrace{-\int\limits_{S_{W^{\text{Slip,Rot}}}} \Psi_5 \frac{\delta}{\delta b_i}\left(q_k \mathsf{n}_k \mathrm{d}S\right)}_{\text{ABC}^{(q_k n_k)}} \tag{3.1.4.28}$$

The development of term $\mathcal{S}_{3,4}$ leads to

$$\begin{aligned}\mathcal{S}_{3,4} = &\underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}} \tau^{\text{adj}}_{k\ell} \mathsf{n}_k \mathsf{t}_\ell \frac{\delta}{\delta b_i}\left(v^R_m \mathsf{t}_m \mathrm{d}S\right)}_{\text{ABC}^{\left(v^R_m \mathsf{t}_m\right)}} \underbrace{- \int\limits_{S_{W^{\text{Slip,Rot}}}} \tau^{\text{adj}}_{km} \mathsf{n}_k \frac{\partial v^A_m}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i} + \int\limits_{S_{W^{\text{Slip,Rot}}}} \tau^{\text{adj}}_{km} \mathsf{n}_k \frac{\partial v^F_m}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}} \\ &\underbrace{- \int\limits_{S_{W^{\text{Slip,Rot}}}} \tau^{\text{adj}}_{k\ell} \mathsf{n}_k \mathsf{n}_\ell v^R_m \frac{\delta\left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \cancel{\int\limits_{S_{W^{\text{Slip,Rot}}}} \tau^{\text{adj}}_{k\ell} \mathsf{n}_k \mathsf{n}_\ell \frac{\delta}{\delta b_i}\left(v^R_m \mathsf{n}_m \mathrm{d}S\right)} + \underbrace{\int\limits_{S_{W^{\text{Slip,Rot}}}} \tau^{\text{adj}}_{k\ell} \mathsf{n}_k \mathsf{t}_\ell v^R_m \frac{\delta\left(\mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_1}\end{aligned} \tag{3.1.4.29}$$

The remaining terms $(\mathcal{S}_6,\ \mathcal{S}_{11})$ are expanded below

$$\mathcal{S}_6 = \cancel{\int\limits_{S_{W^{\text{Slip,Rot}}}} \tilde{\nu}_a \frac{\delta}{\delta b_i}\left(\rho\tilde{\nu} v^R_k \mathsf{n}_k \mathrm{d}S\right)} \underbrace{- \int\limits_{S_{W^{\text{Slip,Rot}}}} \tilde{\nu}_a \mathsf{n}_k \frac{\partial\left(\rho\tilde{\nu} v^R_k\right)}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S - \int\limits_{S_{W^{\text{Slip,Rot}}}} \tilde{\nu}_a \rho\tilde{\nu} v^R_k \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.1.4.30}$$

$$\begin{aligned}\mathcal{S}_{11} = &\underbrace{-\int\limits_{S_{W^{\text{Slip,Rot}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m}\varepsilon_{kqr} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \frac{\partial v^R_r}{\partial x_n}\frac{\delta x_n}{\delta b_i}\mathrm{d}S}_{\text{SD}} \\ &+ \cancel{\int\limits_{S_{W^{\text{Slip,Rot}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m}\varepsilon_{kqs} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \mathsf{n}_s \frac{\delta}{\delta b_i}\left(v^R_r \mathsf{n}_r \mathrm{d}S\right)} \\ &\underbrace{- \int\limits_{S_{W^{\text{Slip,Rot}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m}\varepsilon_{kqs} \frac{\partial v^A_m}{\partial x_\ell} \mathsf{n}_q \mathsf{n}_s v^R_r \frac{\delta\left(\mathsf{n}_r \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}\end{aligned}$$

$$\begin{aligned}
&\underbrace{+ \int\limits_{S_{W^{\text{Slip,Rot}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m}\varepsilon_{kqs} \frac{\partial v_m^A}{\partial x_\ell} \mathsf{n}_q \mathsf{t}_s \frac{\delta}{\delta b_i}\left(v_r^R \mathsf{t}_r \mathrm{d}S\right)}_{\text{ABC}^{\left(v_r^R \mathsf{t}_r\right)}} \\
&\underbrace{- \int\limits_{S_{W^{\text{Slip,Rot}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m}\varepsilon_{kqs} \frac{\partial v_m^A}{\partial x_\ell} \mathsf{n}_q \mathsf{t}_s v_r^R \frac{\delta\left(\mathsf{t}_r \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}^2}
\end{aligned} \qquad (3.1.4.31)$$

The third integral of Eq. 3.3 for inviscid rotating wall boundaries can be expressed as

$$\begin{aligned}
\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\delta F_S}{\delta b_i} \mathrm{d}S = &\underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial x_k}\frac{\delta x_k}{\delta b_i} \mathrm{d}S + \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\mathsf{n}_k \mathrm{d}S\right)} \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\mathsf{t}_k \mathrm{d}S\right)} \frac{\delta\left(\mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_3} \\
&+ \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial p}\frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial T}\frac{\delta T}{\delta b_i} \mathrm{d}S}_{\text{ABC}^T} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(q_k \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta\left(q_k \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(q_k \mathsf{n}_k\right)}} \\
&+ \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)} \frac{\delta\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_s}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)} \frac{\delta\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)}{\delta b_i} \mathrm{d}S} \\
&+ \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v_k^R \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta\left(v_k^R \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v_k^R \mathsf{t}_k \mathrm{d}S\right)} \frac{\delta\left(v_k^R \mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(v_k^R \mathsf{t}_k\right)}} \\
&+ \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial \tilde{\nu}}\frac{\delta \tilde{\nu}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{\tilde{\nu}}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)} \frac{\delta}{\delta b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)}
\end{aligned} \qquad (3.1.4.32)$$

In order to eliminate terms ABC^p and ABC$^{(\tau_{km}\mathsf{n}_k\mathsf{n}_m)}$, the following adjoint boundary conditions are imposed

$$\left(\Psi_{m+1}+\Psi_5 v_m^F\right)\mathsf{n}_m=\begin{cases}-\dfrac{\partial F_S}{\partial p}\text{ , at }S_{W^{\text{Slip,Rot}}}^{\text{Obj}}\\ 0\text{ , at }S_{W^{\text{Slip,Rot}}}\setminus S_{W^{\text{Slip,Rot}}}^{\text{Obj}}\end{cases}\tag{3.1.4.33}$$

$$\left(\Psi_{m+1}+\Psi_5 v_m^F\right)\mathsf{n}_m=\begin{cases}\dfrac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}\text{ , at }S_{W^{\text{Slip,Rot}}}^{\text{Obj}}\\ 0\text{ , at }S_{W^{\text{Slip,Rot}}}\setminus S_{W^{\text{Slip,Rot}}}^{\text{Obj}}\end{cases}\tag{3.1.4.34}$$

From the combination of Eqs. 3.1.4.33 and 3.1.4.34, in order to obtain a unique adjoint boundary condition it must hold that $\dfrac{\partial F_S}{\partial p}=-\dfrac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}$.

The elimination of the $\text{ABC}^{\left(v_k^R\mathsf{t}_k\right)}$ term leads to one more boundary condition for rotating slip wall boundaries, namely

$$\tau_{k\ell}^{\text{adj}}\mathsf{n}_k\mathsf{t}_\ell+\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqs}\frac{\partial v_m^A}{\partial x_\ell}\mathsf{n}_q\mathsf{t}_s=\begin{cases}-\dfrac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(v_k^R\mathsf{t}_k\mathrm{d}S\right)}\text{ , at }S_{W^{\text{Slip,Rot}}}^{\text{Obj}}\\ 0\text{ , at }S_{W^{\text{Slip,Rot}}}\setminus S_{W^{\text{Slip,Rot}}}^{\text{Obj}}\end{cases}\tag{3.1.4.35}$$

Terms $\text{ABC}^{\tilde{\nu}}$, $\text{ABC}^{(q_k\mathsf{n}_k)}$ and ABC^T are eliminated as for the stationary slip walls. Again, we note that the boundary condition described by Eq. 3.1.4.35 leads to the elimination of terms $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, when F is such that $\dfrac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(\mathsf{t}_k\mathrm{d}S\right)}=\dfrac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(v_k^R t_k\mathrm{d}S\right)}$.

No-Slip Wall Boundaries $S_{W^{\text{NoSlip}}}$

Along no-slip walls, the boundary condition imposed for the Spalart-Allmaras variable is $\tilde{\nu}=0$. The integrals whose treatment is common between stationary and rotating no-slip wall boundaries are presented first. These are treated as follows:

Term $\mathcal{S}_5$ is expanded as for slip walls (Eq. 3.1.4.2). Terms $\mathcal{S}_7$ to $\mathcal{S}_{10}$ lead to

$$\mathcal{S}_7=\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int_{S_{W^{\text{NoSlip}}}}\rho\tilde{\nu}_a\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\delta}{\delta b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\mathrm{d}S\right)}_{\text{ABC}^{\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}}$$
$$\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int_{S_{W^{\text{NoSlip}}}}\rho\tilde{\nu}_a\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\mathcal{P}\left(\nu,\rho\right)\mathcal{P}\left(\rho,p\right)\frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p}$$

$$\underbrace{-\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\rho\tilde{\nu}_a\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\left[\mathcal{P}\left(\nu,\mu\right)\mathcal{P}\left(\mu,T\right)+\mathcal{P}\left(\nu,\rho\right)\mathcal{P}\left(\rho,T\right)\right]\frac{\delta T}{\delta b_i}\mathrm{d}S}_{\mathrm{ABC}^T}$$

$$\cancel{-\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\rho\tilde{\nu}_a\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\left(1+c_{b_2}\right)\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}$$

$$\underbrace{+\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\rho\tilde{\nu}_a\mathsf{n}_k\frac{\partial}{\partial x_\ell}\left\{\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\mathrm{SD}}$$

$$\underbrace{+\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\rho\tilde{\nu}_a\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\mathrm{SD}} \tag{3.1.4.36}$$

$$\mathcal{S}_8=\cancel{\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\mathsf{n}_k\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}$$

$$\underbrace{-\frac{1}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\mathsf{n}_k\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\mathrm{SD}} \tag{3.1.4.37}$$

$$\mathcal{S}_9=\cancel{\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\rho\tilde{\nu}_a\tilde{\nu}\mathsf{n}_k\frac{\partial}{\partial b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)\mathrm{d}S} \tag{3.1.4.38}$$

$$\mathcal{S}_{10}=\cancel{-\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\mathsf{n}_k\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}+\underbrace{\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{S_{W^{\mathrm{NoSlip}}}}\frac{\partial\left(\rho\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\mathsf{n}_k\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\mathrm{SD}} \tag{3.1.4.39}$$

In the following two paragraphs, the rest of the terms are developed separately for stationary and rotating no-slip wall boundaries.

Stationary No-Slip Wall Boundaries $S_{W^{\mathrm{NoSlip,St}}}$

Along the stationary no-slip wall boundaries, the flow boundary condition is expressed as $v_k^A=0,(k=1,2,3)$. Consequently, it holds that $\frac{\delta v_k^A}{\delta b_i}=0$. The $\mathcal{S}_1$ and $\mathcal{S}_2$ integrals lead to

$$
\begin{aligned}
\mathcal{S}_1^{\text{CONTINUITY}} =& \cancel{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_1 \frac{\delta}{\delta b_i}\left(\rho v_k^A \mathsf{n}_k \mathrm{d}S\right)} - \cancel{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_1 \frac{\delta}{\delta b_i}\left(\rho v_k^F \mathsf{n}_k \mathrm{d}S\right)} - \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_1 \mathsf{n}_k \frac{\partial\left(\rho v_k^R\right)}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}} \\
&- \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_1 \rho v_k^R \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}
\end{aligned} \tag{3.1.4.40}
$$

$$
\begin{aligned}
\mathcal{S}_1^{\text{MOMENTUM}_m} =& \cancel{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \frac{\delta}{\delta b_i}\left(\rho v_m^A v_k^R \mathsf{n}_k \mathrm{d}S\right)} - \cancel{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \rho v_m^A v_k^R \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} + \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \mathsf{n}_m \frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p} \\
&- \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \mathsf{n}_k \frac{\partial}{\partial x_\ell}\left(\rho v_m^A v_k^R + p\delta_{km}\right)\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}}
\end{aligned} \tag{3.1.4.41}
$$

$$
\begin{aligned}
\mathcal{S}_1^{\text{ENERGY}} =& \cancel{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_5 \frac{\delta}{\delta b_i}\left(\rho v_k^R h_t \mathsf{n}_k \mathrm{d}S\right)} - \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_5 \rho h_t v_k^R \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \cancel{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_5 v_m^F \mathsf{n}_k \frac{\delta p}{\delta b_i}\mathrm{d}S} \\
&+ \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_5 p \mathsf{n}_k \frac{\partial v_k^F}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i} - \int\limits_{S_W^{\text{NoSlip,St}}} \Psi_5 \mathsf{n}_k \frac{\partial}{\partial x_\ell}\left(\rho h_t v_k^R + p v_k^F\right)\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}}
\end{aligned} \tag{3.1.4.42}
$$

$$
\begin{aligned}
\mathcal{S}_2^{\text{MOM}_m} =& - \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \mathsf{n}_m \frac{\delta}{\delta b_i}\left(\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell\right)\mathrm{d}S}_{\text{ABC}^{(\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell)}} - \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \mathsf{t}_m \frac{\delta}{\delta b_i}\left(\tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell\right)\mathrm{d}S}_{\text{ABC}^{(\tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell)}} \\
&+ \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{q+1} \tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell\mathsf{t}_q\mathsf{t}_m \frac{\delta\left(\mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}^1} + \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \mathsf{n}_k \frac{\partial \tau_{km}}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}} \\
&+ \underbrace{\int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \tau_{mk} \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i} + \int\limits_{S_W^{\text{NoSlip,St}}} \Psi_{m+1} \tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}
\end{aligned}
$$

$$\underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{q+1} \mathsf{n}_q \tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell \mathsf{t}_m \frac{\delta (\mathsf{n}_m \mathrm{d}S)}{\delta b_i}}_{\text{SD}} \tag{3.1.4.43}$$

$$\mathcal{S}_2^{\text{ENER}} = \cancel{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_5 \mathsf{n}_k \frac{\delta \left(v_m^A \tau_{mk}\right)}{\delta b_i} \mathrm{d}S} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_5 \frac{\delta (q_k \mathsf{n}_k \mathrm{d}S)}{\delta b_i}}_{\text{ABC}^{(q_k \mathsf{n}_k)}} \underbrace{+ \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_5 q_k \frac{\delta (\mathsf{n}_k \mathrm{d}S)}{\delta b_i}}_{\text{SD}}$$
$$\underbrace{+ \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_5 \mathsf{n}_k \frac{\partial}{\partial x_\ell} \left(v_m^A \tau_{km} + q_k\right) \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \tag{3.1.4.44}$$

Analyzing the $\mathcal{S}_{3,4}$ term, we get

$$\mathcal{S}_{3,4} = \cancel{\int\limits_{S_{W^{\text{NoSlip,St}}}} \tau_{km}^{\text{adj}} \mathsf{n}_k \frac{\delta v_m^A}{\delta b_i} \mathrm{d}S} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \tau_{km}^{\text{adj}} \mathsf{n}_k \frac{\partial v_m^A}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \tag{3.1.4.45}$$

Similarly, terms $\mathcal{S}_6$ and $\mathcal{S}_{11}$ become

$$\mathcal{S}_6 = \cancel{\int\limits_{S_{W^{\text{NoSlip,St}}}} \tilde{\nu}_a \frac{\delta}{\delta b_i} \left(\rho \tilde{\nu} v_k^R \mathsf{n}_k \mathrm{d}S\right)} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \tilde{\nu}_a \rho v_k^R \tilde{\nu} \frac{\delta (\mathsf{n}_k \mathrm{d}S)}{\delta b_i} - \int\limits_{S_{W^{\text{NoSlip,St}}}} \tilde{\nu}_a \mathsf{n}_k \frac{\partial \left(\rho \tilde{\nu} v_k^R\right)}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \tag{3.1.4.46}$$

$$\mathcal{S}_{11} = \cancel{\int\limits_{S_{W^{\text{NoSlip,St}}}} \rho \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqr} \frac{\partial v_m^A}{\partial x_\ell} \mathsf{n}_q \frac{\delta v_r^A}{\delta b_i} \mathrm{d}S}$$
$$= \underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \rho \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqr} \frac{\partial v_m^A}{\partial x_\ell} \mathsf{n}_q \frac{\partial v_r^A}{\partial x_p} \frac{\delta x_p}{\delta b_i} \mathrm{d}S}_{\text{SD}} \tag{3.1.4.47}$$

For stationary no-slip stationary wall boundaries, the third integral of Eq. 3.3 , can be expressed as

$$
\begin{aligned}
\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\delta F_S}{\delta b_i} \mathrm{d}S = & \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial x_k}\frac{\delta x_k}{\delta b_i}\mathrm{d}S + \int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\mathsf{n}_k \mathrm{d}S\right)}\frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\mathsf{t}_k \mathrm{d}S\right)}\frac{\delta \left(\mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_2} \\
& + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial p}\frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial T}\frac{\delta T}{\delta b_i}\mathrm{d}S}_{\text{ABC}^T} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(q_k \mathsf{n}_k \mathrm{d}S\right)}\frac{\delta \left(q_k \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{(q_k \mathsf{n}_k)}} \\
& + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial \left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}\frac{\delta \left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}{\delta b_i}\mathrm{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_s}{\partial \left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)}\frac{\delta \left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)}{\delta b_i}\mathrm{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}} \\
& + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial v^A_k}\frac{\delta v^A_k}{\delta b_i}\mathrm{d}S} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial \tilde{\nu}}\frac{\delta \tilde{\nu}}{\delta b_i}\mathrm{d}S} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)}\frac{\delta}{\delta b_i}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)}_{\text{ABC}^{\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}}
\end{aligned}
\tag{3.1.4.48}
$$

Collecting and eliminating terms denoted as ABC^p in Eqs. 3.1.4.36-3.1.4.48, the following relation arises

$$
\Psi_{m+1}\mathsf{n}_m - \frac{1}{\text{Re}_0\,\sigma}\rho\tilde{\nu}_a\frac{\partial \tilde{\nu}}{\partial x_m}\mathsf{n}_m\mathcal{P}\left(\nu,\rho\right)\mathcal{P}\left(\rho,p\right) = \begin{cases} -\dfrac{\partial F_S}{\partial p} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases}
\tag{3.1.4.49}
$$

Similarly, elimination of terms denoted as ABC$^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}$ and ABC$^{(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}$ leads to

$$
\Psi_{m+1}\mathsf{n}_m = \begin{cases} \dfrac{\partial F_S}{\partial \left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases}
\tag{3.1.4.50}
$$

$$
\Psi_{m+1}\mathsf{t}_m = \begin{cases} \dfrac{\partial F_s}{\partial \left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases}
\tag{3.1.4.51}
$$

In order to eliminate the ABC$^{\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}$ terms, the following relation

$$\frac{1}{\mathrm{Re}_0\,\sigma}\rho\tilde{\nu}_a\nu = \begin{cases} \dfrac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases} \tag{3.1.4.52}$$

must hold. This leads to the adjoint boundary condition for $\tilde{\nu}_a$ on stationary no-slip walls, expressed as

$$\tilde{\nu}_a = \begin{cases} \dfrac{\mathrm{Re}_0\,\sigma}{\rho\nu}\dfrac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases} \tag{3.1.4.53}$$

Importing Eq. 3.1.4.53 into Eq. 3.1.4.49 and combining it with Eq. 3.1.4.50 in order to obtain a single adjoint boundary condition for $\Psi_{m+1}\mathsf{n}_m$, the objective function expression on stationary viscous wall boundaries must satisfy the following constraint

$$\frac{\partial F_S}{\partial\left(\tau_{\ell\ell}\mathsf{n}_k\mathsf{n}_\ell\right)} - \frac{1}{\nu}\frac{\partial\tilde{\nu}}{\partial x_m}\mathsf{n}_m\frac{\partial F_S \mathrm{d}S}{\partial\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}\mathcal{P}\left(\nu,\rho\right)\mathcal{P}\left(\rho,p\right) + \frac{\partial F_S}{\partial p} = 0 \tag{3.1.4.54}$$

If, additionally, the objective function satisfies $\dfrac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)}\tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell\mathsf{t}_m + \dfrac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(\mathsf{t}_m\mathrm{d}S\right)} = 0$, then terms $\mathcal{T}_1$ and $\mathcal{T}_2$ vanish automatically. Else, these terms contribute to the final expression of sensitivity derivatives.

For stationary no-slip adiabatic or constant heat flux walls, where $\frac{\delta(q_k\mathsf{n}_k)}{\delta b_i} = 0$, terms $\mathrm{ABC}^{(q_k\mathsf{n}_k)}$ vanish. If Eq. 3.1.4.53 is taken into account, the elimination of the ABC^{T} terms leads to the following thermal adjoint boundary condition

$$q_k^{\text{adj}}\mathsf{n}_k = \begin{cases} \dfrac{1}{\nu}\dfrac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\left[\mathcal{P}\left(\nu,\mu\right)\mathcal{P}\left(\mu,T\right) + \mathcal{P}\left(\nu,\rho\right)\mathcal{P}\left(\rho,T\right)\right] + \dfrac{\partial F_S}{\partial T} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases} \tag{3.1.4.55}$$

For stationary no-slip constant temperature walls, where $\frac{\delta T}{\delta b_i} = 0$, the ABC^{T} terms vanish and the thermal adjoint boundary conditions arise from the elimination of the $\mathrm{ABC}^{(q_k\mathsf{n}_k)}$ terms, namely

$$\Psi_5 = \begin{cases} \dfrac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(q_k\mathsf{n}_k \mathrm{d}S\right)} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases} \tag{3.1.4.56}$$

Rotating No-Slip Wall Boundaries $S_{W^{\text{NoSlip,Rot}}}$

On rotating no-slip walls, the flow boundary condition is expressed as $v_k^R = 0, (k = 1, 2, 3)$. Consequently, it holds that $\frac{\delta v_k^R}{\delta b_i} = 0$. In addition, $v_k^A = v_k^F$ and so $\frac{\delta v_k^A}{\delta b_i} = \frac{\partial v_k^F}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}$. The integrals $\mathcal{S}_1$ and $\mathcal{S}_2$ lead to

$$\mathcal{S}_1^{\text{CONTINUITY}} = \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_1 \mathsf{n}_k \frac{\delta\left(\rho v_k^R\right)}{\delta b_i} \mathrm{d}S} - \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_1 \mathsf{n}_k \frac{\partial\left(\rho v_k^R\right)}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \tag{3.1.4.57}$$

$$\begin{aligned}\mathcal{S}_1^{\text{MOMENTUM}_m} = & \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \mathsf{n}_k \frac{\delta\left(\rho v_k^R v_m^A\right)}{\delta b_i} \mathrm{d}S} + \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \mathsf{n}_m \frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} \\ & - \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \mathsf{n}_k \frac{\partial}{\partial x_\ell}\left(\rho v_k^R v_m^A + p\delta_{km}\right)\frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}}\end{aligned} \tag{3.1.4.58}$$

$$\begin{aligned}\mathcal{S}_1^{\text{ENERGY}} = & \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 \mathsf{n}_k \frac{\delta}{\delta b_i}\left(\rho h_t v_k^R\right) \mathrm{d}S} + \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_k^F \mathsf{n}_k \frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} \\ & \underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 \mathsf{n}_k \frac{\partial}{\partial x_\ell}\left(\rho h_t v_k^R + v_k^F p\right)\frac{\delta x_\ell}{\delta b_i} \mathrm{d}S + \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 \mathsf{n}_k p \frac{\partial v_k^F}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}}\end{aligned} \tag{3.1.4.59}$$

The $\mathcal{S}_2$ integrals are expanded as follows

$$\begin{aligned}\mathcal{S}_2^{\text{MOMENTUM}_m} = & -\cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{q+1} \mathsf{n}_q \tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell \frac{\delta\left(\mathsf{t}_m \mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}} + \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{q+1} \mathsf{n}_q \tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell \mathsf{t}_m \frac{\delta\left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\ & \underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \mathsf{n}_m \frac{\delta\left(\tau_{\ell k} \mathsf{n}_k \mathsf{n}_\ell\right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{(\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell)}} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \mathsf{t}_m \frac{\delta\left(\tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell\right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{(\tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell)}}\end{aligned}$$

$$
\begin{aligned}
&\underbrace{+ \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \mathsf{n}_k \frac{\partial \tau_{mk}}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S + \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \tau_{mk} \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\
&\underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} \tau_{\ell k} \mathsf{n}_k \mathsf{n}_\ell \frac{\delta \left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{q+1} \mathsf{t}_q \tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell \mathsf{t}_m \frac{\delta \left(\mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_1}
\end{aligned}
\tag{3.1.4.60}
$$

$$
\begin{aligned}
\mathcal{S}_2^{\text{ENERGY}} = &\cancel{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_q^A \mathsf{n}_q \tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell \frac{\delta \left(\mathsf{n}_m \mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}} + \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_q^A \mathsf{n}_q \tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell \mathsf{t}_m \frac{\delta \left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\
&\underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_m^A \mathsf{n}_m \frac{\delta \left(\tau_{\ell k} \mathsf{n}_k \mathsf{n}_\ell\right)}{\delta b_i}}_{\text{ABC}^{(\tau_{\ell k} \mathsf{n}_k \mathsf{n}_\ell)}} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_m^A \mathsf{t}_m \frac{\delta \left(\tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell\right)}{\delta b_i}}_{\text{ABC}^{(\tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell)}} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 \frac{\delta \left(q_k \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{(q_k \mathsf{n}_k)}} \\
&\underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_m^A \tau_{\ell k} \mathsf{n}_k \mathsf{n}_\ell \frac{\delta \left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i} + \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_m^A \tau_{mk} \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i} + \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 q_k \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\
&\underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 \mathsf{n}_k \tau_{mk} \frac{\partial v_m^F}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S + \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 \mathsf{n}_k \frac{\partial}{\partial x_\ell} \left(\tau_{mk} v_m^A + q_k\right) \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}} \\
&\underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_5 v_q^A \mathsf{t}_q \tau_{\ell k} \mathsf{n}_k \mathsf{t}_\ell \mathsf{t}_m \frac{\delta \left(\mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_2}
\end{aligned}
\tag{3.1.4.61}
$$

The $\mathcal{S}_{3,4}$ term leads to

$$
\mathcal{S}_{3,4} = \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tau_{km}^{\text{adj}} \mathsf{n}_k \frac{\partial v_m^F}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S - \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tau_{km}^{\text{adj}} \mathsf{n}_k \frac{\partial v_m^A}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}}
\tag{3.1.4.62}
$$

Finally, terms $\mathcal{S}_6$ and $\mathcal{S}_{11}$ are expanded as follows

$$\mathcal{S}_6 = \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tilde{\nu}_a \mathsf{n}_k \frac{\delta}{\delta b_i}\left(\rho\tilde{\nu} v_k^R\right) \mathrm{d}S} - \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tilde{\nu}_a \mathsf{n}_k \frac{\partial}{\partial x_\ell}\left(\rho\tilde{\nu} v_k^R\right)\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}} \tag{3.1.4.63}$$

$$\begin{aligned}\mathcal{S}_{11} = & \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_m^A}{\partial x_\ell}\mathsf{n}_q\frac{\partial v_r^F}{\partial x_p}\frac{\delta x_p}{\delta b_i}\mathrm{d}S}_{\text{SD}} \\ & \underbrace{- \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \rho\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_m^A}{\partial x_\ell}\mathsf{n}_q\frac{\partial v_r^A}{\partial x_p}\frac{\delta x_p}{\delta b_i}\mathrm{d}S}_{\text{SD}}\end{aligned} \tag{3.1.4.64}$$

The third integral of Eq. 3.3 for rotating no-slip wall boundaries is expressed as for the case of stationary no-slip walls, Eq. 3.1.4.48. Collecting and eliminating the ABC^p terms in Eqs. 3.1.4.36-3.1.4.39, Eq. 3.1.4.48 and Eqs. 3.1.4.57-3.1.4.64 the following relation arises

$$\Psi_{m+1}\mathsf{n}_m + \Psi_5 v_m^F \mathsf{n}_m - \frac{1}{\mathrm{Re}_0\,\sigma}\rho\tilde{\nu}_a\frac{\partial\tilde{\nu}}{\partial x_m}\mathsf{n}_m\mathcal{P}\left(\nu,\rho\right)\mathcal{P}\left(\rho,p\right) = \begin{cases} -\dfrac{\partial F_S}{\partial p} \text{ , at } S_{W^{\text{NoSlip,Rot}}}^{\text{Obj}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,Rot}}} \setminus S_{W^{\text{NoSlip,Rot}}}^{\text{Obj}} \end{cases} \tag{3.1.4.65}$$

Elimination of the ABC$^{(\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell)}$ and ABC$^{(\tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell)}$ terms leads to

$$\Psi_{m+1}\mathsf{n}_m + \Psi_5 v_m^F \mathsf{n}_m = \begin{cases} \dfrac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)} \text{ , at } S_{W^{\text{NoSlip,Rot}}}^{\text{Obj}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,Rot}}} \setminus S_{W^{\text{NoSlip,Rot}}}^{\text{Obj}} \end{cases} \tag{3.1.4.66}$$

$$\Psi_{m+1}\mathsf{t}_m + \Psi_5 v_m^F \mathsf{t}_m = \begin{cases} \dfrac{\partial F_s}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)} \text{ , at } S_{W^{\text{NoSlip,Rot}}}^{\text{Obj}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,Rot}}} \setminus S_{W^{\text{NoSlip,Rot}}}^{\text{Obj}} \end{cases} \tag{3.1.4.67}$$

Elimination of terms ABC$^{\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}$ leads to the same adjoint boundary condition for $\tilde{\nu}_a$ as for stationary no-slip walls (Eq. 3.1.4.53). Importing Eq. 3.1.4.53 into Eq. 3.1.4.65 and combining with Eq. 3.1.4.66, in order to obtain a single adjoint boundary condition for $\Psi_{m+1}\mathsf{n}_m$ the objective function expression on rotating no-slip wall boundaries must satisfy the following constraint

$$\frac{\partial F_S}{\partial\left(\tau_{\ell k}\mathsf{n}_k\mathsf{n}_\ell\right)} - \frac{1}{\nu}\frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}\frac{\partial \tilde{\nu}}{\partial x_m}\mathsf{n}_m \mathcal{P}\left(\nu,\rho\right)\mathcal{P}\left(\rho,p\right) + \frac{\partial F_S}{\partial p} = 0 \tag{3.1.4.68}$$

If, additionally, the objective function satisfies $\frac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)}\tau_{\ell k}\mathsf{n}_k\mathsf{t}_\ell\mathsf{t}_m + \frac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(\mathsf{t}_m\mathrm{d}S\right)} = 0$, then terms $\mathcal{T}_1$, $\mathcal{T}_2$ and term $\mathcal{T}_2$ of Eq. 3.1.4.48, expressed at the $S_{W^{\mathrm{NoSlip,Rot}}}$ boundaries, vanish automatically. Finally, the adjoint thermal boundary conditions for rotating no-slip walls are the same as for the $S_{W^{\mathrm{NoSlip,St}}}$ boundaries, as given by Eqs. 3.1.4.55, 3.1.4.56.

3.1.4.2 Inlet and Outlet Boundaries $S_{I/O}$

The flow conditions for the inlet and outlet boundaries are the ones described in Sections 2.2.2 and 2.2.3, respectively. A set of local flow quantities $(V_\ell^{\mathrm{loc}}, \ell = 1,\ldots,5)$ is defined at these boundaries. This set consists of the flow quantities for which a Dirichlet boundary condition is imposed and the flow quantities which are extrapolated from the flow domain.

The surface integrals that arise from the differentiation of the viscous terms $\mathcal{S}_2$ to $\mathcal{S}_4$ are neglected, by assuming that the total variation in the viscous stresses and heat flux is zero along the inlet and outlet.

In addition, inlet and outlet boundaries are considered unparameterized. Consequently, there is no variation in geometric quantities and no contribution in computing the sensitivity derivatives.

Inlet and outlet are split to subsonic and supersonic, inlet and outlet ones and the development of the surface terms is presented separately for each one of them, in the following paragraphs.

Subsonic Inlet Boundaries $S_{I^{\mathrm{sub}}}$

For subsonic inlet boundaries let $V_\ell^{\mathrm{loc}}, (\ell = 1,\ldots,4)$ be the quantities for which a Dirichlet condition is imposed and V_5^{loc} the flow quantity whose value is extrapolated from the interior domain. This leads to $\frac{\delta V_\ell^{\mathrm{loc}}}{\delta b_i} = 0, (\ell = 1,\ldots,4)$. Taking this into account and developing the $\mathcal{S}_1$ term the following terms arise

$$\mathcal{S}_1 = \cancel{\int\limits_{S_{I^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_1}\frac{\delta V^{\text{loc}}_1}{\delta b_i}\text{d}S} + \cancel{\int\limits_{S_{I^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_2}\frac{\delta V^{\text{loc}}_2}{\delta b_i}\text{d}S} + \cancel{\int\limits_{S_{I^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_3}\frac{\delta V^{\text{loc}}_3}{\delta b_i}\text{d}S}$$
$$+ \cancel{\int\limits_{S_{I^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_4}\frac{\delta V^{\text{loc}}_4}{\delta b_i}\text{d}S} + \underbrace{\int\limits_{S_{I^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_5}\frac{\delta V^{\text{loc}}_5}{\delta b_i}\text{d}S}_{\text{ABC}^{V^{\text{loc}}_5}} \qquad (3.1.4.69)$$

Eliminating term $\text{ABC}^{V^{\text{loc}}_5}$ leads to the following expression

$$\Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_5} = \begin{cases} -\dfrac{\partial F_S}{\partial V^{\text{loc}}_5}, \text{ at } S^{\text{Obj}}_{I^{\text{sub}}} \\ \qquad 0, \text{ at } S_{I^{\text{sub}}} \setminus S^{\text{Obj}}_{I^{\text{sub}}} \end{cases} \qquad (3.1.4.70)$$

When the adjoint flux is computed along subsonic inlet boundaries, Eq. 3.1.4.70 is taken into account and the boundary flux is modified accordingly. Since a Dirichlet boundary condition is imposed on $\tilde{\nu}$ for inlet boundaries, terms $\mathcal{S}_6$, $\mathcal{S}_8$ and $\mathcal{S}_{10}$ vanish automatically. The remaining terms are eliminated by setting $\tilde{\nu}_a = 0$.

Supersonic Inlet Boundaries $S_{I^{\text{sup}}}$

For supersonic inlet boundaries, all $V^{\text{loc}}_\ell, (\ell = 1, \dots, 5)$ quantities are constant and their values set by the user. As a result, term $\mathcal{S}_1$ vanishes and the adjoint flux is computed by extrapolating quantities from the flow domain. The adjoint boundary condition for $\tilde{\nu}_a$ is the same as for a subsonic inlet.

Subsonic Outlet Boundaries $S_{O^{\text{sub}}}$

For a subsonic outlet, let $V^{\text{loc}}_\ell, (\ell = 1, \dots, 4)$ be the quantities that are extrapolated from the flow domain and V^{loc}_5 the quantity for which a Dirichlet condition is set. As a result $\frac{\delta V^{\text{loc}}_5}{\delta b_i} = 0$. Taking this into account, term $\mathcal{S}_1$ is written as

$$\mathcal{S}_1 = \underbrace{\int\limits_{S_{O^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_1}\frac{\delta V^{\text{loc}}_1}{\delta b_i}\text{d}S}_{\text{ABC}^{V^{\text{loc}}_1}} + \underbrace{\int\limits_{S_{O^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_1}\frac{\delta V^{\text{loc}}_2}{\delta b_i}\text{d}S}_{\text{ABC}^{V^{\text{loc}}_2}} + \underbrace{\int\limits_{S_{O^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V^{\text{loc}}_1}\frac{\delta V^{\text{loc}}_3}{\delta b_i}\text{d}S}_{\text{ABC}^{V^{\text{loc}}_3}}$$

$$+\underbrace{\int\limits_{S_{O^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V_1^{\text{loc}}}\frac{\delta V_4^{\text{loc}}}{\delta b_i}\mathrm{d}S}_{\text{ABC}^{V_4^{\text{loc}}}} + \cancel{\int\limits_{S_{O^{\text{sub}}}} \Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V_5^{\text{loc}}}\frac{\delta V_5^{\text{loc}}}{\delta b_i}\mathrm{d}S} \qquad (3.1.4.71)$$

Eliminating the $\text{ABC}^{V_\ell^{\text{loc}}}$, $(\ell = 1, \ldots, 4)$ terms, the following conditions result

$$\Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m}\frac{\partial U_m}{\partial V_\ell^{\text{loc}}} = \begin{cases} -\dfrac{\partial F_S}{\partial V_\ell^{\text{loc}}} \text{ , at } S^{\text{Obj}}_{O^{\text{sub}}} \\ 0 \text{ , at } S_{O^{\text{sub}}} \setminus S^{\text{Obj}}_{O^{\text{sub}}} \end{cases} \quad \text{for } \ell = 1, \ldots, 4 \qquad (3.1.4.72)$$

Eq. 3.1.4.72 is taken into account in the computation of the adjoint boundary flux along a subsonic outlet. A zero Neumann condition is imposed on $\tilde{\nu}$, which means that $\dfrac{\delta}{\delta b_i}\left(\dfrac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right) = 0$. The remaining integrals from the development of terms $\mathcal{S}_6$ to $\mathcal{S}_{11}$ are eliminated by setting the multiplier of $\frac{\delta \tilde{\nu}}{\delta b_i}$ equal to $\frac{\partial F_S}{\partial \tilde{\nu}}$ at $S^{\text{Obj}}_{O^{\text{sub}}}$ and equal to zero at $S_{O^{\text{sub}}} \setminus S^{\text{Obj}}_{O^{\text{sub}}}$.

Supersonic Outlet Boundaries $S_{O^{\text{sup}}}$

Along a supersonic outlet $(S_{O^{\text{sup}}})$, the process of deriving the adjoint boundary conditions is the same as for a subsonic one. The only difference is that all flow quantities are extrapolated from the flow domain and, consequently, Eq. 3.1.4.72 holds for $\ell = 1, \ldots, 5$.

3.2 The Continuous Adjoint Method - FI Formulation

As for the SI formulation (Section 3.1), instead of directly computing the sensitivity derivatives of F, F_{aug} is introduced which is repeated here for the sake of completeness,

$$F_{\text{aug}} = F + \int\limits_{\Omega} \Psi_n R_n \mathrm{d}\Omega + \int\limits_{\Omega} \tilde{\nu}_a R_{\tilde{\mu}} \mathrm{d}\Omega$$

In contrast to the SI approach, instead of using the Leibniz rule, the volume integrals of F_{aug} are differentiated by directly passing the $\frac{\delta}{\delta b_i}$ operator inside the integrals. Since these integrals contain spatial gradients of flow related quantities (e.g. inviscid and viscous fluxes), a useful relation of treating variations of such quantities is sought [143, 96]. Using Eq. 3.2 and computing the spatial gradient of the total variation of an arbitrary flow related quantity Φ one obtains

$$\frac{\partial}{\partial x_\ell}\left(\frac{\delta \Phi}{\delta b_i}\right) = \frac{\partial}{\partial x_\ell}\left(\frac{\partial \Phi}{\partial b_i}\right) + \frac{\partial^2 \Phi}{\partial x_k \partial x_\ell}\frac{\delta x_k}{\delta b_i} + \frac{\partial \Phi}{\partial x_k}\frac{\partial}{\partial x_\ell}\left(\frac{\delta x_k}{\delta b_i}\right) \tag{3.2.1}$$

From Eq. 3.2 one can also get an expression for the total derivative of the spatial gradient of Φ as

$$\frac{\delta}{\delta b_i}\left(\frac{\partial \Phi}{\partial x_\ell}\right) = \frac{\partial}{\partial b_i}\left(\frac{\partial \Phi}{\partial x_\ell}\right) + \frac{\partial^2 \Phi}{\partial x_k \partial x_\ell}\frac{\delta x_k}{\delta b_i} \tag{3.2.2}$$

Subtracting Eq. 3.2.1 from Eq. 3.2.2, the total variation of the gradient of Φ is linked to the gradient of the total variation of Φ through

$$\frac{\delta}{\delta b_i}\left(\frac{\partial \Phi}{\partial x_\ell}\right) = \frac{\partial}{\partial x_\ell}\left(\frac{\delta \Phi}{\delta b_i}\right) - \frac{\partial \Phi}{\partial x_k}\frac{\partial}{\partial x_\ell}\left(\frac{\delta x_k}{\delta b_i}\right) \tag{3.2.3}$$

Eq. 3.2.3 is extensively used in the remainder of this section, in which the FI continuous adjoint is presented.

Differentiation of F_{aug} results to

$$\frac{\delta F_{\text{aug}}}{\delta b_i} = \frac{\delta F}{\delta b_i} + \underbrace{\int\limits_\Omega \Psi_n \frac{\delta R_n}{\delta b_i} d\Omega}_{\mathcal{I}^{\text{MF}}} + \underbrace{\int\limits_\Omega \tilde{\nu}_a \frac{\delta R_{\tilde{\mu}}}{\delta b_i} d\Omega}_{\mathcal{I}^{\text{SA}}} \tag{3.2.4}$$

3.2.1 Differentiation of the Mean Flow Equations

Term $\mathcal{I}^{\text{MF}}$ is expanded first as

$$\mathcal{I}^{\text{MF}} = \underbrace{\int\limits_\Omega \Psi_n \frac{\delta}{\delta b_i}\left(\frac{\partial f_{nk}^{inv}}{\partial x_k}\right) d\Omega}_{\mathcal{I}^{\text{MF\_inv}}} - \underbrace{\int\limits_\Omega \Psi_n \frac{\delta}{\delta b_i}\left(\frac{\partial f_{nk}^{vis}}{\partial x_k}\right) d\Omega}_{\mathcal{I}^{\text{MF\_vis}}} + \underbrace{\int\limits_\Omega \Psi_n \frac{\delta S_n}{\delta b_i} d\Omega}_{\mathcal{I}^{\text{MF\_src}}} \tag{3.2.1.1}$$

Employing Eq. 3.2.3 on the $\mathcal{I}^{\text{MF\_inv}}$ term we obtain

$$\mathcal{I}^{\text{MF\_inv}} = \int\limits_\Omega \Psi_n \frac{\partial}{\partial x_k}\left(\frac{\delta f_{nk}^{inv}}{\delta b_i}\right) d\Omega - \underbrace{\int\limits_\Omega \Psi_n \frac{\partial f_{nk}^{inv}}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right) d\Omega}_{\text{VSD}} \tag{3.2.1.2}$$

The second term on the r.h.s.of Eq. 3.2.1.2 contains only variations of geometric quantities and, since it is a volume integral, it is marked as volume sensitivity derivative (VSD). All terms denoted as VSD constitute the first of the two main differences between the FI and the SI continuous adjoint formulations, the second being the absence of the Leibniz terms in the FI one. The computation of the VSD terms requires the differentiation of the mesh morphing technique that is employed to deform the CFD mesh in each optimization cycle. This differentiation is done either analytically, if this is permitted by the mesh morphing method, or through finite differences (if a PDE is solved for deforming the mesh). A novel approach has also been developed by PCOpt/LTT where the mesh morphing PDE is included in the adjoint formulation. In this approach, the residual of the mesh morphing PDE multiplied with the so-called adjoint deformation field is added in the expression of the augmented function. This way, the computation of grid sensitivities ($\frac{\delta x_k}{\delta b_i}$) of internal mesh nodes is avoided and the sensitivity derivatives are computed based only on surface integrals. This approach is called the Enhanced Surface Integral (E-SI) , opposed to the severed-SI where the Leibniz term is neglected, and was developed in [96]. In the scope of this thesis the E-SI approach will not be used and any further reference to the SI method will implicitly correspond to the severed-SI one.

The first term of Eq. 3.2.1.2 is further developed as

$$\int\limits_{\Omega} \Psi_n \frac{\partial}{\partial x_k}\left(\frac{\delta f_{nk}^{inv}}{\delta b_i}\right) \mathrm{d}\Omega = \underbrace{\int\limits_{\partial\Omega} \Psi_n \mathsf{n}_k \frac{\delta f_{nk}^{inv}}{\delta b_i} \mathrm{d}S}_{\mathcal{S}_1} \underbrace{- \int\limits_{\Omega} A_{nmk} \frac{\partial \Psi_n}{\partial x_k} \frac{\delta U_m}{\delta b_i} \mathrm{d}\Omega}_{\text{FAE\_MF}} \tag{3.2.1.3}$$

Similarly, term $\mathcal{I}^{\text{MF\_vis}}$ results in

$$\begin{aligned} \mathcal{I}^{\text{MF\_vis}} &= \underbrace{-\int\limits_{\partial\Omega} \Psi_n \mathsf{n}_k \frac{\delta f_{nk}^{vis}}{\delta b_i} \mathrm{d}S}_{\mathcal{S}_2} + \int\limits_{\Omega} \frac{\partial \Psi_n}{\partial x_k} \frac{\delta f_{nk}^{vis}}{\delta b_i} \mathrm{d}\Omega \\ &= \mathcal{S}_2 + \underbrace{\int\limits_{\Omega} \frac{\partial \Psi_{m+1}}{\partial x_k} \frac{\delta \tau_{km}}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MomVis}}} + \underbrace{\int\limits_{\Omega} \frac{\partial \Psi_5}{\partial x_k} \frac{\delta\left(v_m^A \tau_{km}\right)}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{EnerVis1}}} + \underbrace{\int\limits_{\Omega} \frac{\partial \Psi_5}{\partial x_k} \frac{\delta q_k}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{EnerVis1}}} \end{aligned} \tag{3.2.1.4}$$

with

$$\mathcal{I}^{\text{MomVis}} = \underbrace{-\int\limits_{\Omega} \frac{\partial}{\partial x_k}\left[\frac{\mu+\mu_t}{\text{Re}_0}\left(\frac{\partial \Psi_{m+1}}{\partial x_k} + \frac{\partial \Psi_{k+1}}{\partial x_m} - \frac{2}{3}\delta_{km}\frac{\partial \Psi_{\ell+1}}{\partial x_\ell}\right)\right] \frac{\partial v_m^A}{\partial U_\ell} \frac{\delta U_\ell}{\delta b_i} \mathrm{d}\Omega}_{\text{FAE\_MF}}$$

$$
\begin{aligned}
&\underbrace{+\int\limits_{\Omega}\frac{\partial\Psi_{m+1}}{\partial x_k}\frac{\tau_{km}}{\mu+\mu_t}\left[\mathcal{P}\left(\mu_t,\rho\right)+\mathcal{P}\left(\mu_t,f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\rho\right)\right]\frac{\delta\rho}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\\
&\underbrace{+\int\limits_{\Omega}\frac{\partial\Psi_{m+1}}{\partial x_k}\frac{\tau_{km}}{\mu+\mu_t}\left[\mathcal{P}\left(\mu_t,\tilde{\nu}\right)+\mathcal{P}\left(\mu_t,f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\mathcal{P}\left(\chi,\tilde{\nu}\right)\right]\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}\\
&\underbrace{+\int\limits_{\Omega}\frac{\partial\Psi_{m+1}}{\partial x_k}\frac{\tau_{km}}{\mu+\mu_t}\left[1+\mathcal{P}\left(\mu_t,f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\mu\right)\right]\frac{\partial\mu}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\mathrm{d}\Omega}_{Suth}\\
&\underbrace{-\int\limits_{\Omega}\frac{\mu+\mu_t}{\mathrm{Re}_0}\left(\frac{\partial\Psi_{m+1}}{\partial x_k}+\frac{\partial\Psi_{k+1}}{\partial x_m}-\frac{2}{3}\delta_{km}\frac{\partial\Psi_{\ell+1}}{\partial x_\ell}\right)\frac{\partial v_m^A}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}}\\
&\underbrace{+\int\limits_{\partial\Omega}\frac{\mu+\mu_t}{\mathrm{Re}_0}\left(\frac{\partial\Psi_{m+1}}{\partial x_k}+\frac{\partial\Psi_{k+1}}{\partial x_m}-\frac{2}{3}\delta_{km}\frac{\partial\Psi_{\ell+1}}{\partial x_\ell}\right)\mathsf{n}_k\frac{\delta v^A}{\delta b_i}\mathrm{d}S}_{\mathcal{S}_3} \qquad (3.2.1.5)
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{I}^{\text{EnerVis1}} =& \underbrace{\int\limits_{\Omega}\frac{\partial\Psi_5}{\partial x_k}\tau_{km}\frac{\delta v_m^A}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\underbrace{-\int\limits_{\partial\Omega}\frac{\mu+\mu_t}{\mathrm{Re}_0}\left(\frac{\partial\Psi_5}{\partial x_k}v_m^A+\frac{\partial\Psi_5}{\partial x_m}v_k^A-\frac{2}{3}\delta_{km}\frac{\partial\Psi_5}{\partial x_\ell}v_\ell^A\right)\mathsf{n}_k\frac{\delta v_m^A}{\delta b_i}\mathrm{d}S}_{\mathcal{S}_4}\\
&\underbrace{-\int\limits_{\Omega}\frac{\partial}{\partial x_k}\left[\frac{\mu+\mu_t}{\mathrm{Re}_0}\left(\frac{\partial\Psi_5}{\partial x+k}v_m^A+\frac{\partial\Psi_5}{\partial x_m}v_k^A-\frac{2}{3}\delta_{km}\frac{\partial\Psi_5}{\partial x_\ell}v_\ell^A\right)\right]\frac{\partial v_m^A}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\\
&\underbrace{+\int\limits_{\Omega}\frac{\partial\Psi_5}{\partial x_k}v_m^A\frac{\tau_{km}}{\mu+\mu_t}\left[\mathcal{P}\left(\mu_t,\rho\right)+\mathcal{P}\left(\mu_t,f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\rho\right)\right]\frac{\delta\rho}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}\\
&\underbrace{+\int\limits_{\Omega}\frac{\partial\Psi_5}{\partial x_k}v_m^A\frac{\tau_{km}}{\mu+\mu_t}\left[\mathcal{P}\left(\mu_t,\tilde{\nu}\right)+\mathcal{P}\left(\mu_t,f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\mathcal{P}\left(\chi,\tilde{\nu}\right)\right]\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}\\
&\underbrace{+\int\limits_{\Omega}\frac{\partial\Psi_5}{\partial x_k}v_m^A\frac{\tau_{km}}{\mu+\mu_t}\left[1+\mathcal{P}\left(\mu_t,f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\mu\right)\right]\frac{\partial\mu}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\mathrm{d}\Omega}_{Suth}
\end{aligned}
$$

$$\underbrace{-\int_{\Omega}\frac{\mu+\mu_t}{\mathrm{Re}_0}\left(\frac{\partial\Psi_5}{\partial x_k}v_m^A+\frac{\partial\Psi_5}{\partial x_m}v_k^A-\frac{2}{3}\delta_{km}\frac{\partial\Psi_5}{\partial x_\ell}v_\ell^A\right)\frac{\partial v_m^A}{\partial x_q}\frac{\partial}{\partial x_k}\left(\frac{\delta x_q}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}} \tag{3.2.1.6}$$

$$\begin{aligned}
\mathcal{I}^{\text{EnerVis2}} = & \underbrace{-\int_{\Omega}\frac{\partial}{\partial x_k}\left[\frac{\mathcal{C}_p}{\mathrm{Re}_0}\left(\frac{\mu}{\mathrm{Pr}}+\frac{\mu_t}{\mathrm{Pr_t}}\right)\frac{\partial\Psi_5}{\partial x_k}\right]\frac{\partial T}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}} \\
& \underbrace{+\int_{\Omega}\frac{\mathcal{C}_p}{\mathrm{Re}_0}\frac{\partial\Psi_5}{\partial x_k}\frac{\partial T}{\partial x_k}\frac{1}{\mathrm{Pr_t}}\left[\mathcal{P}(\mu_t,\rho)+\mathcal{P}(\mu_t,f_{v_1})\mathcal{P}(f_{v_1},\chi)\mathcal{P}(\chi,\nu)\mathcal{P}(\nu,\rho)\right]\frac{\delta\rho}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}} \\
& \underbrace{+\int_{\Omega}\frac{\mathcal{C}_p}{\mathrm{Re}_0}\frac{\partial\Psi_5}{\partial x_k}\frac{\partial T}{\partial x_k}\frac{1}{\mathrm{Pr_t}}\left[\mathcal{P}(\mu_t,\tilde{\nu})+\mathcal{P}(\mu_t,f_{v_1})\mathcal{P}(f_{v_1},\chi)\mathcal{P}(\chi,\tilde{\nu})\right]\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}} \\
& \underbrace{+\int_{\Omega}\frac{\mathcal{C}_p}{\mathrm{Re}_0}\frac{\partial\Psi_5}{\partial x_k}\frac{\partial T}{\partial x_k}\left[\frac{1}{\mathrm{Pr}}+\frac{1}{\mathrm{Pr_t}}\mathcal{P}(\mu_t,f_{v_1})\mathcal{P}(f_{v_1},\chi)\mathcal{P}(\chi,\nu)\mathcal{P}(\nu,\mu)\right]\frac{\partial\mu}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\mathrm{d}\Omega}_{Suth} \\
& \underbrace{-\int_{\Omega}\frac{\mathcal{C}_p}{\mathrm{Re}_0}\frac{\partial\Psi_5}{\partial x_k}\left(\frac{\mu}{\mathrm{Pr}}+\frac{\mu_t}{\mathrm{Pr_t}}\right)\frac{\partial T}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}}\underbrace{+\int_{\partial\Omega}\frac{\mathcal{C}_p}{\mathrm{Re}_0}\frac{\partial\Psi_5}{\partial x_k}\left(\frac{\mu}{\mathrm{Pr}}+\frac{\mu_t}{\mathrm{Pr_t}}\right)\mathsf{n}_k\frac{\delta T}{\delta b_i}\mathrm{d}S}_{\text{S}_5}
\end{aligned} \tag{3.2.1.7}$$

Finally, term $\mathcal{I}^{\text{MF\_src}}$ leads to

$$\mathcal{I}^{\text{MF\_src}} = \underbrace{\int_{\Omega}\varepsilon_{m\ell k}\Psi_{m+1}\rho\omega_\ell\frac{\partial v_k^A}{\partial U_q}\frac{\delta U_q}{\delta b_i}\mathrm{d}\Omega+\int_{\Omega}\varepsilon_{m\ell k}\Psi_{m+1}\omega_\ell v_k^A\frac{\partial\rho}{\partial U_q}\frac{\delta U_q}{\delta b_i}}_{\text{FAE\_MF}} \tag{3.2.1.8}$$

3.2.2 Differentiation of the Spalart–Allmaras Equation

The term $\mathcal{I}^{\text{SA}}$ in Eq. 3.2.4 is split in three terms arising from the differentiation of the convection, diffusion and source terms of the Spalart–Allmaras PDE,

$$\mathcal{I}^{\text{SA}}=\underbrace{\int_\Omega \tilde{\nu}_a \frac{\delta \text{SA}^c}{\delta b_i}\text{d}\Omega}_{\mathcal{I}^{\text{SA\_conv}}}+\underbrace{\int_\Omega \tilde{\nu}_a \frac{\delta \text{SA}^d}{\delta b_i}\text{d}\Omega}_{\mathcal{I}^{\text{SA\_diff}}}+\underbrace{\int_\Omega \tilde{\nu}_a \frac{\delta \text{SA}^s}{\delta b_i}\text{d}\Omega}_{\mathcal{I}^{\text{SA\_src}}}$$

Convection term ($\mathcal{I}^{\text{SA\_conv}}$) can be developed as

$$\begin{aligned}\mathcal{I}^{\text{SA\_conv}}=&\underbrace{-\int_\Omega \frac{\partial \tilde{\nu}_a}{\partial x_k}\tilde{\nu} v_k^R \frac{\partial \rho}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega-\int_\Omega \frac{\partial \tilde{\nu}_a}{\partial x_k}\rho\tilde{\nu}\frac{\partial v_k^A}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}}\\ &\underbrace{-\int_\Omega \frac{\partial \tilde{\nu}_a}{\partial x_k}\rho v_k^R \frac{\delta \tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}}+\underbrace{\int_{\partial\Omega}\tilde{\nu}_a \mathsf{n}_k \frac{\delta}{\delta b_i}\left(\rho\tilde{\nu}v_k^R\right)\text{d}S}_{\mathcal{S}_6}\\ &\underbrace{-\int_\Omega \tilde{\nu}_a \frac{\partial\left(\rho\tilde{\nu}v_k^R\right)}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\text{d}\Omega+\int_\Omega \frac{\partial \tilde{\nu}_a}{\partial x_k}\rho\tilde{\nu}\frac{\partial v_k^F}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\text{d}\Omega}_{\text{VSD}}\end{aligned} \qquad (3.2.2.1)$$

The diffusion term is treated similarly

$$\begin{aligned}\mathcal{I}^{\text{SA\_diff}}=&\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int_\Omega \tilde{\nu}_a\left\{\frac{\partial}{\partial x_k}\left[\left(\nu+\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial x_k}\right]+c_{b_2}\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\frac{\partial \rho}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}}\\ &\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int_\Omega \rho\tilde{\nu}_a\frac{\delta}{\delta b_i}\left(\frac{\partial}{\partial x_k}\left\{\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\right)\text{d}\Omega}_{\mathcal{I}^{\text{SA\_diff1}}}+\underbrace{\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int_\Omega \rho\tilde{\nu}_a\frac{\delta}{\delta b_i}\left(\tilde{\nu}\frac{\partial^2\tilde{\nu}}{\partial x_k^2}\right)\text{d}\Omega}_{\mathcal{I}^{\text{SA\_diff2}}}\end{aligned}$$

(3.2.2.2)

where

$$\mathcal{I}^{\text{SA\_diff1}}=\underbrace{\frac{1}{\text{Re}_0\,\sigma}\int_\Omega \frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k}\mathcal{P}\left(\nu,\rho\right)\frac{\partial \rho}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}}+\underbrace{\frac{1}{\text{Re}_0\,\sigma}\int_\Omega \frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k}\mathcal{P}\left(\nu,\mu\right)\frac{\partial \mu}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega}_{\textit{Suth}}$$

$$\underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k}\left(1+c_{b_2}\right)\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial}{\partial x_k}\left\{\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\right\}\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}}$$
$$\underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\rho\tilde{\nu}_a\frac{\partial}{\partial x_\ell}\left\{\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\text{d}\Omega}_{\text{VSD}}$$
$$\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\text{d}\Omega}_{\text{VSD}}$$
$$\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\rho\tilde{\nu}_a\mathsf{n}_k\frac{\delta}{\delta b_i}\left\{\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\text{d}S}_{\mathcal{S}_7}$$
$$\underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\frac{\partial\left(\rho\tilde{\nu}_a\right)}{\partial x_k}\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\mathsf{n}_k\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}S}_{\mathcal{S}_8} \tag{3.2.2.3}$$

$$\mathcal{I}^{\text{SA\_diff2}}=\underbrace{\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\rho\tilde{\nu}_a\frac{\partial^2\tilde{\nu}}{\partial x_k^2}\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial}{\partial x_k}\left[\frac{\partial\left(\rho\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\right]\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}}$$
$$\underbrace{-\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\rho\tilde{\nu}_a\tilde{\nu}\frac{\partial}{\partial x_\ell}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\text{d}\Omega+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial\left(\rho\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\text{d}\Omega}_{\text{VSD}}$$
$$\underbrace{+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\rho\tilde{\nu}_a\tilde{\nu}\mathsf{n}_k\frac{\delta}{\delta b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)\text{d}S}_{\mathcal{S}_9}\underbrace{-\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\frac{\partial\left(\rho\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\mathsf{n}_k\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}S}_{\mathcal{S}_{10}} \tag{3.2.2.4}$$

For the differentiation of the turbulence model equation source terms, Eqs. 3.1.2.2-3.1.2.11 are used with the only difference that all direct variations $\left(\frac{\partial}{\partial b_i}\right)$ are replaced by the total variations of the same quantity $\left(\frac{\delta}{\delta b_i}\right)$. Then, the differentiation of the source terms leads to

$$\begin{aligned}
\mathcal{I}^{\text{SA\_src}} = & \underbrace{\int\limits_{\Omega}\tilde{\nu}_a\left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{S}\tilde{\nu}+\frac{1}{\text{Re}_0}\left(c_{w_1}f_w-\frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\Delta}\right)^2\right]\frac{\partial\rho}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}} \\
& \underbrace{+\int\limits_{\Omega}\rho\tilde{\nu}_a\left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{S}+\frac{2}{\text{Re}_0}\left(c_{w_1}f_w-\frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\Delta^2}\right)\right]\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}} \\
& \underbrace{-\frac{2}{\text{Re}_0}\int\limits_{\Omega}\rho\tilde{\nu}_a\left(c_{w_1}f_w-\frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\frac{\tilde{\nu}^2}{\Delta^3}\frac{\delta\Delta}{\delta b_i}\text{d}\Omega+\frac{c_{w_1}}{\text{Re}_0}\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_5\frac{\delta\Delta}{\delta b_i}\text{d}\Omega}_{\text{FAE\_DISTANCE}} \\
& \underbrace{+\frac{c_{w_1}}{\text{Re}_0}\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_4\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}}\underbrace{+\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\frac{\delta\tilde{S}}{\delta b_i}\text{d}\Omega}_{\mathcal{I}^{\tilde{S}}}\underbrace{+\int\limits_{\Omega}\rho\tilde{\nu}_a\left\{c_{b_1}\left[\tilde{S}\tilde{\nu}-\frac{1}{\text{Re}_0\,\kappa^2}\left(\frac{\tilde{\nu}}{\Delta}\right)^2\right]\right\}\frac{\delta f_{t_2}}{\delta b_i}\text{d}\Omega}_{\mathcal{I}^{f_{t_2}}}
\end{aligned} \tag{3.2.2.5}$$

Term $\mathcal{I}^{\tilde{S}}$ leads to

$$\begin{aligned}
\mathcal{I}^{\tilde{S}} = & \underbrace{\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{\delta S}{\delta b_i}\text{d}\Omega}_{\mathcal{I}^{\text{VORTICITY}}}\underbrace{+\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\left[\mathcal{P}\left(\tilde{S},\tilde{\nu}\right)+\mathcal{C}_1\mathcal{P}\left(\chi,\tilde{\nu}\right)\right]\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}} \\
& \underbrace{+\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},\Delta\right)\frac{\delta\Delta}{\delta b_i}\text{d}\Omega}_{\text{FAE\_DISTANCE}}\underbrace{+\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{C}_1\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\rho\right)\frac{\partial\rho}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}} \\
& \underbrace{+\int\limits_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{C}_1\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\mu\right)\frac{\partial\mu}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\text{d}\Omega}_{\textit{Suth}}
\end{aligned} \tag{3.2.2.6}$$

with

$$\mathcal{I}^{\text{VORTICITY}} = \underbrace{-\int\limits_{\Omega}\frac{\partial}{\partial x_\ell}\left[\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\right]\frac{\partial v_m^A}{\partial U_p}\frac{\delta U_p}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}}$$

$$\underbrace{-\int_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\frac{\partial v_m^A}{\partial x_p}\frac{\partial}{\partial x_\ell}\left(\frac{\delta x_p}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}}$$
$$\underbrace{+\int_{\partial\Omega}\rho\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell\frac{\delta v_m^A}{\delta b_i}\mathrm{d}S}_{\mathcal{S}_{11}} \qquad (3.2.2.7)$$

Finally, term $\mathcal{I}^{f_{t_2}}$ expands to

$$\mathcal{I}^{f_{t_2}} = \underbrace{\int_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_7\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\rho\right)\frac{\partial\rho}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_MF}}+\underbrace{\int_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_7\mathcal{P}\left(\chi,\tilde{\nu}\right)\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}$$
$$\underbrace{+\int_{\Omega}\rho\tilde{\nu}_a\mathcal{C}_7\mathcal{P}\left(\chi,\nu\right)\mathcal{P}\left(\nu,\mu\right)\frac{\partial\mu}{\partial U_\ell}\frac{\delta U_\ell}{\delta b_i}\mathrm{d}\Omega} \qquad (3.2.2.8)$$

3.2.3 Field Adjoint Equations and Adjoint Boundary Conditions

Eliminating all volume integrals, denoted as FAE_MF, leads to the mean-flow field adjoint equations, while eliminating the FAE_SA integrals gives rise to the adjoint Spalart–Allmaras equation. Terms denoted as *Suth* contribute to the mean flow field adjoint equations if Sutherland's law is used to derive the fluid's dynamic viscosity. Finally, the FAE_DISTANCE terms are either computed directly by the parameterization and mesh morphing techniques or are included into the adjoint distance equation [144]. It can be observed that the adjoint equations derived are the same as the ones derived in the SI approach.

Similarly, terms $\mathcal{S}_1$–$\mathcal{S}_{11}$ are the same with the only difference being that instead of the partial derivative of flow quantities $\left(\frac{\partial}{\partial b_i}\right)$ the total variation appears $\left(\frac{\delta}{\delta b_i}\right)$. Consequently, eliminating these terms leads to the same adjoint boundary conditions as the ones developed in Section 3.1.4.

However, the surface SD terms that arise (and contribute to the expression of the sensitivity derivatives) are different, since the terms arising from the application of Eq. 3.2 on surface terms, in the SI approach, are absent in the FI one. Finally, all the VSD terms are volume integrals contributing to the final expression of the sensitivity derivatives. Further details on how the variation of the mesh nodes positions $\left(\frac{\delta x_\ell}{\delta b_i}\right)$ and their spatial gradient $\left(\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\right)$ are computed are given in the next chapters where the parameterization and mesh morphing techniques are presented.

3.3 The Continuous Adjoint Method for Incompressible Fluid Flows - FI Formulation

The governing equations in cases of incompressible flows are described in Section 2.5. In this case F_{aug} is expressed as

$$F_{\text{aug}} = F + \int\limits_{\Omega} \Psi_n R_n \mathrm{d}\Omega + \int\limits_{\Omega} \tilde{\nu}_a R_{\tilde{\nu}} \mathrm{d}\Omega \tag{3.3.1}$$

where $n = 1, \ldots, 4$ and the flow variables are $U = \left[p\ v_1^A\ v_2^A\ v_3^A\right]^T$ with p denoting the kinematic pressure (pressure divided by the constant density). It is repeated, herein, as a reminder, that the unknown velocity components are the absolute ones (observed in an inertial reference frame), even though the governing equations are expressed and solved w.r.t. a relative frame of reference.

In what follows, the FI adjoint approach is employed and so Eqs. 3.2, 3.3 as well as Eq. 3.2.3 are used extensively. By differentiating Eq. 3.3.1 in order to derive the field adjoint equations, boundary conditions and sensitivity derivatives, we obtain

$$\frac{\delta F_{\text{aug}}}{\delta b_i} = \frac{\delta F}{\delta b_i} + \underbrace{\int\limits_{\Omega} \Psi_n \frac{\delta R_n}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF}}} + \underbrace{\int\limits_{\Omega} \tilde{\nu}_a \frac{\delta R_{\tilde{\nu}}}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{SA}}} \tag{3.3.2}$$

Each term of Eq. 3.3.2 is separately developed in the following sections.

3.3.1 Differentiation of the Mean Flow Equations

The mean flow equations in the form of Eq. 2.5.1, before applying the artificial compressibility method, are differentiated. Their differentiation leads to

$$\mathcal{I}^{\text{MF}} = \underbrace{\int\limits_{\Omega} \Psi_n \frac{\delta f_{nk}^{inv}}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF\_inv}}} - \underbrace{\int\limits_{\Omega} \Psi_n \frac{\delta f_{nk}^{vis}}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF\_vis}}} + \underbrace{\int\limits_{\Omega} \Psi_n \frac{\delta S_n}{\delta b_i} \mathrm{d}\Omega}_{\mathcal{I}^{\text{MF\_src}}} \tag{3.3.1.1}$$

The term $\mathcal{I}^{\text{MF\_inv}}$ is fully expanded since, for incompressible flows, we cannot take advantage of the identity $f_{nk}^{inv} = A_{nmk} U_m$. The reason for this is that the incompressible flow equations (or more precisely their inviscid part) do not form a homogeneous system of equations of 1st degree (as is the case of compressible fluid flow equations). As a result term $\mathcal{I}^{\text{MF\_inv}}$ is split as

$$\mathcal{I}^{\text{MF\_inv}} = \underbrace{\int_{\Omega} \Psi_1 \frac{\delta f_{1k}^{inv}}{\delta b_i} \text{d}\Omega}_{\mathcal{I}^{\text{MF\_inv,CONTINUITY}}} + \underbrace{\int_{\Omega} \Psi_{m+1} \frac{\delta f_{(m+1)k}^{inv}}{\delta b_i} \text{d}\Omega}_{\mathcal{I}^{\text{MF\_inv,MOMENTUM}_m}} \tag{3.3.1.2}$$

The first term in Eq. 3.3.1.2 is expanded as

$$\begin{aligned}\mathcal{I}^{\text{MF\_inv,CONTINUITY}} = & \underbrace{-\int_{\Omega} \frac{\partial \Psi_1}{\partial x_k} \frac{\delta v_k^A}{\delta b_i} \text{d}\Omega}_{\text{FAE\_MF}} + \underbrace{\int_{\Omega} \frac{\partial \Psi_1}{\partial x_k} \frac{\partial v_k^F}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \text{d}\Omega}_{\text{VSD}} \\ & \underbrace{-\int_{\Omega} \Psi_1 \frac{\partial v_k^R}{\partial x_\ell} \frac{\partial}{\partial x_k} \left(\frac{\delta x_\ell}{\delta b_i} \right) \text{d}\Omega}_{\text{VSD}} + \underbrace{\int_{\partial\Omega} \Psi_1 \mathsf{n}_k \frac{\delta v_k^R}{\delta b_i} \text{d}S}_{\mathcal{S}_1^{\text{INCO}}}\end{aligned} \tag{3.3.1.3}$$

Similarly, term $\mathcal{I}^{\text{MF\_inv,MOMENTUM}_m}$ is expanded as

$$\begin{aligned}\mathcal{I}^{\text{MF\_inv,MOMENTUM}_m} = & \underbrace{-\int_{\Omega} \frac{\partial \Psi_{m+1}}{\partial x_k} v_m^A \frac{\delta v_k^A}{\delta b_i} \text{d}\Omega - \int_{\Omega} \frac{\partial \Psi_{m+1}}{\partial x_k} v_k^R \frac{\delta v_m^A}{\delta b_i} \text{d}\Omega - \int_{\Omega} \frac{\partial \Psi_{m+1}}{\partial x_m} \frac{\delta p}{\delta b_i} \text{d}\Omega}_{\text{FAE\_MF}} \\ & \underbrace{-\int_{\Omega} \Psi_{m+1} \frac{\partial}{\partial x_\ell} \left(v_k^R v_m^A + \delta_{km} p \right) \frac{\partial}{\partial x_k} \left(\frac{\delta x_\ell}{\delta b_i} \right) \text{d}\Omega + \int_{\Omega} \frac{\partial \Psi_{m+1}}{\partial x_k} v_m^A \frac{\partial v_k^F}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \text{d}\Omega}_{\text{VSD}} \\ & \underbrace{+\int_{\partial\Omega} \Psi_{m+1} \mathsf{n}_k \frac{\delta}{\delta b_i} \left(v_k^R v_m^A + p \delta_{km} \right) \text{d}S}_{\mathcal{S}_2^{\text{INCO}}}\end{aligned} \tag{3.3.1.4}$$

The $\mathcal{I}^{\text{MF\_vis}}$ term is also expanded, by taking additionally into account that since the flow is considered isothermal and incompressible, kinematic viscosity remains constant and, so, $\frac{\delta \nu}{\delta b_i} = 0$.

$$\mathcal{I}^{\text{MF\_vis}} = \underbrace{-\int\limits_{\Omega} \frac{\partial}{\partial x_k}\left[\frac{\nu+\nu_t}{\text{Re}_0}\left(\frac{\partial \Psi_{m+1}}{\partial x_k}+\frac{\Psi_{k+1}}{x_m}\right)\right]\frac{\delta v_m^A}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}}$$
$$\underbrace{+\int\limits_{\Omega}\frac{\partial \Psi_{m+1}}{\partial x_k}\frac{\tau_{km}}{\nu+\nu_t}\left(\mathcal{P}\left(\nu_t,\tilde{\nu}\right)+\mathcal{P}\left(\nu_t,f_{v_1}\right)\mathcal{P}\left(f_{v_1},\chi\right)\mathcal{P}\left(\chi,\tilde{\nu}\right)\right)\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}}$$
$$\underbrace{+\int\limits_{\Omega}\Psi_{m+1}\frac{\partial \tau_{km}}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\text{d}\Omega-\int\limits_{\Omega}\frac{\nu+\nu_t}{\text{Re}_0}\left(\frac{\partial \Psi_{m+1}}{\partial x_k}+\frac{\partial \Psi_{k+1}}{\partial x_m}\right)\frac{\partial v_m^A}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\text{d}\Omega}_{\text{VSD}}$$
$$\underbrace{-\int\limits_{\partial\Omega}\Psi_{m+1}\mathsf{n}_k\frac{\delta \tau_{km}}{\delta b_i}\text{d}S}_{\mathcal{S}_3^{\text{INCO}}}\underbrace{+\int\limits_{\partial\Omega}\frac{\nu+\nu_t}{\text{Re}_0}\left(\frac{\partial \Psi_{m+1}}{\partial x_k}+\frac{\partial \Psi_{k+1}}{\partial x_m}\right)\mathsf{n}_k\frac{\delta v_m^A}{\delta b_i}\text{d}S}_{\mathcal{S}_4^{\text{INCO}}} \qquad (3.3.1.5)$$

Finally, term $\mathcal{I}^{\text{MF\_src}}$ expands as follows

$$\mathcal{I}^{\text{MF\_src}} = \underbrace{\int\limits_{\Omega}\Psi_{m+1}\varepsilon_{m\ell k}\omega_\ell\frac{\delta v_k^A}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}} \qquad (3.3.1.6)$$

3.3.2 Differentiation of the Spalart–Allmaras equation

Taking into account Eq. 2.1.13 for incompressible flows (all terms divided by the constant density), term $\mathcal{I}^{\text{SA}}$ is split into terms arising from the differentiation of the convection, diffusion and source terms of the Spalart–Allmaras turbulence model PDE.

The convection term reads

$$\mathcal{I}^{\text{SA\_conv}} = \underbrace{-\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\tilde{\nu}\frac{\delta v_k^A}{\delta b_i}\text{d}\Omega}_{\text{FAE\_MF}}\underbrace{-\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}v_k^R\frac{\delta\tilde{\nu}}{\delta b_i}\text{d}\Omega}_{\text{FAE\_SA}}\underbrace{+\int\limits_{\partial\Omega}\tilde{\nu}_a\mathsf{n}_k\frac{\delta\left(v_k^R\tilde{\nu}\right)}{\delta b_i}\text{d}S}_{\mathcal{S}_5^{\text{INCO}}}$$

$$\underbrace{-\int\limits_{\Omega}\tilde{\nu}_a\frac{\partial\left(v_k^R\tilde{\nu}\right)}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\mathrm{d}\Omega+\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\tilde{\nu}\frac{\partial v_k^F}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}\Omega}_{\text{VSD}} \tag{3.3.2.1}$$

The diffusion term is split into two terms, namely $\mathcal{I}^{\text{SA\_diff1}}$ and $\mathcal{I}^{\text{SA\_diff2}}$, which are expanded separately. Their development follows

$$\mathcal{I}^{\text{SA\_diff}}=\underbrace{-\frac{1}{\text{Re}_0}\int\limits_{\Omega}\tilde{\nu}_a\frac{\delta}{\delta b_i}\left(\frac{\partial}{\partial x_k}\left\{\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\right)\mathrm{d}\Omega}_{\mathcal{I}^{\text{SA\_diff1}}}+\underbrace{\frac{c_{b_2}}{\text{Re}_0}\int\limits_{\Omega}\tilde{\nu}_a\frac{\delta}{\delta b_i}\left(\frac{\partial^2\tilde{\nu}}{\partial x_k^2}\right)\mathrm{d}\Omega}_{\mathcal{I}^{\text{SA\_diff2}}} \tag{3.3.2.2}$$

where

$$\begin{aligned}\mathcal{I}^{\text{SA\_diff1}}=&\underbrace{\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_k}(1+c_{b_2})\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial}{\partial x_k}\left\{\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial\tilde{\nu}_a}{\partial x_k}\right\}\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}\\&\underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\tilde{\nu}_a\frac{\partial}{\partial x_\ell}\left\{\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}}\\&\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}}\\&\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\tilde{\nu}_a\mathsf{n}_k\frac{\delta}{\delta b_i}\left\{\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\right\}\mathrm{d}S}_{\mathcal{S}_6^{\text{INCO}}}+\underbrace{\frac{1}{\text{Re}_0\,\sigma}\int\limits_{\partial\Omega}\frac{\partial\tilde{\nu}_a}{\partial x_k}\left[\nu+(1+c_{b_2})\,\tilde{\nu}\right]\mathsf{n}_k\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}_{\mathcal{S}_7^{\text{INCO}}}\end{aligned} \tag{3.3.2.3}$$

$$\begin{aligned}\mathcal{I}^{\text{SA\_diff2}}=&\underbrace{\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial^2\left(\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k^2}\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\tilde{\nu}_a\frac{\partial^2\tilde{\nu}}{\partial x_k^2}\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}}\\&\underbrace{-\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\tilde{\nu}_a\tilde{\nu}\frac{\partial}{\partial x_\ell}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\mathrm{d}\Omega+\frac{c_{b_2}}{\text{Re}_0\,\sigma}\int\limits_{\Omega}\frac{\partial\left(\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\frac{\partial\tilde{\nu}}{\partial x_\ell}\frac{\partial}{\partial x_k}\left(\frac{\delta x_\ell}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}}\end{aligned}$$

$$\underbrace{+\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{\partial\Omega}\tilde{\nu}_a\tilde{\nu}\mathsf{n}_k\frac{\delta}{\delta b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\right)\mathrm{d}S}_{\mathcal{S}_8^{\mathrm{INCO}}}\underbrace{-\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int\limits_{\partial\Omega}\frac{\partial\left(\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k}\mathsf{n}_k\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}_{\mathcal{S}_9^{\mathrm{INCO}}} \tag{3.3.2.4}$$

Finally, the differentiation of the turbulence model source terms is presented. Eqs. 3.1.2.2–3.1.2.11 are used, again, with $\frac{\partial}{\partial b_i}$ replaced by $\frac{\delta}{\delta b_i}$. In addition, since the flow is incompressible, all dependencies on density vanish and, since the flow is also considered to be isothermal, ν is constant and $\frac{\delta\nu}{\delta b_i}=0$. The $\mathcal{I}^{\mathrm{SA\_src}}$ term is developed as follows

$$\begin{aligned}\mathcal{I}^{\mathrm{SA\_src}} = &\underbrace{\int\limits_{\Omega}\tilde{\nu}_a\left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{S}+\frac{2\tilde{\nu}}{\mathrm{Re}_0\,\Delta^2}\left(c_{w_1}f_w-\frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\right]\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_SA}}\\
&\underbrace{+\frac{c_{w_1}}{\mathrm{Re}_0}\int\limits_{\Omega}\tilde{\nu}_a\mathcal{C}_4\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_SA}}\underbrace{-\frac{2}{\mathrm{Re}_0}\int\limits_{\Omega}\frac{\tilde{\nu}^2}{\Delta^3}\left(c_{w_1}f_w-\frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\frac{\delta\Delta}{\delta b_i}\mathrm{d}\Omega+\frac{c_{w_1}}{\mathrm{Re}_0}\int\limits_{\Omega}\tilde{\nu}_a\mathcal{C}_5\frac{\delta\Delta}{\delta b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_DISTANCE}}\\
&\underbrace{+\int\limits_{\Omega}\tilde{\nu}_a\left[-c_{b_1}\left(1-f_{t_2}\right)\tilde{\nu}+\frac{c_{w_1}}{\mathrm{Re}_0}\mathcal{C}_3\right]\frac{\delta\tilde{S}}{\delta b_i}\mathrm{d}\Omega}_{\mathcal{I}^{\tilde{S}}}\underbrace{+c_{b_1}\int\limits_{\Omega}\tilde{\nu}_a\left[\tilde{S}\tilde{\nu}-\frac{1}{\mathrm{Re}_0\,\kappa^2}\left(\frac{\tilde{\nu}}{\Delta}\right)^2\right]\frac{\delta f_{t_2}}{\delta b_i}\mathrm{d}\Omega}_{\mathcal{I}^{f_{t_2}}}\end{aligned} \tag{3.3.2.5}$$

with

$$\begin{aligned}\mathcal{I}^{\tilde{S}} = &\underbrace{\int\limits_{\Omega}\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{\delta S}{\delta b_i}\mathrm{d}\Omega}_{\mathcal{I}^{\mathrm{VORTICITY}}}\underbrace{+\int\limits_{\Omega}\tilde{\nu}_a\mathcal{C}_6\left[\mathcal{P}\left(\tilde{S},\tilde{\nu}\right)+\mathcal{C}_1\mathcal{P}\left(\chi,\tilde{\nu}\right)\right]\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_SA}}\\
&\underbrace{+\int\limits_{\Omega}\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},\Delta\right)\frac{\delta\Delta}{\delta b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_DISTANCE}}\end{aligned} \tag{3.3.2.6}$$

$$\mathcal{I}^{\mathrm{VORTICITY}} = \underbrace{-\int\limits_{\Omega}\frac{\partial}{\partial x_\ell}\left[\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\right]\frac{\delta v_m^A}{\delta b_i}\mathrm{d}\Omega}_{\mathrm{FAE\_MF}}$$

$$\underbrace{-\int\limits_{\Omega}\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\frac{\partial v_m^A}{\partial x_p}\frac{\partial}{\partial x_\ell}\left(\frac{\delta x_p}{\delta b_i}\right)\mathrm{d}\Omega}_{\text{VSD}}$$

$$\underbrace{+\int\limits_{\partial\Omega}\tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell\frac{\delta v_m^A}{\delta b_i}\mathrm{d}S}_{\mathrm{S}_{10}^{\mathrm{INCO}}} \tag{3.3.2.7}$$

$$\mathcal{I}^{ft_2}=\underbrace{\int\limits_{\Omega}\tilde{\nu}_a\mathcal{C}_7\mathcal{P}\left(\chi,\tilde{\nu}\right)\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}\Omega}_{\text{FAE\_SA}} \tag{3.3.2.8}$$

All coefficients $\mathcal{C}_1$ to $\mathcal{C}_7$ are the same as in the case of compressible flows, see 3.1.2.

3.3.3 Field Adjoint Equations

The mean-flow and Spalart–Allmaras Field Adjoint Equations (FAE) result, as in the case of compressible flows, by eliminating the terms denoted as FAE_MF and FAE_SA, respectively. The mean flow field adjoint equations are expressed as

$$-A_{nmk}\frac{\partial\Psi_n}{\partial x_k}-\mathcal{K}_m+\mathcal{K}_m^{\mathrm{SA}}+S_m^{\mathrm{adj}}+\frac{\partial F_\Omega}{\partial U_m}=0 \tag{3.3.3.1}$$

where A_{nmk} is the inviscid flux Jacobian of the flow equations given by

$$A_{nmk}=\frac{\partial f_{nk}^{inv}}{\partial U_m}=\begin{bmatrix}0 & \delta_{1k} & \delta_{2k} & \delta_{3k}\\ \delta_{1k} & v_k^R+v_1^A\delta_{1k} & v_1^A\delta_{2k} & v_1^A\delta_{3k}\\ \delta_{2k} & v_2^A\delta_{1k} & v_k^R v_2^A\delta_{2k} & v_2^A\delta_{3k}\\ \delta_{3k} & v_3^A\delta_{1k} & v_3^A\delta_{2k} & v_k^R+v_3^A\delta_{3k}\end{bmatrix}$$

Term $\mathcal{K}_m$ stems from the differentiation of the mean-flow viscous part and is given by

$$\mathcal{K}_m=\frac{\partial\tau_{kn}^{\mathrm{adj}}}{\partial x_k}\frac{\partial v_n^A}{\partial U_m}$$

with the adjoint stresses τ_{kn}^{adj} given by

$$\tau_{kn}^{\mathrm{adj}}=\frac{\nu+\nu_t}{\mathrm{Re}_0}\left(\frac{\partial\Psi_{n+1}}{\partial x_k}+\frac{\partial\Psi_{k+1}}{\partial x_n}\right) \tag{3.3.3.2}$$

The $\mathcal{K}_m^{\text{SA}}$ term arises from the differentiation of the convection and source (due to vorticity) terms of the Spalart–Allmaras equation and is expressed as

$$\mathcal{K}_m^{\text{SA}} = -\frac{\partial \tilde{\nu}_a}{\partial x_k}\tilde{\nu}\frac{\partial v_k^A}{\partial U_m} - \frac{\partial}{\partial x_\ell}\left[\tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right)\frac{1}{S}\varepsilon_{k\ell n}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\right]\frac{\partial v_n^A}{\partial U_m} \tag{3.3.3.3}$$

Finally, the S_m^{adj} term stands for the adjoint Coriolis force term and is expressed as $S_m^{\text{adj}} = \varepsilon_{nk\ell}\Psi_{k+1}\omega_\ell \frac{\partial v_n^A}{\partial U_m}$ and, the $\frac{\partial F_\Omega}{\partial U_m}$ term arises from the differentiation of the objective function and is zero if the objective function is defined only along the boundaries of the flow domain. Similarly, the adjoint to the Spalart–Allmaras equation for incompressible flows reads

$$-\frac{\partial}{\partial x_k}\left(v_k^R \tilde{\nu}_a\right) - \mathcal{D}^{\text{SA,adj}} + \mathcal{G}^{\text{SA,diff}} + \mathcal{G}^{\text{SA,src}} + \mathcal{G}^{\nu_t}\frac{\partial \nu_t}{\partial \tilde{\nu}} + \frac{\partial F_\Omega}{\partial \tilde{\nu}} = 0 \tag{3.3.3.4}$$

where

$$\begin{aligned}
\mathcal{D}^{\text{SA,adj}} &= \frac{1}{\text{Re}_0\,\sigma}\frac{\partial}{\partial x_k}\left\{\left[\nu + \left(1 + c_{b_2}\right)\tilde{\nu}\right]\frac{\partial \tilde{\nu}_a}{\partial x_k}\right\} - \frac{c_{b_2}}{\text{Re}_0\,\sigma}\frac{\partial^2\left(\tilde{\nu}_a\tilde{\nu}\right)}{\partial x_k^2}\\
\mathcal{G}^{\text{SA},diff} &= \frac{1 + c_{b_2}}{\text{Re}_0\,\sigma}\frac{\partial \tilde{\nu}_a}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k} + \frac{c_{b_2}}{\text{Re}_0\,\sigma}\tilde{\nu}_a\frac{\partial^2 \tilde{\nu}}{\partial x_k^2}\\
\mathcal{G}^{\text{SA},src} &= \tilde{\nu}_a\left[-c_{b_1}\left(1 - f_{t_2}\right)\tilde{S} + \frac{2}{\text{Re}_0}\left(c_{w_1}f_w - \frac{c_{b_1}}{\kappa^2}f_{t_2}\right)\left(\frac{\tilde{\nu}}{\Delta^2}\right)\right.\\
&\left. + \frac{c_{w_1}}{\text{Re}_0}\mathcal{C}_4 + \mathcal{C}_6\mathcal{P}\left(\tilde{S}, \tilde{\nu}\right) + \mathcal{C}_6\mathcal{C}_1\mathcal{P}\left(\chi, \tilde{\nu}\right) + \mathcal{C}_7\mathcal{P}\left(\chi, \tilde{\nu}\right)\right]\\
\mathcal{G}^{\nu_t} &= \frac{\tau_{km}}{\nu + \nu_t}\frac{\partial \Psi_{m+1}}{\partial x_k}
\end{aligned} \tag{3.3.3.5}$$

From Eq. 3.3.3.1, it can be seen that the mean flow field adjoint equations are similar to the primal ones, yielding only different signs and source terms. More precisely, the adjoint convection term (first term in Eq. 3.3.3.1) is the same as the primal flow convection term with opposite sign and transposed Jacobian matrix. This means that the first adjoint variable Ψ_1 (also called the adjoint pressure) is absent from the adjoint continuity equation, and, thus the system of equations cannot be solved using coupled solvers for hyperbolic systems of equations. This problem is circumvented by employing a preconditioning method to the adjoint equations, similar to the artificial compressibility. Since the eigenvalues of the adjoint system of equations are the same with the primal ones, with different sign though, the transpose of the same preconditioning matrix can be used. Thus, after adding the pseudo-time contribution to the adjoint system of equations, these read

$$\Gamma^{-1}_{nm}\frac{\partial \Psi_n}{\partial t} - A_{nmk}\frac{\partial \Psi_n}{\partial x_k} - \mathcal{K}_m + \mathcal{K}^{\mathrm{SA}}_m + S^{\mathrm{adj}}_m + \frac{\partial F_\Omega}{\partial U_m} = 0, \quad m = 1,\dots,4 \tag{3.3.3.6}$$

where matrix Γ_{nm} is given by Eq. 2.6.3.3.

3.3.4 Adjoint Boundary Conditions

After eliminating all volume integrals, the surface integrals $\mathcal{S}_1$ to $\mathcal{S}_{10}$ remain. Taking into account the flow boundary conditions, these are split into terms containing variations of flow quantities and terms containing variations of geometric quantities. The first are eliminated by applying appropriate adjoint boundary conditions, while the latter contribute to the expression of sensitivity derivatives. Each type of boundary is separately presented in the following paragraphs.

3.3.4.1 Wall Boundaries

Slip Wall Boundaries $S_{W^{\mathrm{Slip}}}$

On slip wall boundaries, the primal boundary condition for $\tilde{\nu}$ is zero Neumann. Thus, $\frac{\delta}{\delta b_i}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right) = 0$.

Integrals $\mathcal{S}_6$ and $\mathcal{S}_8$ are developed as

$$\mathcal{S}_6 = \cancel{-\frac{1}{\mathrm{Re}_0\,\sigma}\int_{S_{W^{\mathrm{Slip}}}} \tilde{\nu}_a \frac{\delta}{\delta b_i}\left\{\left[\nu + (1+c_{b_2})\tilde{\nu}\right]\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right\}} \mathrm{d}S + \underbrace{\frac{1}{\mathrm{Re}_0\,\sigma}\int_{S_{W^{\mathrm{Slip}}}} \tilde{\nu}_a\left[\nu + (1+c_{b_2})\tilde{\nu}\right]\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\delta(\mathsf{n}_k\mathrm{d}S)}{\delta b_i}}_{\mathrm{SD}} \tag{3.3.4.1}$$

$$\mathcal{S}_8 = \cancel{\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int_{S_{W^{\mathrm{Slip}}}} \tilde{\nu}_a\tilde{\nu}\frac{\delta}{\delta b_i}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\mathrm{d}S\right)} \underbrace{-\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma}\int_{S_{W^{\mathrm{Slip}}}} \tilde{\nu}_a\tilde{\nu}\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\delta(\mathsf{n}_k\mathrm{d}S)}{\delta b_i}}_{\mathrm{SD}} \tag{3.3.4.2}$$

whereas integrals $\mathcal{S}_7$ and $\mathcal{S}_9$ become

$$\mathcal{S}_7 = \underbrace{\frac{1}{\mathrm{Re}_0\,\sigma} \int\limits_{S_{W^{\text{Slip}}}} \frac{\partial \tilde{\nu}_a}{\partial x_k} \mathsf{n}_k \left[\nu + (1 + c_{b_2})\,\tilde{\nu}\right] \frac{\delta \tilde{\nu}}{\delta b_i} \mathrm{d}S}_{\mathrm{ABC}^{\tilde{\nu}}} \tag{3.3.4.3}$$

$$\mathcal{S}_9 = \cancel{-\frac{c_{b_2}}{\mathrm{Re}_0\,\sigma} \int\limits_{S_{W^{\text{Slip}}}} \tilde{\nu}_a \frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \frac{\delta \tilde{\nu}}{\delta b_i} \mathrm{d}S} \underbrace{- \frac{c_{b_2}}{\mathrm{Re}_0\,\sigma} \int\limits_{S_{W^{\text{Slip}}}} \tilde{\nu} \frac{\partial \tilde{\nu}_a}{\partial x_k} \mathsf{n}_k \frac{\delta \tilde{\nu}}{\delta b_i} \mathrm{d}S}_{\mathrm{ABC}^{\tilde{\nu}}} \tag{3.3.4.4}$$

The rest of the integrals are developed separately for stationary and rotating walls since the no-penetration condition is expressed differently at these two types of boundaries.

Stationary Slip Wall Boundaries $S_{W^{\text{Slip,St}}}$

For stationary slip walls, the no-penetration condition is expressed as $v_k^A \mathsf{n}_k = 0$. Additionally, in order to simulate stationary walls with a steady solver, these boundaries must be such that $v_k^F \mathsf{n}_k = 0$. Consequently, the following expressions are true for the variation of the absolute and relative velocity components

$$\frac{\delta \left(v_k^A \mathsf{n}_k\right)}{\delta b_i} = 0 \tag{3.3.4.5}$$

$$\frac{\delta \left(v_k^R \mathsf{n}_k\right)}{\delta b_i} = \cancel{\frac{\delta \left(v_k^A \mathsf{n}_k\right)}{\delta b_i}} - \cancel{\frac{\delta \left(v_k^F \mathsf{n}_k\right)}{\delta b_i}} = 0 \tag{3.3.4.6}$$

First, integral $\mathcal{S}_1$ is expanded

$$\mathcal{S}_1 = \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_1 \frac{\delta \left(v_k^R \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} - \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_1 v_k^R \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\mathrm{SD}} \tag{3.3.4.7}$$

and integral $\mathcal{S}_2$ leads to

$$\mathcal{S}_2 = \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\frac{\delta}{\delta b_i}\left(v_k^R \mathsf{n}_k v_m^A \mathrm{d}S\right)} + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\mathsf{n}_m \frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1} v_k^R v_m^A \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.8}$$

The integrals arising from the viscous terms differentiation are expanded as

$$\begin{aligned}\mathcal{S}_3 = &\underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\mathsf{n}_m \frac{\delta\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}{\delta b_i}\mathrm{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell \frac{\delta\left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\ &\cancel{- \int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\frac{\delta}{\delta b_i}\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\mathsf{t}_m \mathrm{d}S\right)} + \underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \Psi_{m+1}\tau_{km}\frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}\end{aligned} \tag{3.3.4.9}$$

$$\begin{aligned}\mathcal{S}_4 = &\cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \tau_{k\ell}^{\text{adj}}\mathsf{n}_k\mathsf{n}_\ell \frac{\delta\left(v_m^A \mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \tau_{k\ell}^{\text{adj}}\mathsf{n}_k\mathsf{n}_\ell v_m^A \frac{\delta\left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \\ &\underbrace{+ \int\limits_{S_{W^{\text{Slip,St}}}} \tau_{k\ell}^{\text{adj}}\mathsf{n}_k\mathsf{t}_\ell \frac{\delta\left(v_m^A \mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(v_m^A \mathsf{t}_m \mathrm{d}S\right)}} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \tau_{k\ell}^{\text{adj}}\mathsf{n}_k\mathsf{t}_\ell v_m^A \frac{\delta\left(\mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_1}\end{aligned} \tag{3.3.4.10}$$

Integrals $\mathcal{S}_5$ and $\mathcal{S}_{10}$ lead to

$$\mathcal{S}_5 = \cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a \frac{\delta\left(v_k^R \mathsf{n}_k \tilde{\nu} \mathrm{d}S\right)}{\delta b_i}} \underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a v_k^R \tilde{\nu} \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.11}$$

$$\begin{aligned}\mathcal{S}_{10} = &\cancel{\int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right)\frac{1}{S}\varepsilon_{k\ell p}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell\mathsf{n}_p \frac{\delta\left(v_m^A \mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}} \\ &\underbrace{- \int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right)\frac{1}{S}\varepsilon_{k\ell p}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell\mathsf{n}_p v_m^A \frac{\delta\left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}\end{aligned}$$

$$+\underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell p}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell \mathsf{t}_p \frac{\delta\left(v_m^A \mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(v_m^A \mathsf{t}_m \mathrm{d}S\right)}}$$
$$-\underbrace{\int\limits_{S_{W^{\text{Slip,St}}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell p}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell \mathsf{t}_p v_m^A \frac{\delta\left(\mathsf{t}_m \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_2} \tag{3.3.4.12}$$

Finally, the last term of Eq. 3.3 is split into the following terms

$$\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\delta F_S}{\delta b_i}\mathrm{d}S = \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial x_k}\frac{\delta x_k}{\delta b_i}\mathrm{d}S + \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\mathsf{n}_k \mathrm{d}S\right)}\frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\mathsf{t}_k \mathrm{d}S\right)}\frac{\delta\left(\mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_3}$$
$$+\underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial p}\frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}\frac{\delta\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}{\delta b_i}\mathrm{d}S}_{\text{ABC}^{\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)}\frac{\delta\left(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell\right)}{\delta b_i}\mathrm{d}S}$$
$$+\cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v_k^A \mathsf{n}_k \mathrm{d}S\right)}\frac{\delta\left(v_k^A \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v_k^A \mathsf{t}_k \mathrm{d}S\right)}\frac{\delta\left(v_k^A \mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(v_k^A \mathsf{t}_k\right)}}$$
$$+\underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial F_S}{\partial\tilde{\nu}}\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}_{\text{ABC}^{\tilde{\nu}}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,St}}}} \frac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)}\frac{\delta}{\delta b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k \mathrm{d}S\right)} \tag{3.3.4.13}$$

Eliminating the ABC^p and $\text{ABC}^{\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}$ terms leads to the following adjoint boundary condition

$$\Psi_{m+1}\mathsf{n}_m = \begin{cases} -\dfrac{\partial F_S}{\partial p}\ , \text{ at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ \qquad 0\ , \text{ at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.3.4.14}$$

$$\Psi_{m+1}\mathsf{n}_m = \begin{cases} \dfrac{\partial F_S}{\partial\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}\ , \text{ at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ \qquad 0\ , \text{ at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.3.4.15}$$

In order to obtain a unique adjoint boundary condition for $\Psi_{m+1}\mathsf{n}_m$, F_S must be such that $\frac{\partial F_S}{\partial p} = -\frac{\partial F_S}{\partial(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}$ at $S^{\text{Obj}}_{W^{\text{Slip,St}}}$. To eliminate the ABC$^{\left(v_k^A \mathsf{t}_k\right)}$ terms, we must set

$$\tau^{\text{adj}}_{k\ell}\mathsf{n}_k\mathsf{t}_\ell + \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right)\frac{1}{S}\varepsilon_{k\ell p}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell\mathsf{t}_p = \begin{cases} -\dfrac{\partial\left(F_S \mathrm{d}S\right)}{\partial\left(v_k^A \mathsf{t}_k \mathrm{d}S\right)} & \text{, at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ 0 & \text{, at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.3.4.16}$$

Eq. 3.3.4.16 is taken into account when computing the adjoint boundary flux at stationary slip wall boundaries. Since Eq. 3.3.4.16 holds the sum of terms $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ vanishes automatically, if $\frac{\partial(F_S \mathrm{d}S)}{\partial(\mathsf{t}_k \mathrm{d}S)} = \frac{\partial(F_S \mathrm{d}S)}{\partial\left(v_k^A \mathsf{t}_k \mathrm{d}S\right)}$. Finally, in order to eliminate the terms denoted as ABC$^{\tilde{\nu}}$, the following boundary condition is imposed:

$$\frac{\partial \tilde{\nu}_a}{\partial x_k}\mathsf{n}_k = \begin{cases} -\dfrac{\text{Re}_0\,\sigma}{\nu + \nu_t}\dfrac{\partial F_S}{\partial \tilde{\nu}} & \text{, at } S^{\text{Obj}}_{W^{\text{Slip,St}}} \\ 0 & \text{, at } S_{W^{\text{Slip,St}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,St}}} \end{cases} \tag{3.3.4.17}$$

Rotating Slip Wall Boundaries $S_{W^{\text{Slip,Rot}}}$

For rotating slip walls, the no-penetration condition is expressed as $v_k^R \mathsf{n}_k = 0$ As a result, $\frac{\delta\left(v_k^R \mathsf{n}_k\right)}{\delta b_i} = 0$. The variation in the absolute velocity is given by

$$\frac{\delta\left(v_k^A \mathsf{n}_k\right)}{\delta b_i} = \frac{\delta\left(v_k^F \mathsf{n}_k\right)}{\delta b_i} \tag{3.3.4.18}$$

Integrals $\mathcal{S}_1$ to $\mathcal{S}_4$ expand as

$$\mathcal{S}_1 = \int_{S_{W^{\text{Slip,Rot}}}} \cancel{\Psi_1 \frac{\delta\left(v_k^R \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} - \underbrace{\int_{S_{W^{\text{Slip,Rot}}}} \Psi_1 v_k^R \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.19}$$

$$\mathcal{S}_2 = \int_{\cancel{S_{W^{\text{Slip,Rot}}}}} \cancel{\Psi_{m+1}\frac{\delta}{\delta b_i}\left(v_k^R \mathsf{n}_k v_m^A \mathrm{d}S\right)} + \underbrace{\int_{S_{W^{\text{Slip,Rot}}}} \Psi_{m+1}\mathsf{n}_m \frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p} - \underbrace{\int_{S_{W^{\text{Slip,Rot}}}} \Psi_{m+1} v_k^R v_m^A \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.20}$$

$$\mathcal{S}_3 = \underbrace{-\int_{S_{W^{\text{Slip,Rot}}}} \Psi_{m+1}\mathsf{n}_m \frac{\delta\left(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell\right)}{\delta b_i}\mathrm{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}} - \underbrace{\int_{S_{W^{\text{Slip,Rot}}}} \Psi_{m+1}\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell \frac{\delta\left(\mathsf{n}_m \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}}$$

$$
- \cancel{\int_{S_W^{\text{Slip,Rot}}} \Psi_{m+1} \frac{\delta}{\delta b_i} \left(\tau_{k\ell} n_k t_\ell t_m \mathrm{d}S \right)} + \underbrace{\int_{S_W^{\text{Slip,Rot}}} \Psi_{m+1} \tau_{km} \frac{\delta \left(n_k \mathrm{d}S \right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.21}
$$

$$
\begin{aligned}
\mathcal{S}_4 = & \cancel{\int_{S_W^{\text{Slip,Rot}}} \tau^{\text{adj}}_{k\ell} n_k n_\ell \frac{\delta \left(v^R_m n_m \mathrm{d}S \right)}{\delta b_i} \mathrm{d}S} + \underbrace{\int_{S_W^{\text{Slip,Rot}}} \tau^{\text{adj}}_{k\ell} n_k t_\ell \frac{\delta \left(v^R_m t_m \mathrm{d}S \right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}\left(v^R_m t_m\right)} \\
& \underbrace{- \int_{S_W^{\text{Slip,Rot}}} \tau^{\text{adj}}_{k\ell} n_k n_\ell v^R_m \frac{\delta \left(n_m \mathrm{d}S \right)}{\delta b_i}}_{\text{SD}} \underbrace{- \int_{S_W^{\text{Slip,Rot}}} \tau^{\text{adj}}_{k\ell} n_k t_\ell v^R_m \frac{\delta \left(t_m \mathrm{d}S \right)}{\delta b_i}}_{\mathcal{T}_1} \underbrace{+ \int_{S_W^{\text{Slip,Rot}}} \tau^{\text{adj}}_{km} n_k \frac{\partial v^F_m}{\partial x_\ell} \frac{\delta x_\ell}{\delta b_i} \mathrm{d}S}_{\text{SD}}
\end{aligned} \tag{3.3.4.22}
$$

Similarly, terms $\mathcal{S}_5$ and $\mathcal{S}_{10}$ expand to

$$
\mathcal{S}_5 = \cancel{\int_{S_W^{\text{Slip,Rot}}} \tilde{\nu}_a \frac{\delta \left(v^R_k n_k \tilde{\nu} \mathrm{d}S \right)}{\delta b_i}} \underbrace{- \int_{S_W^{\text{Slip,Rot}}} \tilde{\nu}_a \tilde{\nu} v^R_k \frac{\delta \left(n_k \mathrm{d}S \right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.23}
$$

$$
\begin{aligned}
\mathcal{S}_{10} = & \cancel{\int_{S_W^{\text{Slip,Rot}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell p} \varepsilon_{kqr} \frac{\partial v^A_r}{\partial x_q} n_\ell n_p \frac{\delta \left(v^R_m n_m \mathrm{d}S \right)}{\delta b_i}} \\
& \underbrace{+ \int_{S_W^{\text{Slip,Rot}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell p} \varepsilon_{kqr} \frac{\partial v^A_r}{\partial x_q} n_\ell t_p \frac{\delta \left(v^R_m t_m \mathrm{d}S \right)}{\delta b_i}}_{\text{ABC}\left(v^R_m t_m\right)} \\
& \underbrace{- \int_{S_W^{\text{Slip,Rot}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell p} \varepsilon_{kqr} \frac{\partial v^A_r}{\partial x_q} n_\ell n_p v^R_m \frac{\delta \left(n_m \mathrm{d}S \right)}{\delta b_i}}_{\text{SD}} \\
& \underbrace{- \int_{S_W^{\text{Slip,Rot}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell p} \varepsilon_{kqr} \frac{\partial v^A_r}{\partial x_q} n_\ell t_p v^R_m \frac{\delta \left(t_m \mathrm{d}S \right)}{\delta b_i}}_{\mathcal{T}_2} \\
& \underbrace{+ \int_{S_W^{\text{Slip,Rot}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right) \frac{1}{S} \varepsilon_{k\ell m} \varepsilon_{kqr} \frac{\partial v^A_r}{\partial x_q} n_\ell \frac{\partial v^F_m}{\partial x_p} \frac{\delta x_p}{\delta b_i} \mathrm{d}S}_{\text{SD}}
\end{aligned} \tag{3.3.4.24}
$$

Finally, the last term of Eq. 3.3 is split into the following terms

$$
\begin{aligned}
\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\delta F_S}{\delta b_i} \mathrm{d}S = & \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial x_k} \frac{\delta x_k}{\delta b_i} \mathrm{d}S + \int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\mathsf{n}_k \mathrm{d}S\right)} \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\mathsf{t}_k \mathrm{d}S\right)} \frac{\delta \left(\mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_3} \\
& + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial p} \frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial \left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)} \frac{\delta \left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell)}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial \left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell\right)} \frac{\delta \left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell\right)}{\delta b_i} \mathrm{d}S} \\
& + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(v^R_k \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta \left(v^R_k \mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(v^R_k \mathsf{t}_k \mathrm{d}S\right)} \frac{\delta \left(v^R_k \mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{ABC}^{\left(v^R_k \mathsf{t}_k\right)}} \\
& + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial F_S}{\partial \tilde{\nu}} \frac{\delta \tilde{\nu}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{\tilde{\nu}}} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{Slip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta}{\delta b_i} \left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \mathrm{d}S\right)}
\end{aligned}
\tag{3.3.4.25}
$$

Eliminating the ABC^p and $\text{ABC}^{(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell)}$ terms leads to the following adjoint boundary conditions

$$
\Psi_{m+1} \mathsf{n}_m = \begin{cases} -\dfrac{\partial F_S}{\partial p} & \text{, at } S^{\text{Obj}}_{W^{\text{Slip,Rot}}} \\ 0 & \text{, at } S_{W^{\text{Slip,Rot}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,Rot}}} \end{cases}
\tag{3.3.4.26}
$$

$$
\Psi_{m+1} \mathsf{n}_m = \begin{cases} \dfrac{\partial F_S}{\partial \left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)} & \text{, at } S^{\text{Obj}}_{W^{\text{Slip,Rot}}} \\ 0 & \text{, at } S_{W^{\text{Slip,Rot}}} \setminus S^{\text{Obj}}_{W^{\text{Slip,Rot}}} \end{cases}
\tag{3.3.4.27}
$$

Eqs. 3.3.4.26, 3.3.4.27 are the same as Eqs. 3.3.4.14, 3.3.4.15. Thus, on slip stationary and rotating wall boundaries, the same adjoint boundary condition is imposed to $\Psi_{m+1} \mathsf{n}_m$. In addition the same restriction on the objective function is implied, namely $\frac{\partial F_S}{\partial p} = -\frac{\partial F_S}{\partial (\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell)}$.

To eliminate the $\text{ABC}^{\left(v^R_k \mathsf{t}_k\right)}$ terms, we must set

$$\tau^{\text{adj}}_{k\ell}\mathsf{n}_k\mathsf{t}_\ell + \tilde{\nu}_a\mathcal{C}_6\mathcal{P}\left(\tilde{S},S\right)\frac{1}{S}\varepsilon_{k\ell p}\varepsilon_{kqr}\frac{\partial v^R_r}{\partial x_q}\mathsf{n}_\ell\mathsf{t}_p = \begin{cases} -\dfrac{\partial\left(F_S\mathrm{d}S\right)}{\partial\left(v^R_k\mathsf{t}_k\mathrm{d}S\right)}\text{ , at } S^{\text{Obj}}_{W^{\text{Slip,Rot}}} \\ 0\text{ , at } S_{W^{\text{Slip,Rot}}} - S^{\text{Obj}}_{W^{\text{Slip,Rot}}} \end{cases} \tag{3.3.4.28}$$

Eq. 3.3.4.28 is taken into account when computing the adjoint boundary flux across the rotating slip wall boundaries. Since Eq. 3.3.4.28 holds the sum of terms $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ vanishes automatically, if $\frac{\partial(F_S\mathrm{d}S)}{\partial(\mathsf{t}_k\mathrm{d}S)} = \frac{\partial(F_S\mathrm{d}S)}{\partial\left(v^R_k\mathsf{t}_k\mathrm{d}S\right)}$. Finally, in order to eliminate terms marked as $\text{ABC}^{\tilde{\nu}}$, the boundary condition expressed by Eq. 3.3.4.17 is applied with $S_{W^{\text{Slip,St}}}$ and $S^{\text{Obj}}_{W^{\text{Slip,St}}}$ replaced by $S_{W^{\text{Slip,Rot}}}$ and $S^{\text{Obj}}_{W^{\text{Slip,Rot}}}$ respectively.

No-Slip Wall Boundaries $S_{W^{\text{NoSlip}}}$

Along no-slip walls, the boundary condition imposed for the turbulence model variable is $\tilde{\nu}=0$. Consequently, integrals $\mathcal{S}_7$-$\mathcal{S}_9$ vanish automatically for these boundaries and integral $\mathcal{S}_6$ leads to

$$\mathcal{S}_6 = \cancel{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{S_{W^{\text{NoSlip}}}}\tilde{\nu}_a\mathsf{n}_k\frac{\partial\tilde{\nu}}{\partial x_k}\left(1+c_{b_2}\right)\frac{\delta\tilde{\nu}}{\delta b_i}\mathrm{d}S}\underbrace{-\frac{1}{\text{Re}_0\,\sigma}\int\limits_{S_{W^{\text{NoSlip}}}}\tilde{\nu}_a\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\delta}{\delta b_i}\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\mathrm{d}S\right)}_{\text{ABC}^{\left(\frac{\partial\tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}}$$
$$\underbrace{+\frac{1}{\text{Re}_0\,\sigma}\int\limits_{S_{W^{\text{NoSlip}}}}\tilde{\nu}_a\left[\nu+\left(1+c_{b_2}\right)\tilde{\nu}\right]\frac{\partial\tilde{\nu}}{\partial x_k}\frac{\delta\left(\mathsf{n}_k\mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.29}$$

Additionally, the no-slip condition is imposed to the velocity components, which is expressed as $v^A_m = 0, (m=1,\ldots,3)$ for stationary walls and $v^R_m = 0, (m=1,\ldots,3)$ for rotating wall boundaries. Integrals including the v^R and v^A components are expanded separately for $S_{W^{\text{NoSlip,St}}}$ and $S_{W^{\text{NoSlip,Rot}}}$ boundaries in the next paragraphs.

Stationary No-Slip Wall Boundaries $S_{W^{\text{NoSlip,St}}}$

Along them, since $v^A_m = 0, (m=1,\ldots,3)$, the variation in the absolute velocity components is zero. In addition, the stationary wall geometry must be such that $v^F_m\mathsf{n}_m = 0$ so as to model such a boundary by solving the steady state equations expressed in a rotating frame of reference. Thus, the following conditions are taken into account

$$\begin{aligned}
&\frac{\delta v_m^A}{\delta b_i} = 0,\ m = 1, \ldots, 3 \\
&\frac{\delta \left(v_m^F \mathsf{n}_m \right)}{\delta b_i} = 0 \Rightarrow \frac{\delta \left(v_m^R \mathsf{n}_m \right)}{\delta b_i} = 0
\end{aligned} \tag{3.3.4.30}$$

Integrals $\mathcal{S}_4$, $\mathcal{S}_5$ and $\mathcal{S}_{10}$ vanish automatically, while integrals $\mathcal{S}_1$ to $\mathcal{S}_3$ and $\mathcal{S}_5$ lead to

$$\mathcal{S}_1 = \cancel{\int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_1 \frac{\delta \left(v_k^R \mathsf{n}_k \mathrm{d}S \right)}{\delta b_i}} + \underbrace{\int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_1 v_k^F \frac{\delta \left(\mathsf{n}_k \mathrm{d}S \right)}{\delta b_i}}_{\text{SD}} \tag{3.3.4.31}$$

$$\mathcal{S}_2 = \cancel{\int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{m+1} \frac{\delta \left(v_k^R \mathsf{n}_k v_m^A \mathrm{d}S \right)}{\delta b_i}} - \cancel{\int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{m+1} v_k^R v_m^A \frac{\delta \left(\mathsf{n}_k \mathrm{d}S \right)}{\delta b_i}} + \underbrace{\int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{m+1} \mathsf{n}_m \frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} \tag{3.3.4.32}$$

$$\begin{aligned}
\mathcal{S}_3 = & \underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{m+1} \mathsf{n}_m \frac{\delta \left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell \right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{m+1} \mathsf{t}_m \frac{\delta \left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell \right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}} \\
& \underbrace{+ \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{m+1} \tau_{km} \frac{\delta \left(\mathsf{n}_k \mathrm{d}S \right)}{\delta b_i} + \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{m+1} \tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell \frac{\delta \left(\mathsf{n}_m \mathrm{d}S \right)}{\delta b_i}}_{\text{SD}} \\
& \underbrace{+ \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{q+1} \mathsf{t}_q \mathsf{t}_m \tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell \frac{\delta \left(\mathsf{t}_m \mathrm{d}S \right)}{\delta b_i}}_{\mathcal{T}_1} \underbrace{- \int\limits_{S_{W^{\text{NoSlip,St}}}} \Psi_{q+1} \mathsf{n}_q \tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell \mathsf{t}_m \frac{\delta \left(\mathsf{n}_m \mathrm{d}S \right)}{\delta b_i}}_{\text{SD}}
\end{aligned} \tag{3.3.4.33}$$

Finally, the term that originates from the differentiation of the F_S part of the objective function is expressed as

$$\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\delta F_S}{\delta b_i}\text{d}S = \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial x_k}\frac{\delta x_k}{\delta b_i}\text{d}S + \int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial (F_S \text{d}S)}{\partial (\mathsf{n}_k \text{d}S)}\frac{\delta (\mathsf{n}_k \text{d}S)}{\delta b_i}}_{\text{SD}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial (F_S \text{d}S)}{\partial (\mathsf{t}_k \text{d}S)}\frac{\delta (\mathsf{t}_k \text{d}S)}{\delta b_i}}_{\mathcal{T}_2}$$
$$+ \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial p}\frac{\delta p}{\delta b_i}\text{d}S}_{\text{ABC}^p} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial (\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}\frac{\delta (\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}{\delta b_i}\text{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial (\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}\frac{\delta (\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}{\delta b_i}\text{d}S}_{\text{ABC}^{(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}}$$
$$+ \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial v^A_k}\frac{\delta v^A_k}{\delta b_i}\text{d}S} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial F_S}{\partial \tilde{\nu}}\frac{\delta \tilde{\nu}}{\delta b_i}\text{d}S} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,St}}}} \frac{\partial (F_S \text{d}S)}{\partial \left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k \text{d}S\right)}\frac{\delta}{\delta b_i}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k \text{d}S\right)}_{\text{ABC}^{\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}} \quad (3.3.4.34)$$

The ABC^p and ABC$^{(\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}$ terms are eliminated in the same way as for the slip stationary walls. Thus, the adjoint boundary conditions are given by Eq. 3.3.4.14 or Eq. 3.3.4.15 , expressed on $S_{W^{\text{NoSlip,St}}}$ with the same constraint on the objective function, namely $\frac{\partial F_S}{\partial p} = -\frac{\partial F_S}{\partial (\tau_{k\ell}\mathsf{n}_k\mathsf{n}_\ell)}$ on $S_{W^{\text{NoSlip,St}}}$.

The ABC$^{(\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}$ terms are eliminated by setting

$$\Psi_{m+1}\mathsf{t}_m = \begin{cases} \dfrac{\partial F_S}{\partial (\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases} \quad (3.3.4.35)$$

If, additionally, the objective function satisfies the following constraint

$$\frac{\partial F_S}{\partial (\tau_{k\ell}\mathsf{n}_k\mathsf{t}_\ell)}\mathsf{t}_m\tau_{pq}\mathsf{n}_p\mathsf{t}_q + \frac{\partial (F_S \text{d}S)}{\partial (\mathsf{t}_k \text{d}S)} = 0 \quad (3.3.4.36)$$

then the sum of terms $\mathcal{T}_1$ and $\mathcal{T}_2$ vanishes automatically.

Finally, terms ABC$^{\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}$ are eliminated by setting

$$\tilde{\nu}_a = \begin{cases} \dfrac{\text{Re}_0\,\sigma}{\nu + (1 + c_{b_2})}\dfrac{\partial (F_S \text{d}S)}{\partial \left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k \text{d}S\right)} \text{ , at } S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \\ 0 \text{ , at } S_{W^{\text{NoSlip,St}}} \setminus S^{\text{Obj}}_{W^{\text{NoSlip,St}}} \end{cases} \quad (3.3.4.37)$$

Rotating No-Slip Wall Boundaries $S_{W^{\text{NoSlip,Rot}}}$

For rotating no-slip walls, the no-slip condition is expressed as $v_m^R = 0, \ (m = 1, \ldots, 3)$ and, consequently, the variation in the relative velocity is zero. Hence, the conditions that must be taken into account read

$$
\begin{aligned}
&\frac{\delta v_m^R}{\delta b_i} = 0 \\
&\frac{\delta v_m^A}{\delta b_i} = \cancel{\frac{\delta v_m^R}{\delta b_i}} + \frac{\partial v_m^F}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}
\end{aligned}
\tag{3.3.4.38}
$$

Integrals $\mathcal{S}_1$ and $\mathcal{S}_5$ vanish automatically. Integral $\mathcal{S}_3$ is expanded in the same way as for the stationary no-slip walls (Eq. 3.3.4.33). The remaining surface integrals are treated as follows

$$
\begin{aligned}
\mathcal{S}_2 = & \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1}\frac{\delta}{\delta b_i}\left(v_k^R \mathsf{n}_k v_m^A\right)\mathrm{d}S} - \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1} v_k^R v_m^A \frac{\delta\left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}} \\
& \underbrace{+ \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \Psi_{m+1}\mathsf{n}_m \frac{\delta p}{\delta b_i}\mathrm{d}S}_{\text{ABC}^p}
\end{aligned}
\tag{3.3.4.39}
$$

$$
\mathcal{S}_4 = \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tau_{km}^{\text{adj}}\mathsf{n}_k \frac{\delta v_m^R}{\delta b_i}\mathrm{d}S} + \underbrace{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tau_{km}^{\text{adj}}\mathsf{n}_k \frac{\partial v_m^F}{\partial x_\ell}\frac{\delta x_\ell}{\delta b_i}\mathrm{d}S}_{\text{SD}}
\tag{3.3.4.40}
$$

$$
\begin{aligned}
\mathcal{S}_{10} = & \cancel{\int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell \frac{\delta v_m^R}{\delta b_i}\mathrm{d}S} \\
& \underbrace{+ \int\limits_{S_{W^{\text{NoSlip,Rot}}}} \tilde{\nu}_a \mathcal{C}_6 \mathcal{P}\left(\tilde{S}, S\right)\frac{1}{S}\varepsilon_{k\ell m}\varepsilon_{kqr}\frac{\partial v_r^A}{\partial x_q}\mathsf{n}_\ell \frac{\partial v_m^F}{\partial x_p}\frac{\delta x_p}{\delta b_i}\mathrm{d}S}_{\text{SD}}
\end{aligned}
\tag{3.3.4.41}
$$

The term originating from the differentiation of the objective function (part F_S) is expressed as

$$
\begin{aligned}
\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\delta F_S}{\delta b_i} \mathrm{d}S = & \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial F_S}{\partial x_k} \frac{\delta x_k}{\delta b_i} \mathrm{d}S + \int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\mathsf{n}_k \mathrm{d}S\right)} \frac{\delta \left(\mathsf{n}_k \mathrm{d}S\right)}{\delta b_i}}_{\text{SD}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\mathsf{t}_k \mathrm{d}S\right)} \frac{\delta \left(\mathsf{t}_k \mathrm{d}S\right)}{\delta b_i}}_{\mathcal{T}_2} \\
& + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial F_S}{\partial p} \frac{\delta p}{\delta b_i} \mathrm{d}S}_{\text{ABC}^p} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial F_S}{\partial \left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)} \frac{\delta \left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{\left(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell\right)}} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial F_S}{\partial \left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell\right)} \frac{\delta \left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell\right)}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{\left(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell\right)}} \\
& + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial F_S}{\partial v^R_k} \frac{\delta v^R_k}{\delta b_i} \mathrm{d}S} + \cancel{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial F_S}{\partial \tilde{\nu}} \frac{\delta \tilde{\nu}}{\delta b_i} \mathrm{d}S} + \underbrace{\int\limits_{S^{\text{Obj}}_{W^{\text{NoSlip,Rot}}}} \frac{\partial \left(F_S \mathrm{d}S\right)}{\partial \left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \mathrm{d}S\right)} \frac{\delta}{\delta b_i} \left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k \mathrm{d}S\right)}_{\text{ABC}^{\left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k\right)}}
\end{aligned}
\tag{3.3.4.42}
$$

Terms ABC^p and $\text{ABC}^{(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell)}$ are eliminated in the same way as for no-slip rotating wall boundaries. Thus, the adjoint boundary conditions are given by Eq. 3.3.4.26 or Eq. 3.3.4.27, expressed on $S_{W^{\text{NoSlip,Rot}}}$ with the same constraint on the objective function definition, namely $\frac{\partial F_S}{\partial p} = -\frac{\partial F_S}{\partial(\tau_{k\ell} \mathsf{n}_k \mathsf{n}_\ell)}$ on $S_{W^{\text{NoSlip,Rot}}}$.

Terms $\text{ABC}^{(\tau_{k\ell} \mathsf{n}_k \mathsf{t}_\ell)}$ are eliminated by the same adjoint boundary condition expressed by Eq. 3.3.4.35. Finally, the adjoint boundary condition Eq. 3.3.4.37 expressed along $S_{W^{\text{NoSlip,Rot}}}$ leads to the elimination of the $\text{ABC}^{\left(\frac{\partial \tilde{\nu}}{\partial x_k} \mathsf{n}_k\right)}$ terms.

3.3.4.2 Inlet and Outlet Boundaries $S_{I/O}$

The flow boundary conditions for the inlet and outlet are described in Sections 2.6.1 and 2.6.2, respectively. A set of local flow quantities (V^{loc}_ℓ, $\ell = 1, \ldots, 4$) is defined at these boundaries. This set is formed by the flow quantities for which a Dirichlet condition is imposed and the flow quantities which are extrapolated from the interior of the domain.

As in the case of compressible flows, the surface integrals that arise from the differentiation of the viscous terms, $\mathcal{S}_3$ and $\mathcal{S}_4$, are neglected, by assuming that the variation in viscous stresses is zero along the inlet and outlet. Inlet and outlet are considered unparameterized, so their contribution on the surface sensitivity derivative terms is zero.

Inlet Boundaries S_I

For the inlet, let $V_\ell^{\text{loc}}, (\ell = 1, \dots, 3)$ be the quantities for which a Dirichlet condition is imposed and V_4^{loc} the flow quantity the value of which is extrapolated from the interior of the domain. This leads to $\frac{\delta V_\ell^{\text{loc}}}{\delta b_i} = 0, (\ell = 1, \dots, 3)$. Developing terms $\mathcal{S}_1$ and $\mathcal{S}_2$, the following terms arise

$$\begin{aligned}\mathcal{S}_1 = & \cancel{\int_{S_I} \Psi_1 \mathsf{n}_k \frac{\partial f_{1k}^{inv}}{\partial U_m} \frac{\partial U_m}{\partial V_1^{\text{loc}}} \frac{\delta V_1^{\text{loc}}}{\delta b_i} \mathrm{d}S} + \cancel{\int_{S_I} \Psi_1 \mathsf{n}_k \frac{\partial f_{1k}^{inv}}{\partial U_m} \frac{\partial U_m}{\partial V_2^{\text{loc}}} \frac{\delta V_2^{\text{loc}}}{\delta b_i} \mathrm{d}S} \\ & + \cancel{\int_{S_I} \Psi_1 \mathsf{n}_k \frac{\partial f_{1k}^{inv}}{\partial U_m} \frac{\partial U_m}{\partial V_3^{\text{loc}}} \frac{\delta V_3^{\text{loc}}}{\delta b_i} \mathrm{d}S} + \underbrace{\int_{S_I} \Psi_1 \mathsf{n}_k \frac{\partial f_{1k}^{inv}}{\partial U_m} \frac{\partial U_m}{\partial V_4^{\text{loc}}} \frac{\delta V_4^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_4^{\text{loc}}}}\end{aligned} \tag{3.3.4.43}$$

$$\begin{aligned}\mathcal{S}_2 = & \cancel{\int_{S_I} \Psi_{m+1} \mathsf{n}_k \frac{\partial f_{(m+1)k}^{inv}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_1^{\text{loc}}} \frac{\delta V_1^{\text{loc}}}{\delta b_i} \mathrm{d}S} + \cancel{\int_{S_I} \Psi_{m+1} \mathsf{n}_k \frac{\partial f_{(m+1)k}^{inv}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_2^{\text{loc}}} \frac{\delta V_2^{\text{loc}}}{\delta b_i} \mathrm{d}S} \\ & + \cancel{\int_{S_I} \Psi_{m+1} \mathsf{n}_k \frac{\partial f_{(m+1)k}^{inv}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_3^{\text{loc}}} \frac{\delta V_3^{\text{loc}}}{\delta b_i} \mathrm{d}S} + \underbrace{\int_{S_I} \Psi_{m+1} \mathsf{n}_k \frac{\partial f_{(m+1)k}^{inv}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_4^{\text{loc}}} \frac{\delta V_4^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_4^{\text{loc}}}}\end{aligned} \tag{3.3.4.44}$$

Eliminating the $\text{ABC}^{V_4^{\text{loc}}}$ term leads to the following adjoint boundary condition which is taken into account when computing the boundary adjoint flux

$$\Psi_n \mathsf{n}_k \frac{\partial f_{nk}^{inv}}{\partial U_m} \frac{\partial U_m}{\partial V_4^{\text{loc}}} = \begin{cases} -\dfrac{\partial F_S}{\partial V_4^{\text{loc}}} \text{ , at } S_I^{\text{Obj}} \\ 0 \text{ , at } S_I \setminus S_I^{\text{Obj}} \end{cases} \tag{3.3.4.45}$$

Since a Dirichlet boundary condition is imposed on $\tilde{\nu}$ along the inlet, terms $\mathcal{S}_5$, $\mathcal{S}_7$ and $\mathcal{S}_9$ vanish automatically. The remaining terms are eliminated by setting $\tilde{\nu}_a = 0$.

Outlet Boundaries S_O

For the outlet, let V_ℓ^{loc}, $(\ell = 1, \dots, 3)$ be the quantities extrapolated from the interior of the domain and V_4^{loc} the quantity for which a Dirichlet boundary condition is set. As a result $\frac{\delta V_4^{\text{loc}}}{\delta b_i} = 0$. Taking this into account, terms $\mathcal{S}_1$ and $\mathcal{S}_2$ are written as

$$
\begin{aligned}
\mathcal{S}_1 = & \underbrace{\int\limits_{S_O} \Psi_1 \mathsf{n}_k \frac{\partial f^{inv}_{1k}}{\partial U_m} \frac{\partial U_m}{\partial V_1^{\text{loc}}} \frac{\delta V_1^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_1^{\text{loc}}}} + \underbrace{\int\limits_{S_O} \Psi_1 \mathsf{n}_k \frac{\partial f^{inv}_{1k}}{\partial U_m} \frac{\partial U_m}{\partial V_2^{\text{loc}}} \frac{\delta V_2^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_2^{\text{loc}}}} \\
& + \underbrace{\int\limits_{S_O} \Psi_1 \mathsf{n}_k \frac{\partial f^{inv}_{1k}}{\partial U_m} \frac{\partial U_m}{\partial V_3^{\text{loc}}} \frac{\delta V_3^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_3^{\text{loc}}}} + \cancel{\int\limits_{S_O} \Psi_1 \mathsf{n}_k \frac{\partial f^{inv}_{1k}}{\partial U_m} \frac{\partial U_m}{\partial V_4^{\text{loc}}} \frac{\delta V_4^{\text{loc}}}{\delta b_i} \mathrm{d}S}
\end{aligned}
\tag{3.3.4.46}
$$

$$
\begin{aligned}
\mathcal{S}_2 = & \underbrace{\int\limits_{S_O} \Psi_{m+1} \mathsf{n}_k \frac{\partial f^{inv}_{(m+1)k}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_1^{\text{loc}}} \frac{\delta V_1^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_1^{\text{loc}}}} + \underbrace{\int\limits_{S_O} \Psi_{m+1} \mathsf{n}_k \frac{\partial f^{inv}_{(m+1)k}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_2^{\text{loc}}} \frac{\delta V_2^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_2^{\text{loc}}}} \\
& + \underbrace{\int\limits_{S_O} \Psi_{m+1} \mathsf{n}_k \frac{\partial f^{inv}_{(m+1)k}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_3^{\text{loc}}} \frac{\delta V_3^{\text{loc}}}{\delta b_i} \mathrm{d}S}_{\text{ABC}^{V_3^{\text{loc}}}} + \cancel{\int\limits_{S_O} \Psi_{m+1} \mathsf{n}_k \frac{\partial f^{inv}_{(m+1)k}}{\partial U_\ell} \frac{\partial U_\ell}{\partial V_4^{\text{loc}}} \frac{\delta V_4^{\text{loc}}}{\delta b_i} \mathrm{d}S}
\end{aligned}
\tag{3.3.4.47}
$$

Eliminating the $\text{ABC}^{V_\ell^{\text{loc}}}$, $(\ell = 1, \dots, 3)$ terms gives rise to the following adjoint boundary conditions

$$
\Psi_n \mathsf{n}_k \frac{\partial f^{inv}_{nk}}{\partial U_m} \frac{\partial U_m}{\partial V_\ell^{\text{loc}}} = \begin{cases} -\dfrac{\partial F_S}{\partial V_\ell^{\text{loc}}} \text{ , at } S_O^{\text{Obj}} \\ \qquad 0 \text{ , at } S_O \setminus S_O^{\text{Obj}} \end{cases}
\tag{3.3.4.48}
$$

Eq. 3.3.4.48 is taken into account in the computation of the adjoint boundary flux along the outlet. Since a zero Neumann condition is imposed on $\tilde{\nu}$ along the outlet, $\frac{\delta\left(\frac{\partial \tilde{\nu}}{\partial x_k}\mathsf{n}_k\right)}{\delta b_i} = 0$ and integrals $\mathcal{S}_6$ and $\mathcal{S}_8$ vanish automatically. The remaining integrals are eliminated by setting the multiplier of $\frac{\delta \tilde{\nu}}{\delta b_i}$ equal to $\frac{\partial F_S}{\partial \tilde{\nu}}$ along S_O^{Obj} and equal to zero along $S_O \setminus S_O^{\text{Obj}}$.

Chapter 4

Basics of Shape Parameterization

In Chapter 3, the adjoint method for aerodynamic/hydrodynamic shape optimization was presented. The adjoint method computes the sensitivity derivatives of the objective function w.r.t. a set of design variables. In order to update the shape during the optimization loop, the design variables need to be linked to the geometry through a parameterization method. In addition, the differentiation of the parameterization process provides the means to compute $\frac{\delta x_k}{\delta b_i}$, $(k = 1, \ldots, 3,\ i = 1, \ldots, N)$, that is involved in the expression of sensitivity derivatives and, from now on, is referred to as geometric sensitivities.

The simplest approach to compute such terms is to assume that any mesh node can move independently from the rest of the computational mesh and, thus, its coordinates are design variables. Let x_k^j be the k-coordinate of mesh node j. Then

$$\frac{\delta x_k^j}{\delta x_m^i} = \delta_{ij}\delta_{km} \tag{4.1}$$

The variation of the surface normal vector w.r.t. the coordinates of each node can also be computed by expanding the geometric formulas that lead to the computation of normal vector and applying the chain rule combined with Eq. 4.1.

The field of the objective function sensitivity w.r.t. the coordinates of the mesh nodes $\left(\frac{\delta F}{\delta x_k},\ k = 1, \ldots, 3\right)$ gives the so-called sensitivity map. When using the adjoint SI approach the map is a vector field on the surface to be optimized. Based on differential geometry considerations and assuming smooth surfaces, any node movement tangent to the surface does not change its shape (only its discretization). Thus, the sensitivity map can be plotted as a scalar field containing only $\left(\frac{\delta F}{\delta x_k}\mathsf{n}_k\right)$. When using the adjoint FI approach the sensitivity map is a vector field in the whole flow domain. If a certain mesh morphing technique, linking the movement of internal mesh nodes with the movement of surface nodes, is also taken into account a surface sensitivity map can be computed on the shape under consideration.

Using the sensitivity map during an automatic optimization process has some obvious drawbacks. Firstly, if each node is displaced independently, the resulting boundary surface will most likely lack smoothness. Smoothness is very important since it is associated with manufacturability and endurance of a structure. To overcome the smoothness problem, smoothing of the sensitivity map can be performed. However, a posteriori smoothing of the sensitivity map may potential destroy useful information for the optimization progress and, therefore slow down the optimization algorithm or even cause the optimization to fail. A remedy to this problem is the inclusion of the smoothing process in the computation of the sensitivity derivatives that comprise the sensitivity map. Nevertheless, such an approach adds complexity in the computation of the smoothed sensitivity derivatives and, even though it seems as parameterization-free, it actually implies some type of space parameterization, in the sense that nodes are not allowed to move independently in space but interact with each other. Another drawback of direct, sensitivity map-based node displacement is that imposition of geometric constraints is far from trivial.

Finally, in case of population based optimization methods (such as Evolutionary Algorithms) or, if the cost of the method used to compute the sensitivity derivatives depends on the number of design variables (not the case for the adjoint method), the large number of nodes renders the optimization prohibitively expensive.

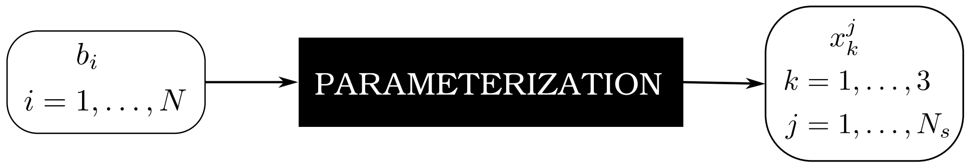


Figure 4.1: Shape parameterization may be considered as a process linking the N design variables to the coordinates of the N_s nodes.

Solution to the aforementioned problems is sought through the use of a parameterization method. One may consider a parameterization as a function/process that links the Cartesian coordinates of a point with a set of parameters b_i which, for optimization problems, also serve as design variables (Fig. 4.1). The desired features from a parameterization method are:

- Smoothness: A parameterization method must be smooth and allow smooth deformations to be performed on the shape/space under consideration.

- Flexibility: A parameterization method must be applicable to a variety of shapes and provide the means for a wide range of shapes to be generated.

- Constraints: Geometric constraints must be easy to implement in the parameterization.

- Intuitive: The less intuitive a parameterization method, the greater the number of design variables needed to describe a shape and the more difficult for the engineer to decide on design variables values and limits.

- Compactness: A parameterization method must be as compact as possible in order to provide low dimensionality design spaces. The higher the dimensionality of the design space, the higher the optimization cost when population based techniques are used and/or the sensitivity derivatives are computed at a cost proportional to the number of design variables. Usually, this feature is contradicting with the need for flexibility and a compromise is sought by most parameterization methods.

- Differentiability: If a parameterization method is to be used within a gradient-based optimization process, this must be differentiable. For analytically differentiable parameterization methods $\frac{\delta x_k^j}{\delta b_i}$ can be computed by closed form relations. On the other hand, for more complicated methods (which cannot be differentiated analytically) an algorithmic differentiation approach must be followed for the computation of the geometric sensitivities.

- Computational efficiency: Any possible shape must be computed with as little computational effort as possible, so that the impact of the parameterization on the overall optimization wall-clock time is minimal.

- CAD compatibility: Ideally, if the parameterization method provides the optimized shape in neutral CAD formats (such as IGES, STEP etc.), used by the majority of commercial and open-source CAD packages, the link between the optimization and the development/production phase of a product is greatly facilitated. Of course, this means that the shape under consideration must be described with a mathematical model that each format uses (typically parametric curves and surfaces and/or solid modeling).

One way to categorize shape parameterization methods is the following. A certain parameterization can describe the shape solely (it is only defined along a shape) or it can describe/parameterize a region in space, in which the shape to be designed is embedded. We will refer to the first class as direct boundary parameterization methods while the ones falling to the second class will be referred to as free-form parameterization methods.

In this chapter, the basic mathematical tools used by both methods are presented. Specifically, Non-Uniform Rational B-Spline (NURBS) curves and surfaces are described, together with a few fundamental algorithms/operations involving them. An extensive analysis of NURBS, as well as algorithms related to NURBS, can be found in the classic book from Piegl and Tiller [149].

4.1 B-Spline Basis Functions

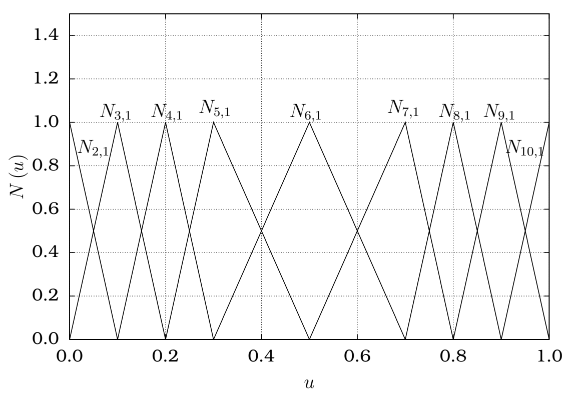
In order to define the B-Spline basis functions, we first need to define a sequence of non-decreasing real values u_i, $i = 0, \ldots, m-1$ called the knots. The knots divide a curve (surface, volume etc. for higher dimensions) in segments linked with a certain degree of continuity. The collection of all u_i is called the knot vector. The ith B-Spline basis function of degree p ($N_{i,p}$) is then defined by the recursive formula

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 \ , & u_i \leq u < u_{i+1} \\ 0 \ , & \text{elsewhere} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u_i}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned} \tag{4.1.1}$$

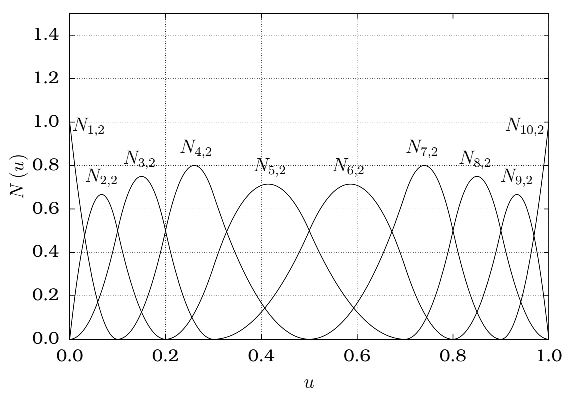
Eq. 4.1.1 may yield the quotient $0/0$ which is defined to be zero [149]. The basis function $N_{i,p}(u)$ is a piecewise polynomial defined on the whole real line. Since the knots in the knot vector are arranged in a non-decreasing order, they need not be distinct. Two or more consecutive knots with the same value constitute a multiple knot. The times a knot is repeated is called the multiplicity of the knot. Multiple knots imply knot spans of zero length. Some important properties of B-Spline basis functions can be found in [149]. Out of them, the most important ones when it comes to parameterization methods, are

- $N_{i,p} \geq 0$ on the whole real line (non-negativity).
- $N_{i,p}(u) \neq 0$ only in the knot span $[u_i, u_{i+p+1})$ (local support).
- $\sum\limits_{j=i-p}^{i} N_{j,p} = 0$ for all $u \in [u_i, u_{i+1})$.
- only $p+1$ basis functions are non-zero in any given knot span, namely $N_{i-p,p}, \ldots, N_{i,p}$ in knot span $[u_i, u_{i+1})$.
- All derivatives of a basis function exist in the interior of a knot span. At a knot of multiplicity s, the basis function is $p-s$ times differentiable.

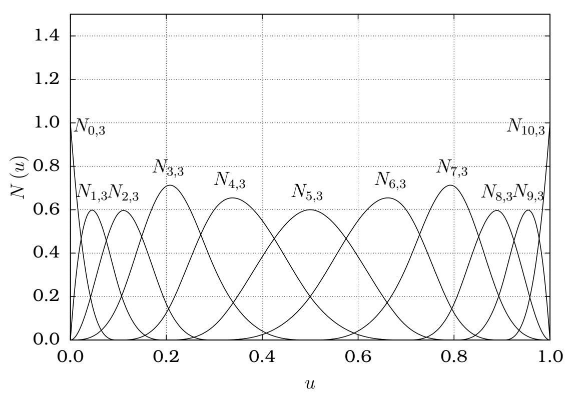
Some examples of basis functions are shown in Fig. 4.2. The effect of adding a multiple knot several times is presented in Fig. 4.3 where it can be seen that the differentiability of the basis functions on the multiple knot position is reduced up to 0th order when the knot multiplicity equals the degree of the basis functions. In these figures, only the non-zero basis functions are presented.



(a)



(b)



(c)

Figure 4.2: B-Spline basis functions of (a) 1st, (b) 2nd and (c) 3rd degree defined on the knot vector $U = (0\ 0\ 0\ 0\ 0.1\ 0.2\ 0.3\ 0.5\ 0.7\ 0.8\ 0.9\ 1\ 1\ 1\ 1)$

Considering the derivatives of the B-Spline basis functions the first derivative, is given by

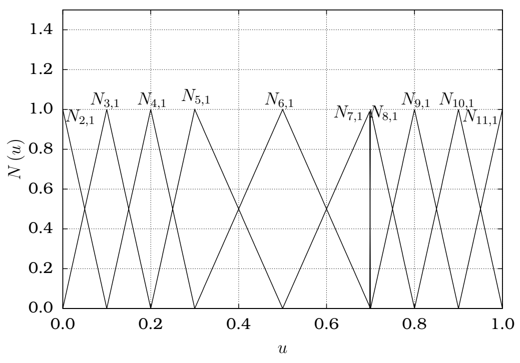
$$N'_{i,p}(u) = \frac{p}{u_{i+p} - u_i} N_{i,p-1} - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1} \qquad (4.1.2)$$

while the kth derivative is computed by

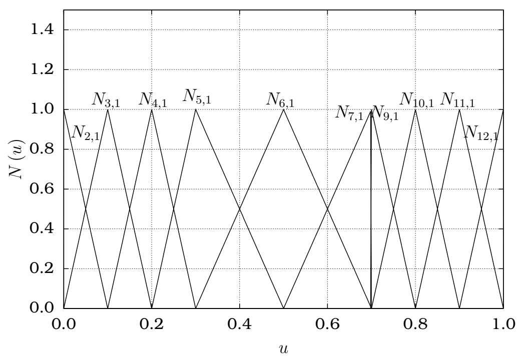
$$N^{(k)}_{i,p} = \frac{p}{p-k} \left(\frac{u - u_i}{u_{i+p} - u_i} N^{(k)}_{i,p-1} + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N^{(k)}_{i+1,p-1} \right) \qquad (4.1.3)$$

For a proof of Eqs. 4.1.2 , 4.1.3 the reader is referred to [149] and [24].

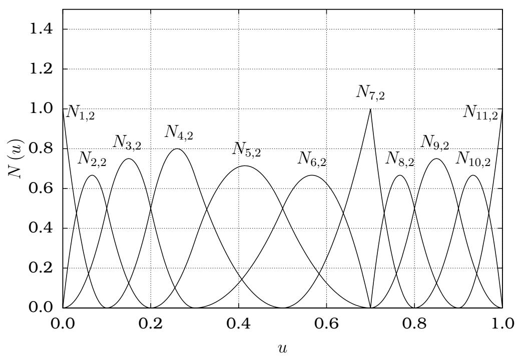
Another discussion on the B-Spline basis functions is related to the categorization of knot vectors. A knot vector can be clamped or unclamped. A clamped knot vector of degree p has the first and last knot with multiplicity equal to $p + 1$.



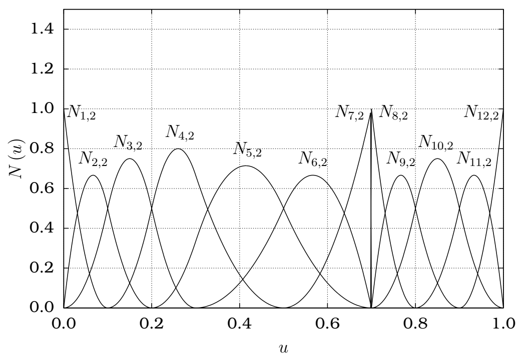
(a) 1st degree



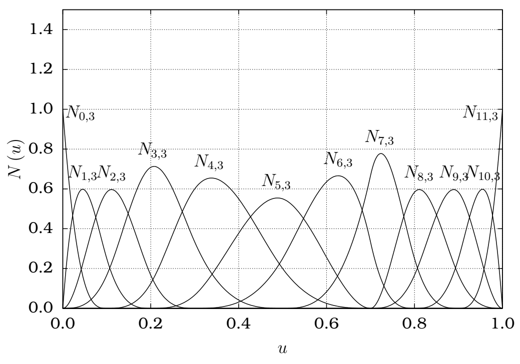
(d) 1st degree



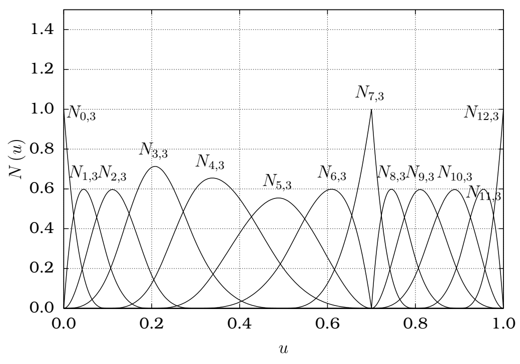
(b) 2nd degree



(e) 2nd degree



(c) 3rd degree



(f) 3rd degree

Figure 4.3: (a)–(c): B-Spline basis functions for the knot vector $U = (0\ 0\ 0\ 0\ 0.1\ 0.2\ 0.3\ 0.5\ 0.7\ 0.7\ 0.8\ 0.9\ 1\ 1\ 1\ 1)$. (d)–(f): B-Spline basis functions for the knot vector $U = (0\ 0\ 0\ 0\ 0.1\ 0.2\ 0.3\ 0.5\ 0.7\ 0.7\ 0.7\ 0.8\ 0.9\ 1\ 1\ 1\ 1)$. The effect of knot multiplicity on the continuity of the basis functions is evident comparing with Fig. 4.2.

If this does not hold, the knot vector is characterized as unclamped. In literature, unclamped knot vectors are also referred to as periodic and clamped ones as non-periodic. Taking into account Eq. 4.1.1 it is evident that, for a given knot vector containing m knots, the basis functions of degree p can be computed only for $u \in [u_p, u_{m-p-1}]$. Unclamped knot vectors are useful for representing closed shapes and merging or extending shapes represented by B-Splines with a certain degree of continuity.

Knot vectors are also characterized by the length of their knot spans (distance between knots in terms of parameter u). An unclamped knot vector is called uniform if all knot spans have equal length. A clamped knot vector is called uniform if all internal knot spans (not the zero length ones in the beginning and end of the knot vector) have the same length. In any other case, the knot vector is characterized as non-uniform.

Finally, a knot vector of the form

$$U = (\underbrace{0,\ldots,0}_{p+1},\underbrace{1,\ldots,1}_{p+1}) \tag{4.1.4}$$

yields the Bernstein polynomials which are alternatively defined as

$$B_{i,p}(u) = \frac{p!}{i!\,(p-i)\,!} u^i (1-u)^{p-i} \tag{4.1.5}$$

4.2 NURBS Curves

If we define a set of n points $\mathbf{P}_i,\ i = 0,\ldots,n-1$ then, using the B-Spline basis functions we can define a parametric curve $\mathbf{C}(u)$ as

$$\mathbf{C}(u) = N_{i,p}(u)\,\mathbf{P}_i\ , \text{ with } u \in [a,b] \text{ and } i = 0,\ldots,n-1 \tag{4.2.1}$$

and $N_{i,p}$ is the ith B-Spline basis function defined on the knot vector

$$U = (\underbrace{a,\ldots,a}_{p+1}, u_{p+1},\ldots,u_{m-p-2},\underbrace{b,\ldots,b}_{p+1})$$

The points $\mathbf{P}_i$ are called control points and the polygon they form, if connected by lines, is called the control polygon.

In what follows bold letters correspond to vector fields in the 2D,3D (or 4D as will be seen hereafter) space. With this convention, the position of the ith control point in the 3D Cartesian space is $\mathbf{P}_i = (x_i\ y_i\ z_i)^T$.

Some of the many properties of B-Spline curves are

- If a clamped knot vector is used, $\mathbf{C}$ interpolates $\mathbf{P}_0$ and $\mathbf{P}_{n-1}$, that is $\mathbf{C}(a) = \mathbf{P}_0$ and $\mathbf{C}(b) = \mathbf{P}_{n-1}$ (endpoint interpolation).

- $\mathbf{C}$ is a piecewise polynomial curve, formed by polynomials of degree p.
- The number of knots m, number of control points n and the degree p are linked through $m = n + p + 1$.
- Any affine transformation can be applied to $\mathbf{C}$ by applying it to its control points (affine invariance).
- $\mathbf{C}$ is contained in the convex hull of the polygon formed by $\mathbf{P}_{i-p}, \ldots, \mathbf{P}_i$ (strong convex hull property).
- Modification of the position of control point $\mathbf{P}_i$ changes only the part of $\mathbf{C}(u)$ for $u \in [\, u_i, u_{i+p+1}\,)$ (local modification property).
- $\mathbf{C}$ has continuity and differentiability at a certain u that corresponds to the continuity and differentiability of the basis functions at this parametric position. Hence, $\mathbf{C}$ is infinitely differentiable in the interior of knot intervals and, only $p - s$ times differentiable at a knot with multiplicity s.

A B-Spline curve defined by $n = p + 1$ control points and a knot vector of the form $U = (\underbrace{0, \ldots, 0}_{p+1}, \underbrace{1, \ldots, 1}_{p+1})$ is a Bezier curve. Thus, Bezier curves are a subset of B-Spline curves.

An extensive discussion on the properties of B-Spline curves can be found in [149]. Fig. 4.4presents a cubic B-Spline with its control polygon. The curve segments influenced by $\mathbf{P}_4$ and $\mathbf{P}_6$ are also shown.

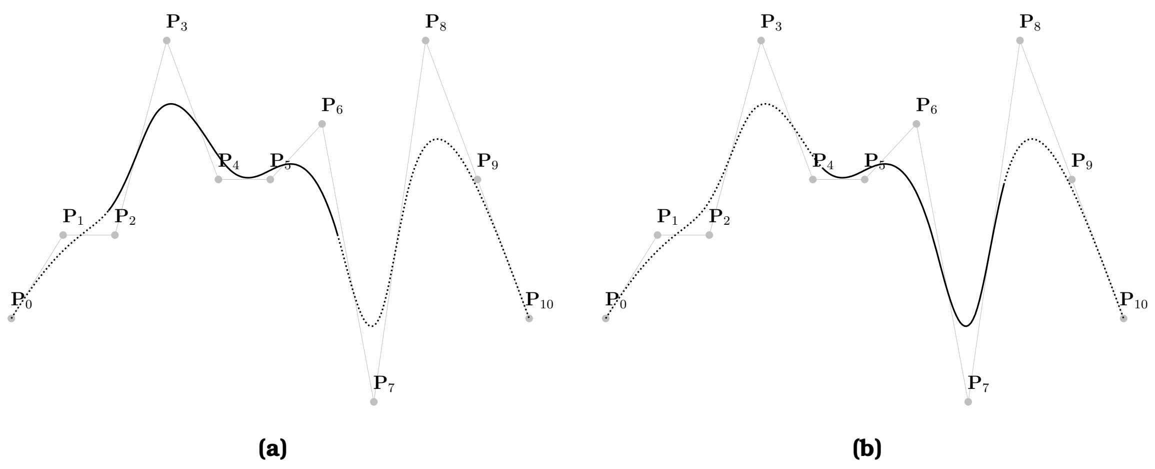


Figure 4.4: B-Spline cubic curve: A cubic curve (black) with its control polygon (gray). (a) The black solid part of the curve is the domain of influence of $\mathbf{P}_4$. (b) The black solid part is the domain of influence of $\mathbf{P}_6$. The curve is non-uniform since it is defined on the knot vector $U = (0, 0, 0, 0, 0.1, 0.2, 0.4, 0.5, 0.7, 0.8, 0.85, 1, 1, 1, 1)$.

Even though non-uniform B-Spline curves offer great flexibility they have a shortcoming when it comes to exactly representing conics. Specifically, the only family of conics that can be represented exactly by polynomial curves, such as B-Splines, is parabolas. Circles, ellipses and hyperbolas are rational curves by definition. By using weights to scale the effect of each control point on the curve, Eq. 4.2.1 becomes

$$\mathbf{C}\left(u\right) = R_{i,p}\left(u\right)\mathbf{P}_i \tag{4.2.2}$$

where, $R_{i,p}$ is the ith rational basis function of degree p given by

$$R_{i,p} = \frac{N_{i,p}\left(u\right)w_i}{N_{j,p}\left(u\right)w_j} \tag{4.2.3}$$

no summation on p and i, while w_i are the weights corresponding to each basis function and control point. Eq. 4.2.2 defines what is called a Non-Uniform Rational B-Spline (NURBS) curve. In that sense NURBS are a superset of B-Spline curves. All the aforementioned properties of B-Spline curves are also valid for NURBS curves.

An alternative expression for defining a NURBS curve stems from the use of homogeneous coordinates. A point $\mathbf{P}\left(x,y,z\right)$ in 3D Euclidean space can also be written as $\mathbf{P}^{\mathbf{w}}\left(wx,wy,wz,w\right)$ in a 4D space (assuming that $w \neq 0$). The coordinates $\left(wx,wy,wz,w\right)$ are called the homogeneous coordinates. $\mathbf{P}$ can be retrieved from $\mathbf{P}^{\mathbf{w}}$ by dividing all coordinates with w. Geometrically, this is equivalent to a perspective projection Π of $\mathbf{P}^{\mathbf{w}}$ with the origin as the vanishing point and $w = 1$ as the projection plane Fig. 4.5.

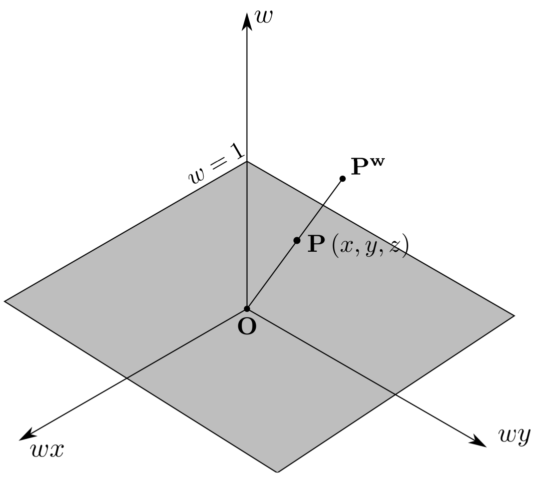


Figure 4.5: Perspective projection Π from 3D homogeneous coordinates to 2D Euclidean space. The same process is generalized to map 4D homogeneous coordinates to 3D Euclidean space.

Using homogeneous coordinates a NURBS curve is expressed as

$$\mathbf{C}^{w}(u) = N_{i,p}(u)\,\mathbf{P^{w}}_{i},\ \ i = 0, \ldots, n-1 \tag{4.2.4}$$

The curve resulting from Eq. 4.2.4 is embedded in the 4D space. Using the perspective projection Π the locus described by Eq. 4.2.4 can be mapped onto the 3D Euclidean space. Homogeneous coordinates are very useful when developing geometric algorithms, since their use allows the treatment of a 3D piecewise rational curve as a 4D piecewise polynomial one.

4.2.1 Derivatives of NURBS Curves

Let $\mathbf{C}^{(k)}(u)$ denote the k-th derivative of a B-Spline curve w.r.t. the parameter u. Differentiating (k) times Eq. 4.2.1 w.r.t. u yields

$$\mathbf{C}^{(k)}(u) = N_{i,p}^{(k)}(u)\,\mathbf{P}_{i} \tag{4.2.1.1}$$

where, the derivatives of the basis functions $N_{i,p}^{(k)}$ are computed using Eq. 4.1.3.

Since the coordinates of the points on the curve are given as univariate functions of the parameter u, $\mathbf{C}^{(1)}(u)$ represents the direction tangent to the curve at parametric position u (Fig. 4.6).

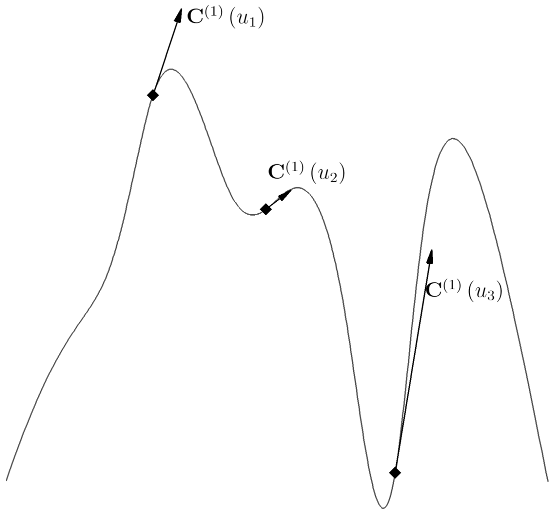


Figure 4.6: The first derivative of a B-Spline curve $\mathbf{C}^{(1)}(u)$ is a vector tangent to the curve at position u. The magnitude of the vector is equal to the parameterization speed, namely $\frac{\mathrm{d}s}{\mathrm{d}u}$ where s would represent a natural or arc-length parameterization of the curve.

The derivatives of a NURBS curve are slightly more complicated to compute. First, a NURBS curve can be expressed as

$$\begin{gathered}\mathbf{C}(u) = \frac{\mathbf{A}(u)}{W(u)} \\ \text{where } \mathbf{A}(u) = W(u)\,\mathbf{C}(u) \quad \text{and} \quad W(u) = N_{i,p}(u)\,w_i\end{gathered} \tag{4.2.1.2}$$

Using the general Leibniz rule, the k-th derivative of $\mathbf{A}(u)$ is obtained as

$$\begin{gathered}\mathbf{A}^k(u) = \binom{k}{j}\mathbf{C}^{(k-j)}(u)\,W^{(i)}(u)\,\delta_{ij} \text{ with } i, j = 0, \ldots, k \\ \text{and } \binom{k}{j} = \frac{k!}{j!\,(k-j)}\end{gathered} \tag{4.2.1.3}$$

In Eq. 4.2.1.3 no summation is implied on k. Rearrangement of the terms in Eq. 4.2.1.3 lead to the following expression for $\mathbf{C}^{(k)}(u)$

$$\mathbf{C}^{(k)}(u) = \frac{\mathbf{A}^{(k)}(u) - \binom{k}{j}\mathbf{C}^{(k-j)}(u)\,W^{(i)}(u)\,\delta_{ij}}{W(u)} \text{ with } i, j = 1, \ldots, k \tag{4.2.1.4}$$

where no summation is implied on k.

4.2.2 Construction of Circular Arcs

Circular arcs are a common geometrical shape in many engineering applications. In the field of turbomachinery, which is the main focus of this thesis, they are used in defining the shape of leading and trailing edges, designing cooling holes on turbine blades e.t.c.. Furthermore, surfaces of revolution are all constructed using the mathematical description of circular arcs. For these reasons, the ability of the mathematical tool, used by a parameterization method, to be able to handle circular arcs is of great importance.

Construction of circles, ellipses and hyperbolas through NURBS is described in [149]. Herein, only the construction of circular arcs with sweep angle $0 \leq \Delta\theta \leq 90°$ is considered. Circular arcs of sweep angle larger than $90°$ are created in pieces and, then, merged together to form a single NURBS curve. For a proof and detailed analysis of the NURBS circle the reader is referred to [148]. Let us assume the circle centered at the origin $\mathbf{O}$ and $\hat{\mathbf{i}}_1$, $\hat{\mathbf{i}}_2$ the unit vectors for the x, and y axis respectively, and sweeping from angle θ_S to θ_E with radius r. The two endpoints (which are also control points) are given by

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{O} + r\cos\theta_S \hat{\mathbf{i}}_1 + r\sin\theta_S \hat{\mathbf{i}}_2 \\ \mathbf{P}_2 &= \mathbf{O} + r\cos\theta_E \hat{\mathbf{i}}_1 + r\sin\theta_E \hat{\mathbf{i}}_2 \end{aligned} \tag{4.2.2.1}$$

where both control points are associated with a unit weight ($w_0 = w_2 = 1$). The circle's tangent directions $\hat{\mathbf{t}}_0$, $\hat{\mathbf{t}}_2$, at points $\mathbf{P}_0$ and $\mathbf{P}_2$ respectively, are given as

$$\begin{aligned} \hat{\mathbf{t}}_0 &= -\sin\theta_S \hat{\mathbf{i}}_1 + \cos\theta_S \hat{\mathbf{i}}_2 \\ \hat{\mathbf{t}}_2 &= -\sin\theta_E \hat{\mathbf{i}}_1 + \cos\theta_E \hat{\mathbf{i}}_2 \end{aligned} \tag{4.2.2.2}$$

The Euclidean coordinates of the intermediate control point $\mathbf{P}_1$ are given by computing the tangents intersection. Finally, the weight associated with $\mathbf{P}_1$ is $w_1 = \cos\left(\frac{\Delta\theta}{2}\right)$. The control points $\mathbf{P}_0^w$, $\mathbf{P}_1^w$ and $\mathbf{P}_2^w$ form a rational Bezier curve that represents the target circular arc. A graphical representation is shown in Fig. 4.7.

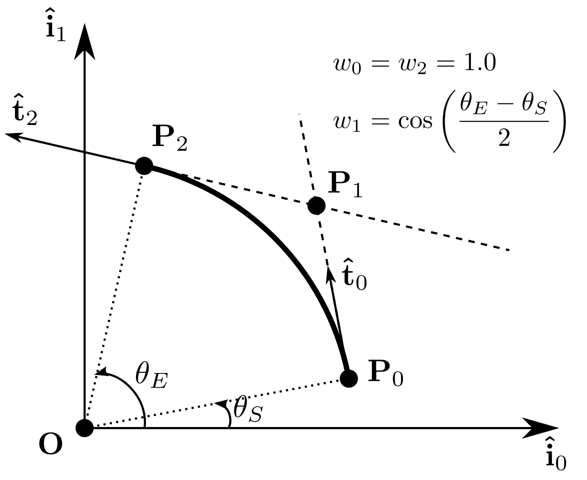


Figure 4.7: Construction of circular arc sweep less than 90° using NURBS curves.

4.3 NURBS Surfaces

NURBS curves presented in Section 4.2 can be considered as a mapping of a straight line segment (represented by the single parameter u) to the multi-dimensional Euclidean space of dimensionality corresponding to the this of the control points. Similarly, a surface may be considered as the mapping of a part of a planar area (represented by two parameters u and v) to the multi-dimensional space.

B-Spline and NURBS surfaces are tensor product surfaces. Tensor product surfaces follow a bidirectional curve scheme, meaning that they use bivariate basis functions which are the product of univariate basis functions defined on the two distinct parametric directions.

Considering a $m \times n$ grid of control points $\mathbf{P}_{ij}$, ($i = 0, \ldots, n-1$ and $j = 0, \ldots, m-1$), a B-Spline surface $\mathbf{S}(u,v)$ is expressed as

$$\mathbf{S}(u,v) = N_{i,p_u}(u)\, N_{j,p_v}(v)\, \mathbf{P}_{i,j} \tag{4.3.1}$$

where, N_{i,p_u} (N_{i,p_v}) are the basis functions of degree p_u (p_v) defined on the knot vectors U (V respectively) as given by Eq. 4.1.1. An example of a B-Spline surface is presented in Fig. 4.8.

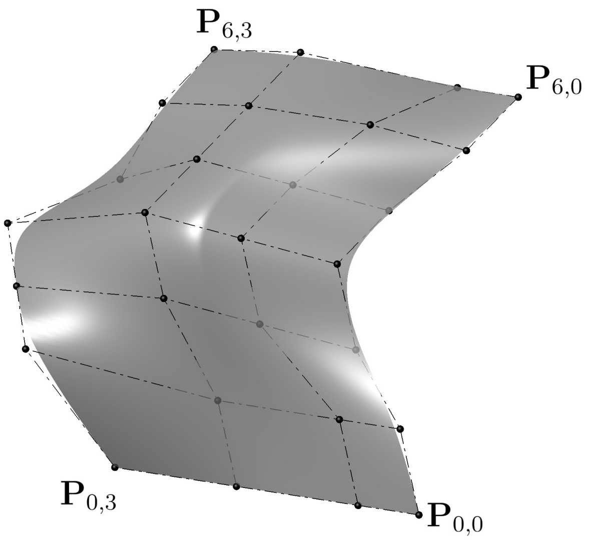


Figure 4.8: Example of a polynomial B-Spline surface defined by a 7×4 grid of control points and basis functions of degree 3 and 2, respectively, defined on uniform knot vectors. Some of the control points lay below the surface and, for this reason, the surface is rendered slightly transparent.

As with curves, if each control point is associated with a weight, a NURBS Surface is created. A NURBS surface is, thus, defined as

$$\mathbf{S}(u,v) = R_{ij,p_u p_v}(u,v)\, \mathbf{P}_{i,j} \tag{4.3.2}$$

where, $R_{ij,p_u p_v}$ is the $\{i,j\}$-th rational basis function of degree $\{p_u, p_v\}$ defined as

$$R_{ij,p_u p_v} = \frac{N_{i,p_u}(u)\, N_{j,p_v}(v)\, w_{ij}}{N_{k,p_u}(u)\, N_{\ell,p_v}(v)\, w_{k\ell}} \tag{4.3.3}$$

where no summation is implied on i, j, p_u and p_v. Using homogeneous coordinates a NURBS surface is expressed as

$$\mathbf{S}^w = N_{i,p_u} N_{j,p_v} \mathbf{P^w}_{ij}, \; i = 0, \ldots, n-1, \; j = 0, \ldots, m-1 \tag{4.3.4}$$

The main properties of NURBS curves (endpoint interpolation, local support, strong convex hull property, differentiability e.t.c.) are also extended to NURBS surfaces. Clearly, freezing one of the two parametric coordinates yields an isoparametric curve. For example the isoparametric curve $\mathbf{C}^W_{u^*}(v)$ at $u = u^*$ obtained by the surface $\mathbf{S}^w(u,v)$ is expressed as

$$\begin{aligned} \mathbf{C}^w_{u^*}(v) &= N_{j,p_v}(v)\, \mathbf{Q}^w j \\ \text{with } \mathbf{Q}^w_j &= N_{i,p_u}(u^*)\, \mathbf{P^w}_{ij} \end{aligned} \tag{4.3.5}$$

and a similar equation hold for $\mathbf{C}^W_{v^*}(u)$ iso-v curves.

Several basic families of surfaces can be constructed as NURBS surfaces. Some examples are described in the following sections.

4.3.1 Derivatives of NURBS Surfaces

By differentiating Eq. 4.3 k times w.r.t. u and ℓ times w.r.t. v, the following expression results for the derivative of a B-Spline surface

$$\frac{\partial^{k+\ell}}{\partial^k u \partial^\ell v} \mathbf{S}(u,v) = N^{(k)}_{i,p_u}(u)\, N^{(\ell)}_{j,p_v}(v)\, \mathbf{P}_{ij} \tag{4.3.1.1}$$

A rational B-Spline (NURBS) surface can be expressed as

$$\begin{aligned} \mathbf{S}(u,v) &= \frac{W(u,v)\,\mathbf{S}(u,v)}{W(u,v)} = \frac{\mathbf{A}(u,v)}{W(u,v)} \\ \text{where } W(u,v) &= N_{i,p_u}(u)\, N_{j,p_v}(v)\, w_{ij} \end{aligned} \tag{4.3.1.2}$$

For brevity, in what follows in this section, $\mathbf{S}$ will be used instead of $\mathbf{S}(u,v)$, $\mathbf{A}$ instead of $\mathbf{A}(u,v)$ and W instead of $W(u,v)$. Moreover, the partial derivatives $\frac{\partial^{k+\ell}\mathbf{S}}{\partial^k u \partial^\ell v}$, $\frac{\partial^{k+\ell}\mathbf{A}}{\partial^k u \partial^\ell v}$ and $\frac{\partial^{k+\ell}W}{\partial^k u \partial^\ell v}$ will shortly be referred to as $\mathbf{S}^{(k,\ell)}$, $\mathbf{A}^{(k,\ell)}$ and $W^{(k,\ell)}$, respectively. Application of the general Leibniz's rule twice leads to

$$\begin{aligned}
\mathbf{A}^{(k,\ell)} &= \left(\mathbf{A}^{(k,0)}\right)^{(0,\ell)} = \left[(W\mathbf{S})^{(k,0)}\right]^{(0,\ell)} \\
\Rightarrow \mathbf{A}^{(k,\ell)} &= W^{(0,0)}\mathbf{S}^{(k,\ell)} + \binom{k}{j}\mathbf{S}^{(k-j,\ell)}W^{(i,0)}\delta_{ij} + \binom{\ell}{q}\mathbf{S}^{(k,\ell-q)}W^{(0,r)}\delta_{qr} \\
&+ \binom{k}{j}\binom{\ell}{q}\mathbf{S}^{(k-j,\ell-q)}W^{(i,r)}\delta_{ij}\delta_{qr}
\end{aligned} \tag{4.3.1.3}$$

where no summation is implied on k and ℓ, $i,j = 1,\ldots,k$ and $q,r = 1,\ldots,\ell$. Rearranging terms in Eq. 4.3.1.3 $\mathbf{S}^{(k,\ell)}$ is expressed as

$$\begin{aligned}
\mathbf{S}^{(k,\ell)} = \frac{1}{W}\Big[\mathbf{A}^{(k,\ell)} &- \binom{k}{j}\mathbf{S}^{(k-j,\ell)}W^{(i,0)}\delta_{ij} - \binom{\ell}{q}\mathbf{S}^{(k,\ell-q)}W^{(0,r)}\delta_{qr} \\
&- \binom{k}{j}\binom{\ell}{q}\mathbf{S}^{(k-j,\ell-q)}W^{(i,r)}\delta_{ij}\delta_{qr}\Big]
\end{aligned} \tag{4.3.1.4}$$

Similar to NURBS curves the first derivatives of a B-Spline surface represent the direction of the tangent vectors on the surface. Specifically, $\mathbf{S}^{(1,0)}(u,v)$ results in the tangent vector on the surface that is also tangent to the iso-v line passing through (u,v), and $\mathbf{S}^{(0,1)}(u,v)$ results in the tangent vector of $\mathbf{S}$ that is also tangent to the iso-u line passing through (u,v). Finally, the normal to the surface vector $\mathbf{N}$ on (u,v) can be computed as

$$\mathbf{N} = \frac{\mathbf{S}^{(1,0)} \times \mathbf{S}^{(0,1)}}{\left|\mathbf{S}^{(1,0)} \times \mathbf{S}^{(0,1)}\right|} \tag{4.3.1.5}$$

4.3.2 Bilinear Surfaces

Given four points $\mathbf{P}_{0,0}$, $\mathbf{P}_{1,0}$, $\mathbf{P}_{0,1}$ and $\mathbf{P}_{1,1}$ in the 3D Cartesian space, a surface that linearly interpolates the four lines defined by these points, namely $\{\mathbf{P}_{0,0}\mathbf{P}_{1,0}\}$, $\{\mathbf{P}_{1,0}\mathbf{P}_{1,1}\}$, $\{\mathbf{P}_{1,1}\mathbf{P}_{0,1}\}$ and $\{\mathbf{P}_{0,1}\mathbf{P}_{0,0}\}$ is sought. Such a surface is easily constructed as

$$\mathbf{S}(u,v) = N_{i,1}(u)\,N_{j,1}(v)\,\mathbf{P}_{ij} \tag{4.3.2.1}$$

with $i,j = 0,1$ and $N_{i,1}$, $N_{j,1}$ the 1st degree B-Spline basis functions defined on the knot vectors $U = V = (0\ 0\ 1\ 1)$.

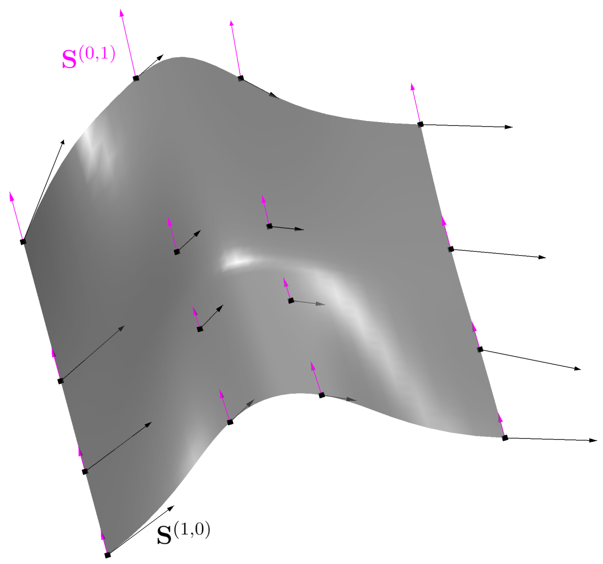


Figure 4.9: The first order partial derivatives of a NURBS surface define vectors tangent to the surface in the two parametric directions. $\mathbf{S}^{(1,0)}$ is tangent to the iso-v lines while $\mathbf{S}^{(0,1)}$ is tangent to the iso-u ones.

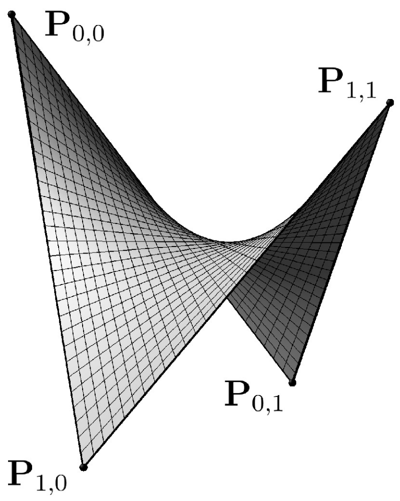


Figure 4.10: Example of a bilinear NURBS surface constructed from the four corner control points. Isoparametric curves are also drawn on the surface, in the form of a structured grid.

4.3.3 General Cylinder Surfaces

A general cylinder surface is obtained by sweeping a profile curve $\mathbf{C}^w(u)$ by a distance d along the unit vector $\hat{\mathbf{D}}^w$. Point $\hat{\mathbf{D}}^w$ results from a 3D point $\hat{\mathbf{D}}$ by mapping it to homogeneous space with $w = 1$. Such a surface is computed as

$$\begin{aligned} &\mathbf{S}^w(u,v) = N_{i,p_u}(u)\, N_{j,1}(v)\, \mathbf{Q}^w_{ij} \\ &\text{with } \mathbf{Q}^w_{i0} = \mathbf{P}^w_i \text{ and } \mathbf{Q}^w_{i1} = \mathbf{P}^w_i + d\hat{\mathbf{D}}^w \end{aligned} \tag{4.3.3.1}$$

where $\mathbf{P}^w_i$ are the control points defining the profile curve $\mathbf{C}^w$, N_{i,p_u} the B-Spline basis functions of degree p_u defined on the knot vector U and $N_{j,1}$ the basis functions of 1st degree defined on the knot vector $V = (0\ 0\ 1\ 1)$. An example of a general cylinder surface is show in Fig. 4.11.

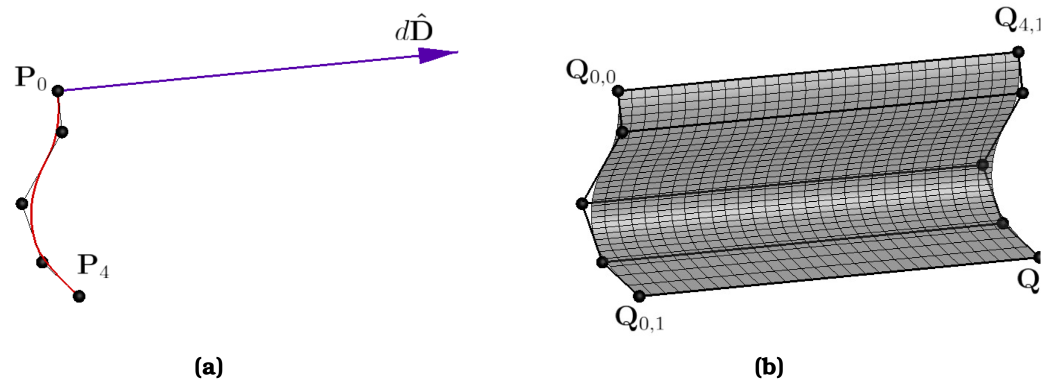


Figure 4.11: (a) A profile curve, its control points $\mathbf{P}_i$ and the displacement vector are used to create a general cylinder surface. (b) The resulting general cylinder surface and its control points $\mathbf{Q}_{ij}$. On both figures only the first and last control points are labeled. In addition, the base curve is chosen to be polynomial for ease of graphic representation.

4.3.4 Surfaces of Revolution

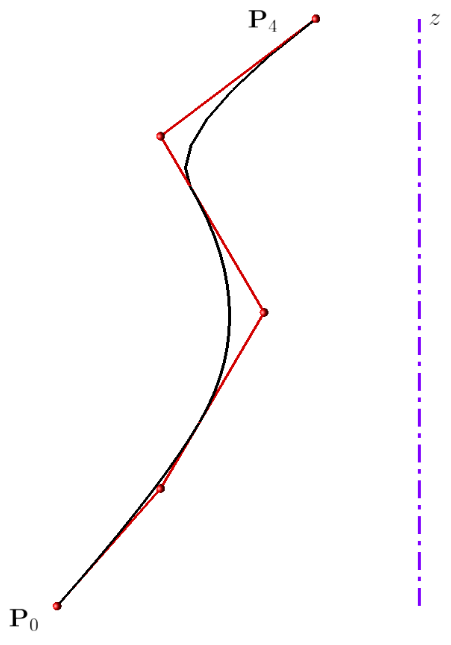
Surfaces of revolution are very useful, especially when dealing with turbomachinery geometries due to the axisymmetric configurations encountered. They are used extensively in the turbomachinery specific parameterization methods that are developed in the following chapters.

A surface of revolution is defined by revolving a curve $\mathbf{G}^w(u)$, called the generatrix, around a given axis. For the sake of simplicity and, without lacking generality, the axis around which curve $\mathbf{G}^w$ is revolved is assumed to coincide with the z-axis passing through the origin O. Let us also assume that the sweep angle of the surface of revolution is $0 \leq \Delta\theta \leq 90°$. The surface of revolution exhibits the following characteristics

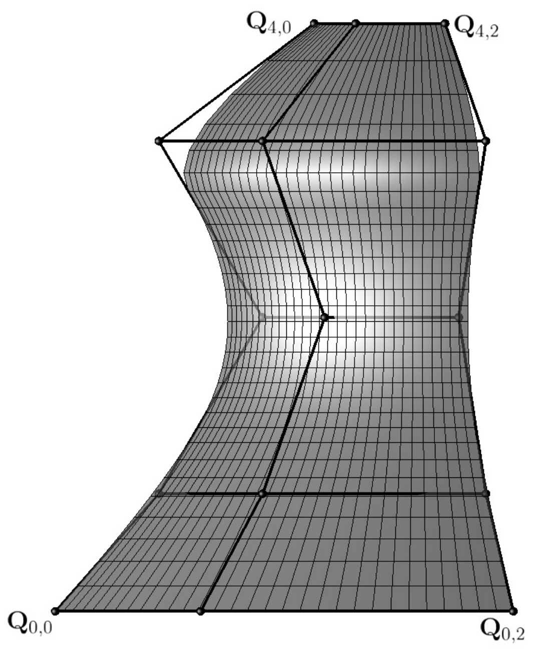
- For a fixed value $v = v^*$ the isoparametric curve $\mathbf{C}^w_{v^*}(u)$ is the generatrix rotated by a certain angle around z corresponding to the value v^*.

- For a fixed value $u = u^*$ the isoparametric curve $\mathbf{C}^w_{u^*}(v)$ is a circular arc lying on a plane perpendicular to axis z. The intersection of this plane with the z-axis is the centre of this arc.

The surface of revolution is created by creating circular arcs starting from each control point of the generatrix, centered at the projection of each respective control point on z-axis. As for circular arcs, surfaces of revolution sweeping more than $90°$ are created by merging surfaces sweeping less than $90°$ degrees. For brevity, we refrain from the presentation of the full algorithm of constructing a surface of revolution using NURBS, which can be found in several texts (e.g. [149]). It is important to note that even if the generatrix is a piecewise polynomial curve, the resulting surface of revolution is still a rational tensor product surface. An example of a surface of revolution as well as its generatrix is shown in Fig. 4.12.



(a)



(b)

Figure 4.12: (a) Example of a generatrix curve and the axis that define a surface of revolution. (b) Resulting NURBS surface and its control polygon. In both figures, only some of the control points are labeled.

4.3.5 NURBS Parameterization Examples

In this section some simple parameterization examples using NURBS curves and surfaces are presented. Examples of parameterization of more complex geometries are presented in the following chapters where more advanced parameterization techniques are introduced.

The first example presents the parameterization of an airfoil shape using two NURBS curves, namely one for the pressure and one for the suction side of the airfoil (Fig. 4.13). The effect of modifying a control point position and its weight is presented in Fig. 4.14.

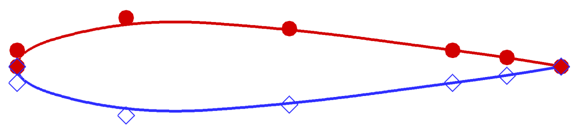
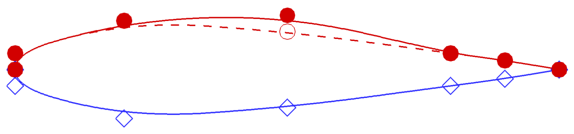
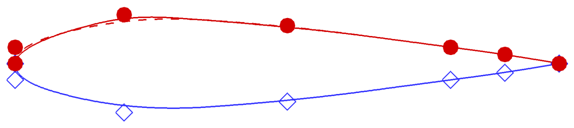


Figure 4.13: Parameterization of the two sides of an airfoil using two NURBS curves. In order to keep tangent continuity on the leading edge, the three control points in the leading edge area must remain co-linear.

The second example is a 3D flying wing configuration. The wing planform is of cranked arrow type. The pressure and suction sides are parameterized by different NURBS surfaces. The control grid is 7×5 and results in a total of 35 control points. The wing and its parameterization are presented in Fig. 4.15.

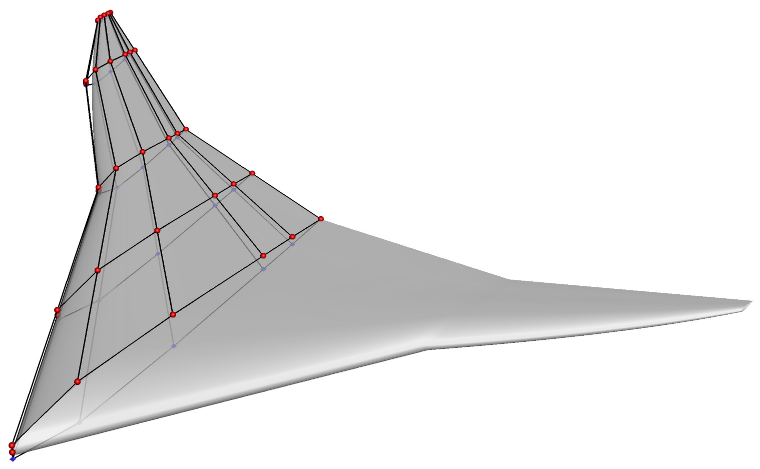


(a)

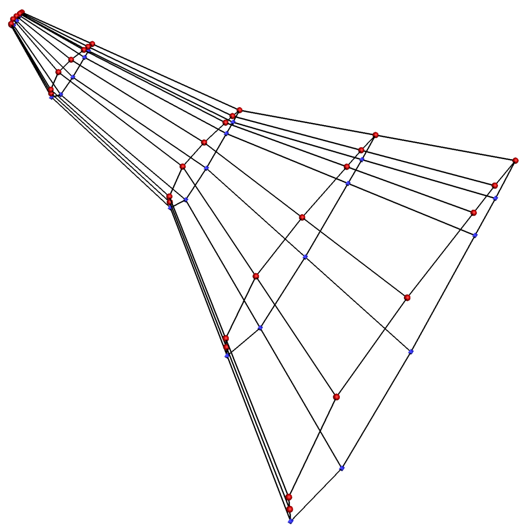


(b)

Figure 4.14: (a) Effect of displacing the 4th control point on the suction side. The original shape is shown with dashed line and empty control points, while the modified one with solid line and filled control points. (b) Effect of modification of the weight of the 3rd suction side control point on the suction side shape. The original shape is shown with dashed line, while the modified one is displayed with solid line. On the original shape all control points have $w = 1$ while on the modified one the third control point has $w = 2$.



(a)



(b)

Figure 4.15: Shape parameterization of a flying wing configuration with cranked arrow planform. (a) The wing geometry. Only half of the flying wing is parameterized due to symmetry. The pressure and suction sides are represented by separate NURBS surfaces. (b) The control points creating the half-wing geometry. Even though a crank exists (discontinuity of the tangent to the surface), a multiple knot in the spanwise direction allows the full span of the half-wing to be modeled using a single quadratic (in both parametric directions) NURBS surface.

Chapter 5

Geometric Modeling of Turbomachinery Components

In Chapter 4, the basics of shape parameterization techniques have been presented. Geometric modeling and parameterization of turbomachinery components is a challenging field of application of shape parameterization techniques, due to the complexity of the components, presence of axisymmetric features and high sensitivity of the flow to the changes in shape.

Directly using NURBS surfaces to model a compressor or turbine blade, for example, would fail to provide the required compactness, while the effects of control point variations on the geometry would not be intuitive. In addition, geometric requirements such as thickness extrema, leading and trailing edge shape, tip gaps etc. would be extremely difficult to handle.

Additionally, altering the blade shape can, potentially lead to surfaces that are either detached from or intersect the hub and shroud. These geometries are invalid (if not corrected properly), which can become a major performance bottleneck when using gradient-free population based optimization methods such as Evolutionary Algorithms (EAs). Thus, modeling a blade directly using NURBS surfaces may also harm computational efficiency.

The aforementioned problems are even more pronounced when dealing with mixed and radial flow components. To circumvent these shortcomings, a turbomachinery-oriented parameterization technique is developed within this thesis. This technique is implemented in the software GMTurbo, using an object oriented programming model and C++ as a programming language. Most of the operations involved in geometry generation by GMTurbo are based on NURBS theory (presented in Chapter 4), ensuring thus CAD compatibility.

In addition, since turbomachinery blades are typically arranged in multiple row configurations, GMTurbo provides the option for designing multiple rows simultaneously, ensuring proper assemblage among them.

In this chapter, the process of designing turbomachinery components with

GMTurbo is presented, together with the inputs that define a geometry and may serve as design variables in a shape optimization process. At the end of the chapter several example geometries are presented.

At this point it must be noted that, apart from using GMTurbo during an optimization, the engineering significance of the parameters involved renders GMTurbo a useful tool even for initial and preliminary turbomachinery component design.

5.1 Meridional Section Design

The first step that defines a turbomachine geometry is the meridional section shape. Different types of components that can be designed by GMTurbo are classified as follows:

- Shrouded blade rows, namely compressor/pump/turbine rotor/runner or stator/bladed diffuser rows.
- Blade rows with tip clearance, where a tip gap is present either between the hub and the blade (for stators) or between the shroud and the blade (for rotors).
- Open blade rows, where the shroud surface is absent (like helicopter rotors, aircraft propellers and horizontal axis wind turbine rotors).
- Duct components, such as inlet or outlet ducts, nozzles etc. in which the hub surface is still present.
- Duct components that do not include the hub surface (like draft tubes of hydraulic turbines and aircraft engine nozzles).
- Components where only the hub surface is present and exposed to external flow (like jet engine inlet and exhaust cones).

Some examples of components that can be designed and parameterized using GMTurbo are presented in Fig. 5.1.

The meridional section geometry definition and parameterization is described for the case of a blade row with tip-clearance since this is the most complex component in terms of meridional section shape. Without loss of generality it will be assumed that the axis of rotation of the machine (or, else, axis of axisymmetry for hub, shroud and tip surfaces) is the z axis and consequently, the meridional plane is the (z, r) plane, with $r = \sqrt{x^2 + y^2}$. The hub, shroud and tip surfaces are defined by their generatrices, which are specified as NURBS curves on the meridional (z, r) plane. An upstream and downstream boundary curve is also needed for each component. These two curves must intersect the generatrices of

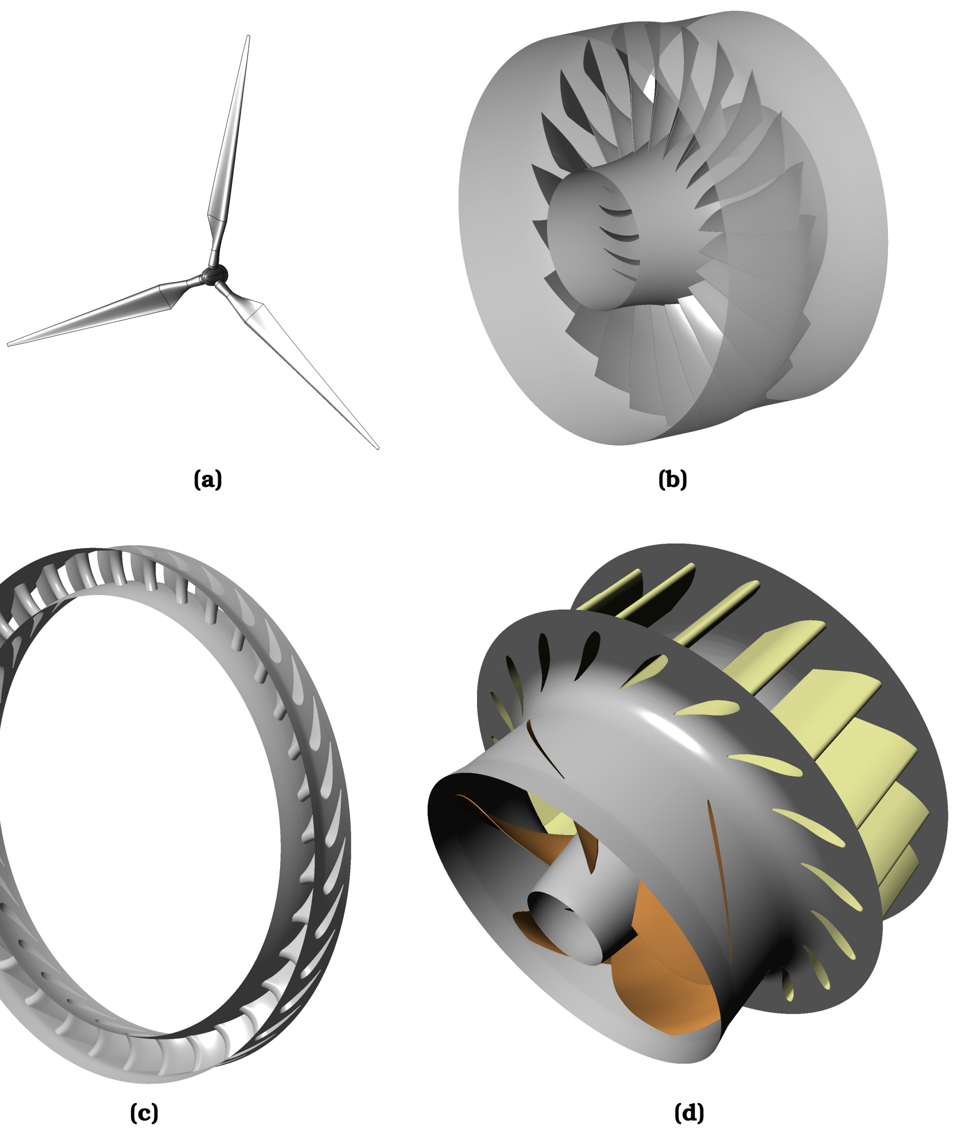


Figure 5.1: Example geometries generated by GMTurbo. (a) Rotor of an horizontal axis wind turbine. This case is treated by GMTurbo as an open rotor component since there is no shroud surface. (b) A transonic fan rotor with tip clearance. The blades, hub, shroud and tip surfaces are modeled by GMTurbo. (c) A high-pressure turbine nozzle guide vane row. The blades are attached to both hub and shroud surfaces. (d) A propeller type hydraulic turbine stage. The whole assembly (guide vanes and the runner) are modeled by GMTurbo.

hub, shroud and tip. The part of the geometry that lies outside the area bounded by the hub and shroud generatrices and the upstream and downstream boundary curves is not modeled at all. Next, the meridional curves defining the leading and trailing edge of the blade are specified again as NURBS curves.

Since multiple components can be modeled simultaneously by GMTurbo, the hub and shroud generatrices need not be distinct for each component.

If, however, they are specified separately for each component, then proper assembly of the generatrices is checked to avoid geometry gaps between components. In Fig. 5.2, an example of the meridional shape definition for a compressor row is presented. The meridional section definition is the basis for constructing the blade since all blade sections are constructed on isospan surfaces of revolution defined between hub and shroud by generatrices interpolated from the hub, shroud and tip ones. If the hub, shroud and tip generatrices are of the same degree and defined on the same knot vector, then an intermediate generatrix can be computed by directly interpolating the control points. In case the generatrices are NURBS curves of either different degree or defined on different knot vectors, they are brought to the least common multiple of the degrees and to common knot vectors, using knot insertion and degree elevation algorithms for NURBS curves. Then, intermediate generatrices are computed by control point interpolation.

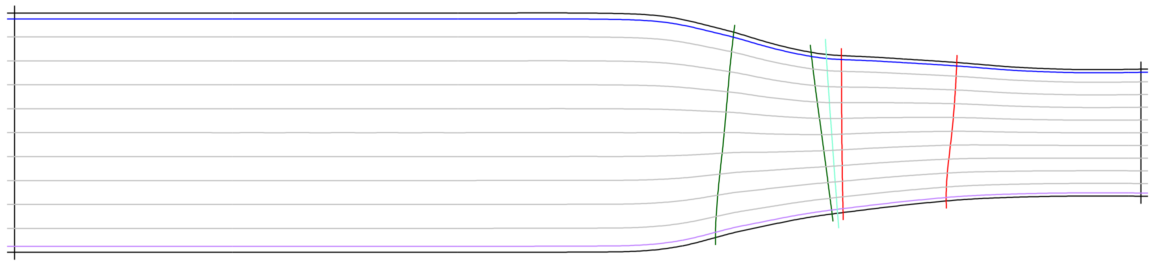


Figure 5.2: Definition of the meridional section of a compressor row. Hub and shroud generatrices, as well as upstream (left) and downstream (right) curves are shown in black. The rotor leading and trailing edge curves are shown in dark green, while the stator ones in red. The generatrix defining the rotor tip surface (tip gap between rotor blade and shroud) is in blue color. The generatrix defining the stator tip surface (tip gap between stator blade and hub) is drawn with purple color. The light green line between the rotor trailing edge and the stator leading edge shows the positioning and shape of the rotor/stator interface surface (auxiliary surface for mesh generation). Finally, intermediate generatrices, arising from interpolating the hub and shroud generatrices, are drawn in gray. It is important that the blade and upstream/downstream boundary curves clearly intersect the generatrices, since any geometrical entity lying outside these boundaries is not modeled.

5.2 Conformal Mapping of Surfaces of Revolution

Since a unified algorithm is sought for all kinds of turbomachinery, namely axial, radial and mixed flow, instead of using a Cartesian coordinate system to create blade sections, these are defined on different surfaces of revolution along the blade span. The mathematical tool that enables a blade section to be compactly and easily drawn directly onto a surface of revolution is the conformal mapping.

A diffeomorphism between two manifolds, with their metrics being conformally equivalent is called a conformal map. Two metrics g and h are said to be conformally equivalent if there exists a positive function c, such that $g = ch$, where c is the so-called conformal factor. Conformal mapping refers to the process of building a conformal map. Now, assume two curves $\mathbf{C_1}$ and $\mathbf{C_2}$ lying on surface $\mathbf{S}$ intersecting at point $\mathbf{P}$ with signed angle ζ. Let $\mathbf{C}_1'$, $\mathbf{C}_2'$ be the mapping of these curves on surface $\mathbf{S}'$, and their intersection with angle ζ' being located at point $\mathbf{P}'$. If the mapping $\mathbf{S} \mapsto \mathbf{S}'$ is conformal, then $\zeta = \zeta'$ (Fig. 5.3).

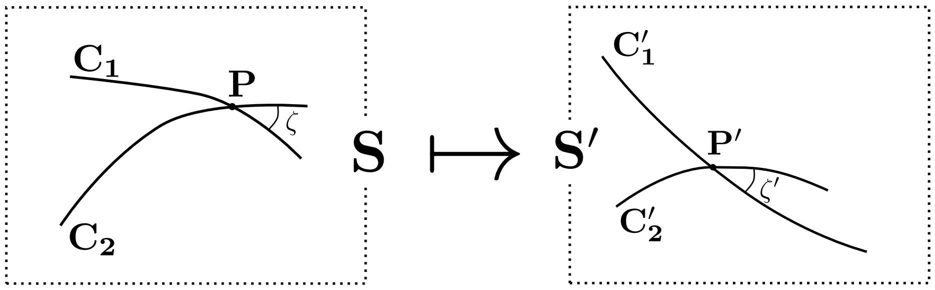


Figure 5.3: Conformal map of a surface $\mathbf{S}$ to $\mathbf{S}'$. The angles between directed curves are preserved throughout the mapping $(\zeta = \zeta')$.

The conformal map that is of interest in the design process of a turbomachinery blade is the mapping of a surface of revolution $\mathbf{S}\,(r\cos\theta, r\sin\theta, z)$ to a plane $\mathbf{\Pi}\,(m,\theta)$. Since $r = r\,(u)$, $z = z\,(u)$ and $m = m\,(u)$, both $\mathbf{S}$ and $\mathbf{\Pi}$ are bivariate vector valued functions with variables u and θ and domain of definition $[u_a, u_b] \times [-\infty, \infty]$, where $[u_a, u_b]$ is the parameter range in which the generatrix of the surface of revolution is defined. The mapping is defined as

$$\mathbf{S}\,(r\cos\theta, r\sin\theta, z) \mapsto \mathbf{\Pi}\,(m,\theta)\,, \quad \text{with} \quad m\,(u) = \int_{u_a}^{u} \frac{\sqrt{\dot{r}^2\,(t) + \dot{z}^2\,(t)}}{r\,(t)} \mathrm{d}t \tag{5.2.1}$$

Proposition 1. The mapping introduced by Eq. 5.2.1 is conformal [155].

Proof. The coefficients of the first fundamental form of $\mathbf{S}$ and $\mathbf{\Pi}$ are

$$\begin{aligned} E_{\mathbf{S}} &= \frac{\partial \mathbf{S}}{\partial u} \cdot \frac{\partial \mathbf{S}}{\partial u} = \dot{r}^2 + \dot{z}^2 & E_{\mathbf{\Pi}} &= \frac{\partial \mathbf{\Pi}}{\partial u} \cdot \frac{\partial \mathbf{\Pi}}{\partial u} = \frac{\dot{r}^2 + \dot{z}^2}{r^2} \\ F_{\mathbf{S}} &= \frac{\partial \mathbf{S}}{\partial u} \cdot \frac{\partial \mathbf{S}}{\partial \theta} = 0 & F_{\mathbf{\Pi}} &= \frac{\partial \mathbf{\Pi}}{\partial u} \cdot \frac{\partial \mathbf{\Pi}}{\partial \theta} = 0 \\ G_{\mathbf{S}} &= \frac{\partial \mathbf{S}}{\partial \theta} \cdot \frac{\partial \mathbf{S}}{\partial \theta} = r^2 & G_{\mathbf{\Pi}} &= \frac{\partial \mathbf{\Pi}}{\partial \theta} \cdot \frac{\partial \mathbf{\Pi}}{\partial \theta} = 1 \end{aligned} \tag{5.2.2}$$

From Eq. 5.2.2 it is evident that $E_{\mathrm{S}} = r^2 E_{\mathbf{\Pi}}$, $F_{\mathrm{S}} = r^2 F_{\mathbf{\Pi}}$ and $G_{\mathrm{S}} = r^2 G_{\mathbf{\Pi}}$. □

The mapping introduced by Eq. 5.2.1 preserves the angles of intersecting curves between S and **Π**. Hence, a shape with desirable angles can be drawn in the 2D plane **Π** and mapped back to S using the inverse mapping, by preserving all the prescribed angles. The inverse mapping of Eq. 5.2.1 requires the inversion of the function $m\left(u\right)$ as $u\left(m\right)$, which cannot be done analytically. To overcome this obstacle, a look-up table is created when performing the direct mapping. Then, the inverse mapping is performed by searching the look-up table and interpolating between stored pairs of (m, u). The interpolation can be either linear or higher order. For higher order interpolation the mapping curve M is constructed as a NURBS curve $\mathrm{M} : [u_a, u_b] \to [m_a, m_b]$ interpolating the known (m, u) pairs. Computation of u that corresponds to a given m is performed by point inversion on curve M. Once u is computed, the evaluation of S can also be completed.

Proposition 2. The infinitesimal lengths $\mathrm{d}\ell_{\mathrm{S}}$ and $\mathrm{d}\ell_{\mathbf{\Pi}}$ on S and **Π**, respectively, are linked through $\mathrm{d}\ell_{\mathrm{S}} = r\mathrm{d}\ell_{\mathbf{\Pi}}$.

Proof. The infinitesimal length on S is given by

$$\mathrm{d}\ell_{\mathrm{S}} = \sqrt{E_{\mathrm{S}}\mathrm{d}u^2 + F_{\mathrm{S}}\mathrm{d}u\mathrm{d}\theta + G_{\mathrm{S}}\mathrm{d}\theta^2} \tag{5.2.3}$$

while on **Π** is given by

$$\mathrm{d}\ell_{\mathbf{\Pi}} = \sqrt{E_{\mathbf{\Pi}}\mathrm{d}u^2 + F_{\mathbf{\Pi}}\mathrm{d}u\mathrm{d}\theta + G_{\mathbf{\Pi}}\mathrm{d}\theta^2} \tag{5.2.4}$$

Combining Eqs. 5.2.3 , 5.2.4 and 5.2.2 leads to

$$\mathrm{d}\ell_{\mathrm{S}} = r\mathrm{d}\ell_{\mathbf{\Pi}} \tag{5.2.5}$$

□

5.3 Camber Surface Design

Having defined the meridional shape, the next step is the definition of the blade's camber surface. This is constructed by defining camber lines at several positions along the blade span. Each camber line is defined on the plane Π. The method described herein provides two options for defining the camber line, namely

- Circular arc camber line.
- Cubic Bézier curve camber line.

For both types of camber lines, the leading (LE) and trailing edge (TE) points need to be positioned. Their m coordinate is computed by intersecting the generatrix of the isospan surface, associated with this blade section, with the curves defining the meridional shape of the LE and TE curves, respectively. The θ coordinate of the LE and TE point is given as a function of the spanwise position η (from hub to shroud), namely $\theta_{\text{LE}} : \eta\left(u\right) \in \left[0, 1\right] \rightarrow \theta_{\text{LE}}\left(u\right)$ and $\theta_{\text{TE}} : \eta\left(u\right) \in \left[0, 1\right] \rightarrow \theta_{\text{TE}}\left(u\right)$. Both functions, $\theta_{\text{LE}}\left(u\right)$ and $\theta_{\text{TE}}\left(u\right)$ are defined as NURBS curves. Special care must be taken when specifying $\theta_{\text{LE}}\left(u\right)$ and $\theta_{\text{TE}}\left(u\right)$ as NURBS curves so that their spanwise distribution constitutes a function (i.e. a single θ for a given η). An example distribution of the circumferential position of the LE and TE is shown in Fig. 5.4.

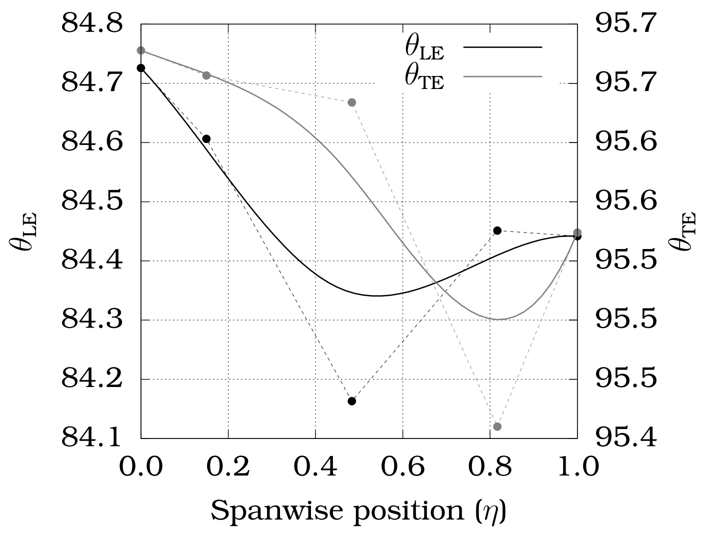


Figure 5.4: Spanwise distribution of the LE and TE circumferential positions. The distributions are defined as NURBS curves on a (η, θ) space and can be modified by moving their control points. Note that the LE curve corresponds to the left ordinate, while the TE to the right one.

The following subsections introduce the method the construction of the two types of camber lines is based upon. Once a number of camber lines at different spanwise sections is computed, the blade's camber surface can be computed by spanwise skinning. Skinning is the process of constructing a surface that passes from a number of section curves. The implementation of NURBS skinning in GMTurbo is based on the algorithm presented in [149].

5.3.1 Circular Arc Camber Lines

In this case the camber line is part of a circular arc. In order to compute the center, radius and sweep angle of the arc, apart from the leading and trailing edge positioning, one more parameter need to be specified. This is the metal angle β of either the leading edge (β_{LE}) or the trailing edge (β_{TE}). The β_{LE} or β_{TE}, are specified as spanwise distributions, similar to the θ angles. Once all the necessary parameters are defined, the radius ρ of the arc is computed by $\mathbf{r}_B - \mathbf{r}_A = \rho\left(\mathsf{n}_{\mathrm{LE}} - \mathsf{n}_{\mathrm{TE}}\right)$ and the center of the circle K by $\mathbf{r}_K = \mathbf{r}_A + \rho\mathsf{n}_{\mathrm{LE}}$. The symbol $\mathbf{r}$ is used to denote the position vector of a point on plane $\mathbf{\Pi}$. Useful definitions for the aforementioned construction are shown in Fig. 5.5. From this figure, the following relation between the stagger angle γ and the metal angles β_{LE} and β_{TE} must hold

$$2\gamma = \beta_{\mathrm{LE}} + \beta_{\mathrm{TE}} \tag{5.3.1.1}$$

If the position of the leading and trailing edge is fixed, then, from Eq. 5.3.1.1, it is evident that only one of β_{LE} and β_{TE} can be defined for creating a circular arc camber line.

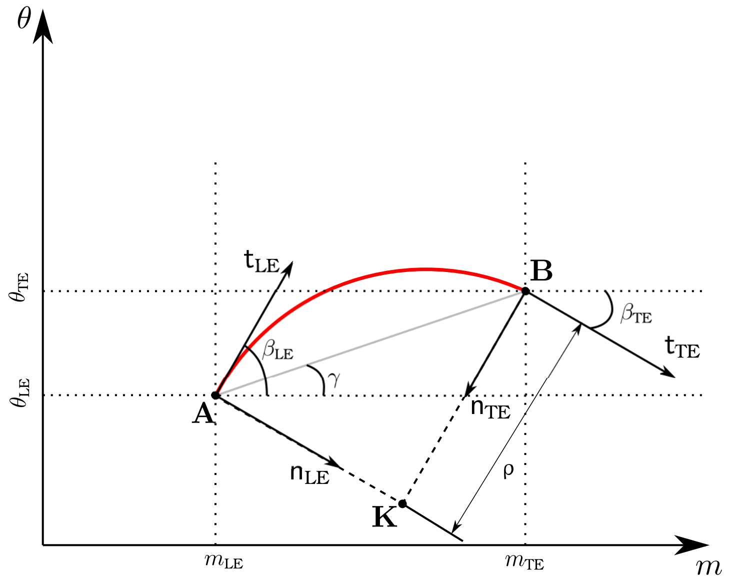


Figure 5.5: Construction of a circular arc camber line. Points **A** and **B** represent the leading and trailing edges respectively. The stagger angle γ is, then, fully defined. If one of the metal angles β_{LE}, β_{TE} is defined, the second is computed by Eq. 5.3.1.1. With known β_{LE}, β_{TE} the tangents to the camber line t_{LE}, t_{TE} and their corresponding normal vectors n_{LE}, n_{TE} are computed. The center of the circle **K** on which the camber line belongs is computed by intersecting the straight line from point **A** and direction n_{LE} and the straight line from point **B** and direction n_{TE}. Once point **K** is computed, the radius ρ and sweep angle of the circular arc can easily be computed as well.

5.3.2 Cubic Bézier Camber Lines

In this case, the camber line is a cubic polynomial Bézier curve. In order to define a cubic Bézier curve, four control points must be positioned. Let these control points be denoted as $\mathbf{P}_i$, $i = 0, \ldots, 3$. Clearly, $\mathbf{P}_0$ and $\mathbf{P}_3$ correspond to the leading and trailing edge, respectively. Since the metal angles β_{LE} and β_{TE} are two parameters that the designer wishes to control directly, the other two control points should lie on the tangents defined by these angles. More specifically, the positions of control points $\mathbf{P}_1$ and $\mathbf{P}_2$ must satisfy

$$\begin{aligned} \mathbf{P}_1 &= \mathbf{P}_0 + \lambda_1 \mathbf{t}_{\text{LE}} \\ \mathbf{P}_2 &= \mathbf{P}_1 + \lambda_2 \mathbf{t}_{\text{TE}} \end{aligned} \tag{5.3.2.1}$$

From Eq. 5.3.2.1 it is evident that two more free parameters arise for controlling the position of control points $\mathbf{P}_1$ and $\mathbf{P}_2$. Instead of controlling directly the two free parameters of Eq. 5.3.2.1, namely λ_1 and λ_2 we define two angles starting from the chord midpoint and intersection the lines of $\mathbf{t}_{\text{LE}}$ and $\mathbf{t}_{\text{TE}}$. These angles are denoted by δ_{LE} and δ_{TE}, respectively. The construction of a cubic Bézier camber line is shown in Fig. 5.6. Angles δ_{LE} and δ_{TE} are controlled through spanwise distributions in a similar manner as the β and θ angles.

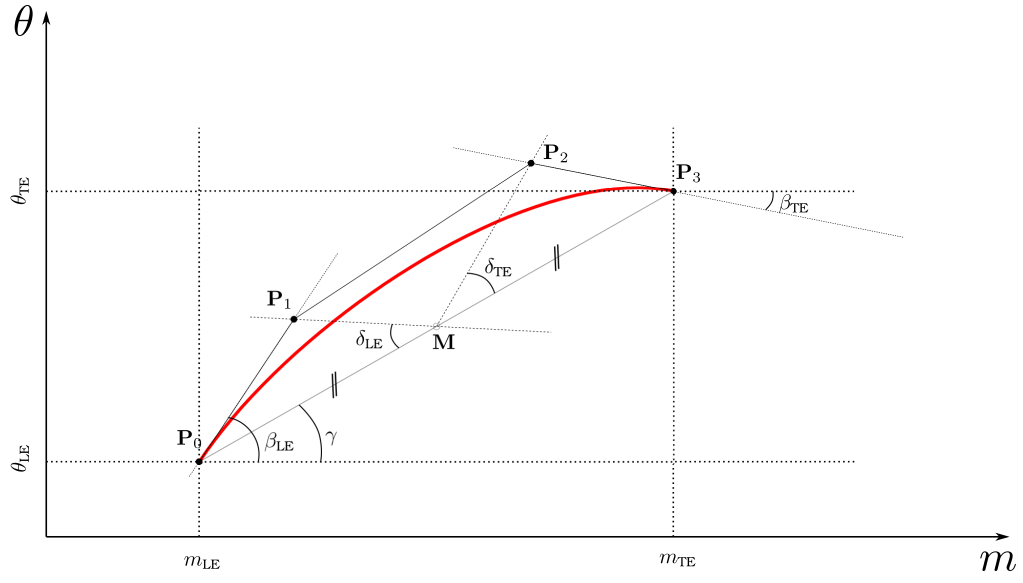


Figure 5.6: Construction of cubic Bézier camber line. The leading and trailing edge metal angles β_{LE} and β_{TE}, together with the leading and trailing edge positions (points $\mathbf{P}_0$ and $\mathbf{P}_3$) and angles δ_{LE} and δ_{TE}, define the control points of the cubic Bézier camber line.

There are some combinations of β_{LE}, γ and δ_{LE} that do not allow the definition of point $\mathbf{P}_1$. This happens if one of the following relations holds

$$\begin{aligned} \beta_{\text{LE}} - \gamma &= 0 \\ \beta_{\text{LE}} + \gamma &= 0 \\ \delta_{\text{LE}} &= 0 \\ \gamma + \delta_{\text{LE}} - \beta_{\text{LE}} &= 0 \end{aligned} \tag{5.3.2.2}$$

Similarly, if one of the following relations is true, then, the combination of β_{TE}, γ and δ_{TE} does now allow the definition of point $\mathbf{P}_2$.

$$\begin{aligned} \beta_{\text{TE}} - \gamma &= 0 \\ \beta_{\text{TE}} + \gamma &= 0 \\ \delta_{\text{TE}} &= 0 \\ \gamma + \delta_{\text{TE}} - \beta_{\text{TE}} &= 0 \end{aligned} \tag{5.3.2.3}$$

Special care is taken to remedy such occasions, specifically, if one of $\mathbf{P}_1$, $\mathbf{P}_2$ cannot be defined, then, instead of a cubic Bézier curve a quadratic one is created. In such a case, there is no control on the metal angle of the leading edge (if $\mathbf{P}_1$ is undefined) or the trailing edge (if $\mathbf{P}_2$ is undefined). If both $\mathbf{P}_1$ and $\mathbf{P}_2$ are undefined, then a linear camber line results.

Compared to the circular arc camber line, the cubic Bézier camber line provides more flexibility. More complex camber lines with change in curvature can be created. The price to pay for the additional flexibility is the increased number of parameters needed to specify the camber line shape. Additionally, it must also be noted that, a circular arc camber line cannot be created using the technique for cubic Bézier lines, since polynomial Bézier curves are used instead of rational ones and, a circular arc cannot be exactly represented by a polynomial curve.

5.4 Adding Blade Thickness

Once the camber line of each section is computed, thickness is assigned so as to create the airfoil for the section at hand. The half-thickness is assigned for each blade side, normal to the camber line. In order to allow flexibility to the designer/user and also maintain parameterization compactness, half-thicknesses are specified in two steps.

The first step is to define the normalized half-thickness profile. Each profile is defined as a NURBS curve with the normalized camber line arc length (s) as the abscissa and the normalized half-thickness $(\hat{\tau})$ as the ordinate. If τ is the actual half-thickness at a certain point and $\tau_{\max}$ the maximum half-thickness for the given blade section, then $\hat{\tau} = \frac{\tau}{\tau_{\max}}$. Several normalized profiles can be specified. When building a certain section, the corresponding normalized half-thickness profile is computed by interpolating the given profiles in the spanwise direction. An example of normalized half-thickness profiles can be seen in Fig. 5.7.

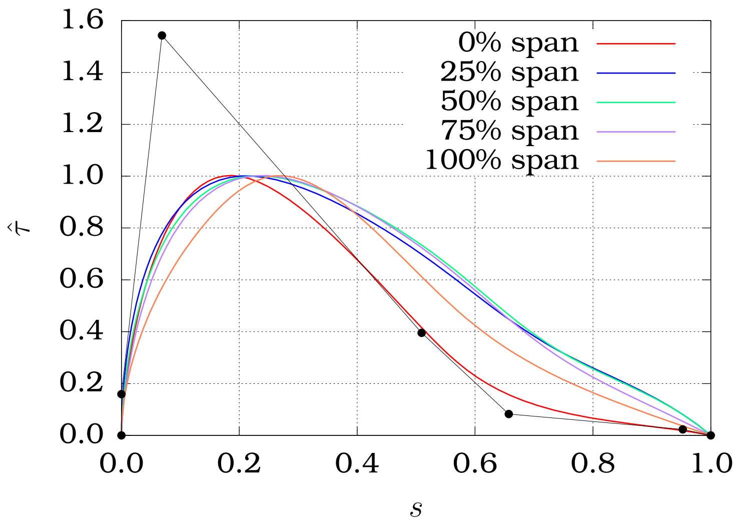


Figure 5.7: Normalized half-thickness profile curves for several spanwise positions are specified as NURBS curves. For clarity, only the control points that correspond to the 0% span profile are shown. The maximum ordinate value for all curves is 1. However, the control points do not necessarily lie in the range $[0, 1]$. If, by altering the control points, a $\hat{\tau}$ value larger than 1 results, then the whole profile is scaled accordingly so as to maintain normalization.

It must be noted that the normalized half-thickness at the leading and trailing edge need not be 0. The reason for this is that extra modification will be performed later on at these points so as to achieve the desired shape (blunt,sharp, arbitrary or circular). A zero normalized half-thickness at an edge (leading or trailing) is required only if no further geometrical modification is to be performed at this edge.

The second step for specifying half-thickness is to prescribe a spanwise distribution of $\tau_{\max}$. This is specified as a NURBS curve whose control points are defined on $(\eta, \tau_{\max})$. For a certain spanwise position and a certain position along the camber line, the actual half-thickness is computed as

$$\tau(s, \eta) = \hat{\tau}(s, \eta)\, \tau_{\max}(\eta) \tag{5.4.1}$$

Points on the blade side are computed by offseting the camber line points by the appropriate half-thickness value. However, since the offset is carried out on the transformed plane Π the value computed by Eq. 5.4.1 must be divided by the radius r which is the conformal factor (according to Eq. 5.2.5). For a given s, the value of m can be retrieved, then the parameter u on the generatrix (associated with this spanwise position) is computed and, finally, the radius r is computed by evaluating the corresponding point on the generatrix. The offset is performed normal to the camber line and its direction (left of right w.r.t. the camber line) depends on **a)** whether the suction or pressure side is computed, **b)** whether a rotor or stator blade is designed, **c)** whether the row rotates in the positive or negative circumferential direction (for rotor blades) and **d)** whether the blade belongs to a compressor/pump or a turbine. The process of computing half-thickness for a point on the blade given its spanwise position η and its corresponding position on the camber line s is shown in Fig. 5.8.

5.5 Leading and Trailing Edge Shapes

Having drawn the pressure and suction side (PS and SS, respectively), the leading and trailing edge shape should be defined. The options available are the following:

- Sharp edge: The thickness of the blade vanishes.
- Blunt edge: The thickness of the blade is non-zero and the two sides are linked through a linear interpolation between the PS and SS boundary curves.
- Wedge edge: The two sides are linked by two linear surfaces forming a wedge surface.
- Dovetail edge: The two sides are linked by two linear surfaces forming a dove tail.
- Circular arc edge: The two sides are linked with a circular arc with specified radius.
- Smooth edge: The two sides are linked by a smooth surface.

For a sharp edge the thickness profiles must be such that the half-thickness vanishes at the corresponding edge position. For smooth edges, the half-thickness vanishes as well but the NURBS curve of the profiles must be such that a tangent continuity is ensured around the edge. For circular arc edges, the thickness profiles need not vanish. The generated airfoil sides are extended and, then, a part of them is truncated in the process of modifying the edge shape. Finally, for blunt, wedge and dovetail edges the half-thickness must be non-zero. In case of wedge edges part of the profile is truncated, while in case of dovetail ones the blade sides are extended. An example of each kind of edge is presented in Fig. 5.9.

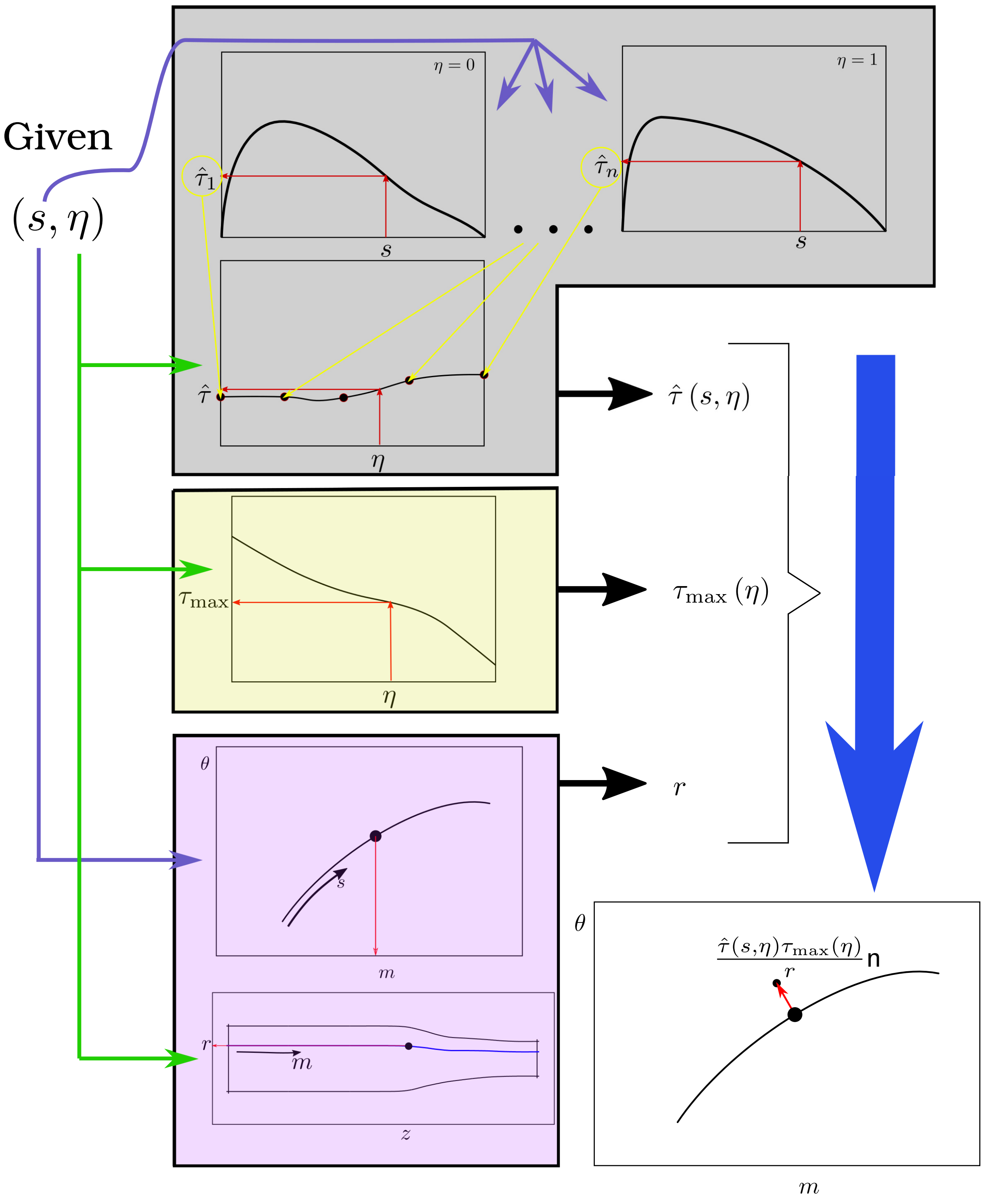


Figure 5.8: Process of adding half-thickness and computing a point on the blade sides in GMTurbo.

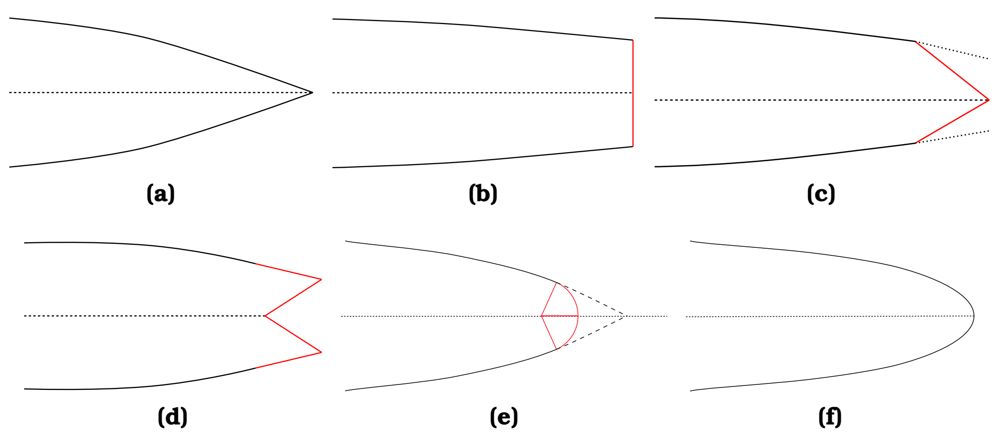


Figure 5.9: The different types of edges are presented. These shapes can be produced either for the leading or the trailing edge of a blade. (a) A sharp edge is created by vanishing half-thickness profiles, without any slope continuity. (b) A blunt edge is created by specifying a non-zero half-thickness. (c) A wedge type edge is created by specifying non-zero half-thickness and controlling the wedge half-angle (for 90° it reduces to a blunt edge). (d) A dovetail type edge is created as a wedge type edge, but with a half-angle greater than 90°. In addition, the blade sides need to be extended. (e) A circular arc edge is created by extending the blade sides and fitting a circle of specified radius between them. (f) A general smooth edge is created by vanishing half-thickness profiles with slope continuity.

The cases of blunt, wedge and dovetail edges can be unified in one. The user needs to control only the half-angle generated by the trailing edge and the mean camber line. These angles are given as spanwise distributions. Depending on the value of these angles, one of the three types of edges emerges or even a hybrid type is possible. For circular arc edges, the radius of the arc is also specified as a spanwise distribution.

5.6 Creating the 3D blade surfaces

After having defined several sections of the blade at different spanwise positions, a spanwise skinning NURBS algorithm is used to create the pressure and suction side surfaces, as well as the leading and trailing edge ones. Before perfoming the skinning, the sections are mapped in 3D Cartesian coordinates using an approximate inverse mapping (by algorithmically inverting Eq. 5.2.1). The process of creating the 3D blade is shown in Fig. 5.10.

Since the curves used by the skinning algorithm may be defined on greatly differing knot vectors, the CAD description of the blade may be rather complex. In order to reduce the number of control points present in the final CAD description and enhance the CAD representation compactness, the blade sides can be re-

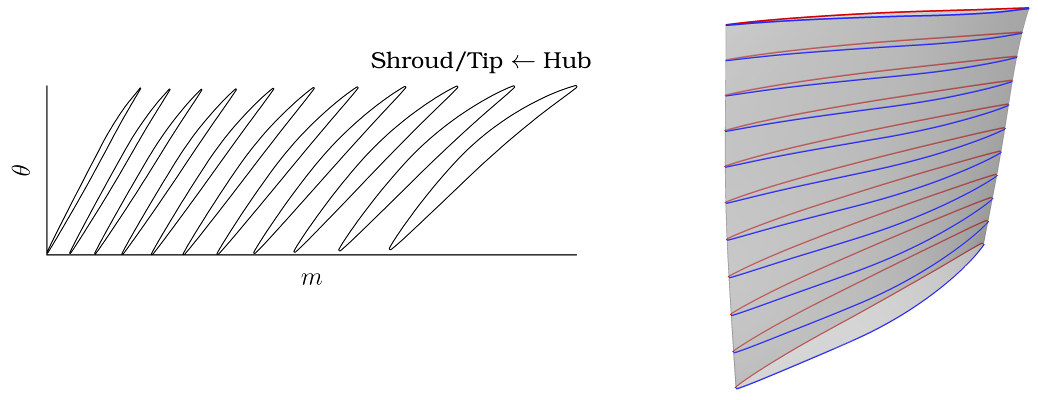


Figure 5.10: Creating the 3D blade from the computed blade sections. (a) The section curves in 3D are created from inverse mapping of the sections created on the several planes Π for each isospan position. (b) Generating the 3D blade surfaces throught spanwise skinning of the section curves.

interpolated using common knot vectors that arise from averaging the section curves knots. Then, the resulting points can be interpolated using a surface interpolation algorithm (instead of skinning).

The hub, shroud and tip surfaces, as well as the upstream and downstream boundary surfaces are generated by revolution of the corresponding generatrices or meridional curves. When no tip gap exists, the hub and shroud surfaces need to be trimmed by the blade. Since most commercial CAD packages and mesh generation software are sensitive to the tolerance used for creating trimmed surfaces, the blade surfaces are extended in the spanwise direction, prior to CAD export, so as to ensure proper trimming from the third party package. When a tip gap exists, the same trimming, between the blade and the surface generated by revolving the tip generatrix, is performed to compute the actual tip surface.

The final collection of surfaces representing the full multi-component configuration is exported in neutral CAD formats such as IGES or STEP. If a faceted geometry representation is sought, a surface triangulation algorithm based on a hybrid Delaunay-Advancing Front process is employed and the resulting surfaces are exported in STL format files.

5.7 Summary of Design Variables

In the following table a list of parameters that arise from the design/parameterization process are summarized. It must be noted that each parameter refers to a set of control points (e.g. the parameter "Hub Generatrix" refers to all the control points that define the shape of the hub generatrix). Some additional variables may be available, depending on the case (e.g. the type of camber line, the type of leading or trailing edge). All of the variables may or may not be altered during the optimization process. However, they need to be defined (even with a constant value) so that the set of design parameters is complete and a geometry can be generated.

In the following tables, the following convention is used for naming component families

- SBR: Shrouded blade row.
- HTBR: Blade row with hub tip.
- STBR: Blade row with shroud tip.
- OBR: Open blade row.
- DH: Duct with hub surface present.
- DNH: Duct with no hub surface present.
- HO: Hub only components (inlet cones etc.).

In addition, must of the distributions are defined as NURBS curves, so their control points $(\mathbf{P})$ are used as parameters.

Parameter	Symbol	Type	Mandatory	Constant
Hub generatrix	$\mathbf{P}_{\text{Hub}}$	(z, r) vector	SBR/HTBR/STBR/ OBR/DH/HO	No
Shroud generatrix	$\mathbf{P}_{\text{Shroud}}$	(z, r) vector	SBR/HTBR/ STBR/DH/DNH	No
Hub tip generatrix	$\mathbf{P}_{\text{HT}}$	(z, r) vector	HTBR	No
Shroud tip generatrix	$\mathbf{P}_{\text{ST}}$	(z, r) vector	STBR/OBR	No
Upstream boundary meridional shape	$\mathbf{P}_{\text{Up}}$	(z, r) vector	All	No
Downstream boundary meridional shape	$\mathbf{P}_{\text{Down}}$	(z, r) vector	All	No
Leading edge meridional shape	$\mathbf{P}_{\text{LE}}$	(z, r) vector	SBR/HTBR/STBR	No
Trailing edge meridional shape	$\mathbf{P}_{\text{TE}}$	(z, r) vector	SBR/HTBR/STBR	No
Leading edge θ	$\mathbf{P}_{\theta_{\text{LE}}}$	$(\eta, \theta_{\text{LE}})$ vector	SBR/HTBR/ STBR/OBR	No
Trailing edge θ	$\mathbf{P}_{\theta_{\text{TE}}}$	$(\eta, \theta_{\text{TE}})$ vector	SBR/HTBR/ STBR/OBR	No

Leading edge β	$\mathbf{P}_{\beta_{\text{LE}}}$	$(\eta, \beta_{\text{LE}})$ vector	SBR/HTBR/ STBR/OBR	No
Trailing edge β	$\mathbf{P}_{\beta_{\text{TE}}}$	$(\eta, \beta_{\text{TE}})$ vector	SBR/HTBRR STBR/OBR	No
Spanwise position of nth half-thickness PS profile	$\eta_{\text{PS\_profile-}n}$	scalar	SBR/HTBR/ STBR/OBR	Yes
nth half-thickness PS profile shape	$\mathbf{P}_{\hat{\tau}_{\text{PS\_profile-}n}}$	$(s, \hat{\tau})$	SBR/HTBR/ STBR/OBR	No
Spanwise position of nth half-thickness SS profile	$\eta_{\text{SS\_profile-}n}$	scalar	SBR/HTBR/ STBR/OBR	Yes
nth half-thickness SS profile shape	$\mathbf{P}_{\hat{\tau}_{\text{SS\_profile-}n}}$	$(s, \hat{\tau})$	SBR/HTBR/ STBR/OBR	No
PS spanwise half-thickness factor	$\mathbf{P}_{\text{PS\_}\tau_{\max}}$	$(\eta, \tau_{\max})$	SBR/HTBR/ STBR/OBR	No
SS spanwise half-thickness factor	$\mathbf{P}_{\text{SS\_}\tau_{\max}}$	$(\eta, \tau_{\max})$	SBR/HTBR/ STBR/OBR	No

Table 5.1: List of parameters used to define a turbomachinery component geometry. The component type for which a parameter is mandatory is also shown. If more rows or multiple components are parameterized, these parameters must be specified for each one of them separately. Some parameters, marked as constant, need to be defined and remain constant during an optimization process, since their modification is not sensible. The type of each parameter (scalar or vector) is also shown.

If the camber line is of cubic Bezier type two additional parameters are available, namely

Parameter	**Symbol**	**Type**	**Mandatory**	**Constant**
Leading edge δ	$\mathbf{P}_{\delta_{\text{LE}}}$	$(\eta, \delta_{\text{LE}})$	SBR/HTBR/ STBR/OBR	No
Trailing edge δ	$\mathbf{P}_{\delta_{\text{TE}}}$	$(\eta, \delta_{\text{TE}})$	SBR/HTBR/ STBR/OBR	No

Table 5.2: Additional parameters available when a cubic Bezier camber type is used.

In addition, when a blunt/wedge/dovetail type edge is used two more parameters become available, namely the distributions of angles ϕ between the camber line and the edge surface towards the pressure (PS) and suction side (SS) of the blade. These are given for each edge separately.

Finally, when a circular arc edge is chosen, instead of the parameters of Table 5.3 the spanwise distribution of the arc radius $\rho(\eta)$ becomes available, again separately for each such edge.

Parameter	**Symbol**	**Type**	**Mandatory**	**Constant**
PS ϕ spanwise distribution	$\mathbf{P}_{\phi_{\text{PS}}}$	(η, ϕ_{SS})	SBR/HTBR/ STBR/OBR	No
SS ϕ spanwise distribution	$\mathbf{P}_{\phi_{\text{SS}}}$	(η, ϕ_{PS})	SBR/HTBR/ STBR/OBR	No

Table 5.3: Additional parameters available when a blunt/wedge/dovetail edge shape is used.

Parameter	**Symbol**	**Type**	**Mandatory**	**Constant**
ρ spanwise distribution	$\mathbf{P}_{\rho}$	(η, ρ)	SBR/HTBR/ STBR/OBR	No

Table 5.4: Additional parameters available when a circular arc edge shape is employed.

Chapter 6

Free-Form Deformation based on Volumetric NURBS

In Chapter 4 the basics of shape parameterization techniques are presented. There, the various parameterization methods are categorized in two classes, namely direct boundary and free-form ones. In the last section of this chapter, some indicative examples of direct boundary parameterizations for simple external aerodynamic shapes are presented.

As it has already been discussed in Chapter 4, direct boundary parameterization methods have a major drawback when coupled with CFD analysis. This originates from the fact that these methods provide a mechanism for controlling only the boundary surface mesh shape and, consequently, an additional tool is needed for propagating the deformation in the interior of the CFD domain.

This chapter is concerned with a method falling in the class of free-form deformation methods. The method is based on trivariate NURBS volumes which will be referred to as volumetric NURBS. As a free-form deformation technique, volumetric NURBS provide the means for controlling the shape of an entire region in space. The method is used for parameterizing the shape under consideration, which is embedded in the trivariate NURBS volume and, also, the part of the CFD mesh that is embedded in the same volume. In previous works using volumetric B-Splines and NURBS, these are used only for controlling the shape under consideration [121]. In this thesis, their use is extended in deforming the volume mesh as well.

Using a parameterization method that directly controls also the volume mesh has an additional advantage when it comes to adjoint-based shape optimization. If the parameterization method is differentiated, then the FI adjoint approach can be used with almost negligible additional cost compared to the SI one, since the costly finite difference approach is avoided.

In this chapter, the volumetric parameterization method is introduced and explained. Then, its differentiation that enables the low-cost exact computation of $\frac{\delta x_k^j}{\delta b_i}$ (with $i = 1, \ldots, N$, $k = 1, \ldots, 3$, $j = 1, \ldots, N_V$, N the number of design variables and N_V the number of mesh nodes) is described.

As shown below, despite their advantages, especially in external aerodynamic shape optimization problems, the use of volumetric NURBS has major shortcomings when applied in shape optimization of turbomachinery blade rows. To improve the applicability of the volumetric NURBS technique in such cases, the turbomachinery volumetric NURBS method (TMVNURBS) is introduced later in this chapter. Finally, the differentiation of TMVNURBS is presented. The parameterization tools developed in the chapter are employed in the next chapters, in several shape optimization applications, both in the field of external and internal aerodynamics, including turbomachinery.

6.1 NURBS volumes

The basis for the volumetric parameterization method to be presented is the NURBS volume. A NURBS volume is a trivariate vector valued function $\mathbf{V}(u, v, w)$ mapping a point from the 3D parametric space (u, v, w) to 3D Cartesian space (x, y, z), defined as

$$\begin{aligned} \mathbf{V}(u,v,w) &= R_{ijk,p_u p_v p_w} \mathbf{P}_{ijk} \\ \text{where } R_{ijk,p_u p_v p_w} &= \frac{N_{i,p_u}(u)\, N_{j,p_v}(v)\, N_{k,p_w}(w)\, \mathrm{w}_{ijk}}{N_{\ell,p_u}(u)\, N_{q,p_v}(v)\, N_{s,p_w}(w)\, \mathrm{w}_{\ell qs}} \\ i = 0, \ldots, n_u - 1,\ j &= 0, \ldots, n_v - 1 \text{ and } k = 0, \ldots, n_w - 1 \end{aligned} \tag{6.1.1}$$

The B-Spline basis functions N_{i,p_u}, N_{j,p_v} and N_{k,p_w}, of degrees p_u, p_v and p_w respectively, are defined on the knot vectors $U = [u_0, \ldots, u_{m_u-1}]$, $V = [v_o, \ldots, v_{m_v-1}]$ and $W = [w_o, \ldots, w_{m_w-1}]$. The control points $\mathbf{P}_{ijk}$ and their corresponding weights w_{ijk} are arranged is a 3D control grid (hence the three indices in the notation). As with curves and surfaces, a NURBS volume can be expressed as a function mapping the parametric space (u, v, w) to the 4D homogeneous space $(\mathrm{w}x, \mathrm{w}y, \mathrm{w}z, \mathrm{w})$. In such a case a NURBS volume can be expressed as

$$\begin{aligned} \mathbf{V}(u,v,w) &= N_{i,p_u}(u)\, N_{j,p_v}(v)\, N_{k,p_w}(w)\, \mathbf{P}^{\mathrm{w}}{}_{ijk} \\ \text{with } \mathbf{P}^{\mathrm{w}}{}_{ijk} &= \left[(\mathrm{w}x)_{ijk}, (\mathrm{w}y)_{ijk}, (\mathrm{w}z)_{ijk}, \mathrm{w}_{ijk} \right] \end{aligned} \tag{6.1.2}$$

Clearly, keeping one of the three parameters constant leads to an iso-parametric surface. The basic arrangement of a NURBS volume is shown in Fig. 6.1.

In order to control a shape embedded inside a NURBS volume, its parametric

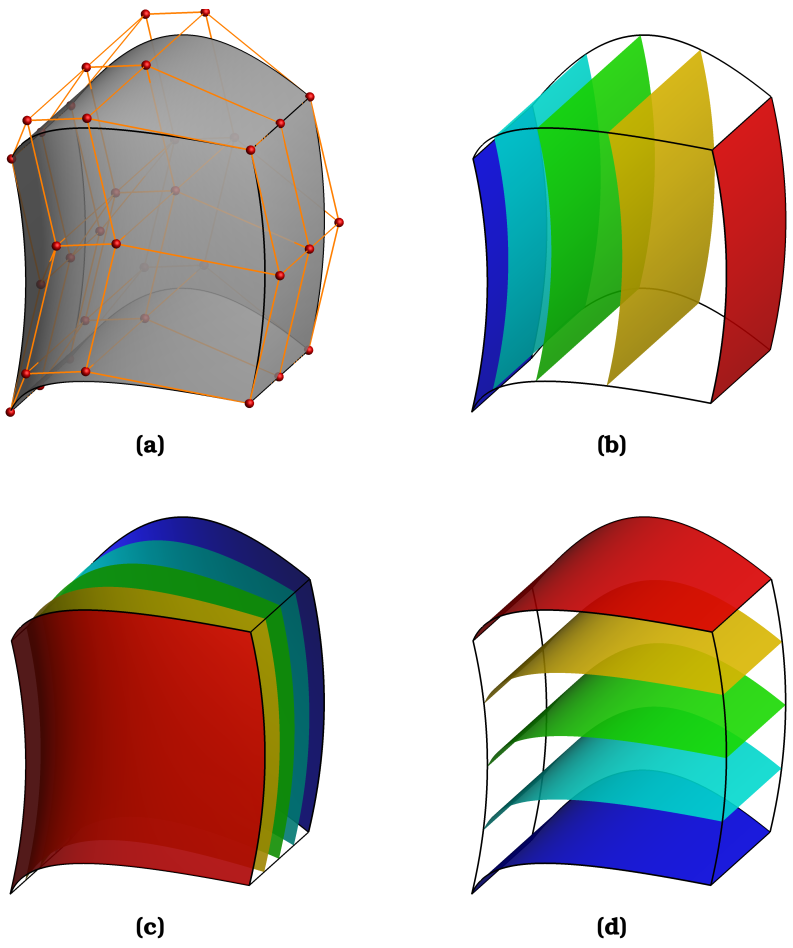


Figure 6.1: (a) Definition of $4 \times 3 \times 3$ NURBS volume. The control points are shown in red. The iso-u, iso-v and iso-w surfaces are shown in (b), (c) and (d) respectively.

coordinates must be known. Practically, for a space region discretized by a mesh, it is enough to compute the parametric coordinates (u, v, w) of its nodes given their Cartesian coordinates. This is done by inverting Eq. 6.1.2 . The inversion leads to a non-linear system of three equations with three unknowns for each mesh node. The system reads

$$\mathbf{V}\left(u, v, w\right) - \mathbf{r} = 0 \tag{6.1.3}$$

where $\mathbf{r} = (x, y, z)$ is the position of the mesh node. The system is solved iteratively by the Newton-Raphson method

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix}^{\text{new}} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}^{\text{old}} + \begin{bmatrix} \frac{\partial x(u,v,w)}{\partial u}\Big|^{\text{old}} & \frac{\partial x(u,v,w)}{\partial v}\Big|^{\text{old}} & \frac{\partial x(u,v,w)}{\partial w}\Big|^{\text{old}} \\ \frac{\partial y(u,v,w)}{\partial u}\Big|^{\text{old}} & \frac{\partial y(u,v,w)}{\partial v}\Big|^{\text{old}} & \frac{\partial y(u,v,w)}{\partial w}\Big|^{\text{old}} \\ \frac{\partial z(u,v,w)}{\partial u}\Big|^{\text{old}} & \frac{\partial z(u,v,w)}{\partial v}\Big|^{\text{old}} & \frac{\partial z(u,v,w)}{\partial w}\Big|^{\text{old}} \end{bmatrix}^{-1} \begin{bmatrix} x - x\left(u^{\text{old}}, v^{\text{old}}, w^{\text{old}}\right) \\ y - y\left(u^{\text{old}}, v^{\text{old}}, w^{\text{old}}\right) \\ z - z\left(u^{\text{old}}, v^{\text{old}}, w^{\text{old}}\right) \end{bmatrix} \tag{6.1.4}$$

To achieve a fast and robust convergence of Eq. 6.1.4, a good approximate solution is needed. The complexity of the NURBS volume shape is directly related to the number of control points. Based on this fact, a number of points are seeded uniformly in the three parametric directions. The number of seed points is directly related to the number of control points in each of these directions. Then, the initial values of the parametric coordinates for each node are chosen to be the parametric coordinates of the nearest seed point. Typically, with a good initialization, the iterative process described by Eq. 6.1.4 converges with adequate accuracy in no more than five to ten iterations. For points lying outside the NURBS volume the iterative process fails to converge. Thus, the failure or success of convergence may serve as a criterion for identifying which nodes lie outside or inside the NURBS volume.

In order to speed-up the point inversion process, Eq. 6.1.4 is solved on the GPU in parallel. An additional gain in computational time is achieved by marking all the nodes that lie outside the bounding box of the control points as outliers. Such a decision is reasonable and based upon the strong convex hull property that characterizes NURBS volumes. The process of point inversion with some details about its GPU implementation is presented in Fig. 6.2.

Once the process of point inversion is completed, any change in a control point position or weight affects the position of the CFD nodes inside the NURBS volume. Consequently, both surface nodes as well as volume mesh nodes inside the NURBS volume can be controlled through this modification. However, special care must be taken to ensure mesh continuity. Specifically, if mesh nodes are positioned on both sides of a NURBS volume boundary, any displacement of this boundary will destroy mesh continuity. The reason is that nodes inside the NURBS volume will move due to control point displacement, but nodes outside the NURBS volume will remain still. This problem is overcome by ensuring that a layer of control points remains intact, thus maintaining $\mathcal{C}^0$ continuity throughout the mesh. Higher degrees of continuity are achieved by freezing additional layers of control points of the NURBS volume.

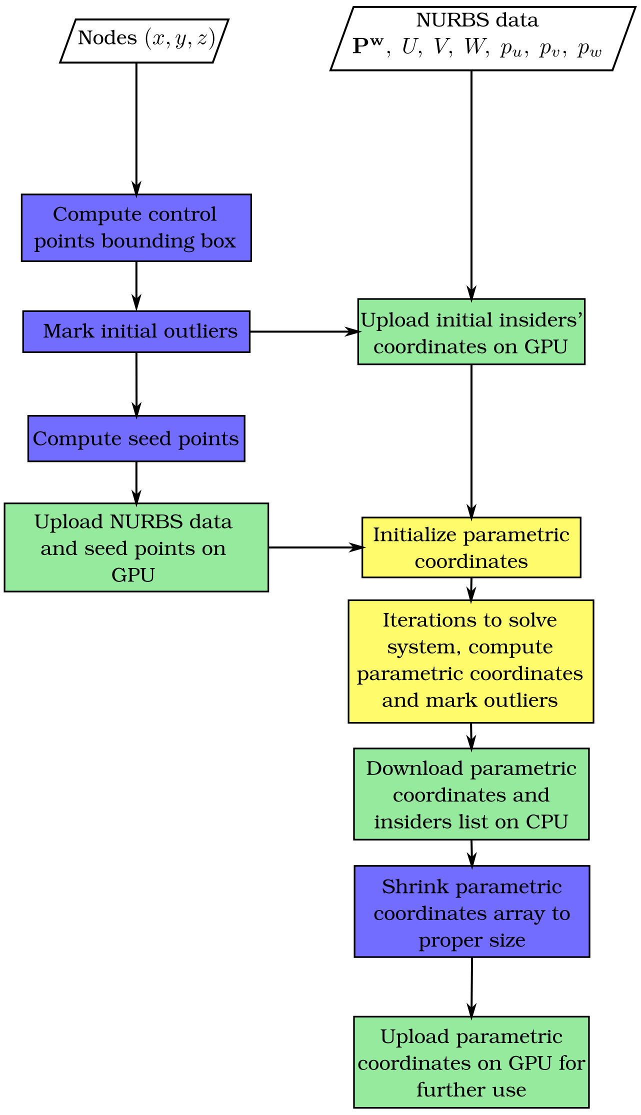

Nodes (x,y,z)
NURBS data
$\mathbf{P^w}$, U, V, W, p_u, p_v, p_w
Compute control points bounding box
Mark initial outliers
Compute seed points
Upload NURBS data and seed points on GPU
Upload initial insiders' coordinates on GPU
Initialize parametric coordinates
Iterations to solve system, compute parametric coordinates and mark outliers
Download parametric coordinates and insiders list on CPU
Shrink parametric coordinates array to proper size
Upload parametric coordinates on GPU for further use

Figure 6.2: Flowchart of the point inversion process for a NURBS volume. The blue processes are executed on the CPU and the yellow ones on the GPU. Practically, all the computationally expensive tasks run on the GPU. Intermediate data transfers between CPU and GPU are shown in green. Partial overlapping of data transfers and computations is possible within the NVIDIA CUDA environment.

6.2 Computation of Geometric Sensitivities

This section is concerned with the computation of the so called geometric sensitivities. These are denoted as $\frac{\delta x_k^j}{\delta b_i}$, where $k = 1, \ldots, 3$, $j = 1, \ldots, N_D$ and $i = 0, \ldots, N$.

N_D is the number of nodes associated with the sensitivity derivative integral (see Chapter 3) to be computed, thus it can be $N_D = N_S$ for surface integrals or $N_D = N_V$ for volume integrals. Moreover, N is the total number of design variables. More specifically, if all three Cartesian coordinates and the weight of a control point are allowed to vary, then four design variables b_i are associated with this control point.

Let x_r^ℓ, $(r = 1, \dots, 3,\ \ell = 1, \dots, N_D)$ denote the Cartesian coordinates of a point in space and $X_{s,ijk}$, $(s = 1, \dots, 3,\ i = 0, \dots, n_u,\ j = 0, \dots, n_v,\ k = 0, \dots, n_w)$ the Cartesian coordinates of a NURBS volume control point. Then, by differentiating Eq. 6.1.1 the geometric sensitivity is computed as

$$\frac{\delta x_r^\ell}{\delta X_{s,ijk}} = R_{ijk,p_u p_v p_w}\left(u^\ell, v^\ell, w^\ell\right)\delta_{rs} \tag{6.2.1}$$

For the derivative of a node's Cartesian coordinates w.r.t. a control point weight, the differentiation of Eq. 6.1.1 leads to

$$\frac{\delta x_r^\ell}{\delta \mathsf{w}_{ijk}} = X_{r,mnq}\frac{\delta R_{mnq}\left(u^\ell, v^\ell, w^\ell\right)}{\delta \mathsf{w}_{ijk}} \tag{6.2.2}$$

where the degrees of the basis functions are omitted for brevity. Let the symbol R_{mnq}^ℓ denote the trivariate basis function of degrees $p_u,\ p_v,\ p_w$ computed with the parametric coordinates $\left(u^\ell, v^\ell, w^\ell\right)$ of point ℓ. Let, also, the symbol $N_m^{u,\ell}$ denote the univariate B-Spline basis function of degree p_u in the u parametric direction, computed at $u = u^\ell$. Taking into account the second of Eqs. 6.1.1, $\frac{\delta R_{mnq}^\ell}{\delta \mathsf{w}_{ijk}}$ is computed as

$$\frac{\delta R_{mnq}^\ell}{\delta \mathsf{w}_{ijk}} = \frac{N_i^{u,\ell}N_j^{v,\ell}N_k^{w,\ell}}{\left(N_a^{u,\ell}N_b^{v,\ell}N_c^{w,\ell}\mathsf{w}_{abc}\right)^2}\left(N_d^{u,\ell}N_h^{v,\ell}N_s^{w,\ell}\mathsf{w}_{dhs} - N_m^{u,\ell}N_n^{v,\ell}N_q^{w,\ell}\mathsf{w}_{mnq}\right) \tag{6.2.3}$$

where no summation is implied on indices $m,\ n,\ q$.

Combining Eqs. 6.2.2 and 6.2.3 leads to

$$\frac{\delta x_r^\ell}{\delta \mathsf{w}_{ijk}} = X_{r,mnq}N_i^{u,\ell}N_j^{v,\ell}N_k^{w,\ell}\frac{\left(N_d^{u,\ell}N_h^{v,\ell}N_s^{w,\ell}\mathsf{w}_{dhs} - N_e^{u,\ell}N_f^{v,\ell}N_g^{w,\ell}\mathsf{w}_{mnq}\delta_{me}\delta_{nf}\delta_{qg}\right)}{\left(N_a^{u,\ell}N_b^{v,\ell}N_c^{w,\ell}\mathsf{w}_{abc}\right)^2} \tag{6.2.4}$$

From Eq. 6.2.1, it can be seen that, once the parametric coordinates $\left(u^\ell, v^\ell, w^\ell\right)$ of a mesh node are computed, and, if the weights of the control points are not design variables, then, the geometric sensitivities can be computed and stored, since they remain fixed during the optimization (no dependence on the control point position). However, since the geometric sensitivities are computed for all mesh nodes inside the NURBS volume, the memory consumption for large meshes can be prohibitively large. Specifically, storing only the non-zero terms leads to the storage of $N_{CP} \times N_I$ double precision values, where N_{CP} is the number of free control points and N_I the number of volume mesh nodes inside the NURBS volume.

Taking into account that for the FI adjoint approach, spatial gradients of the geometric sensitivities are also needed and that a great deal of memory is already consumed for storing the solution of the flow and adjoint field, the storage of all a priori computed geometric sensitivities becomes inapplicable.

In addition, geometric sensitivities w.r.t. control point weights need to be computed again in each optimization cycle, since their value depends also on the current position and weight of the control points (Eq. 6.2.4). Thus, the geometric sensitivities are (re)computed on the fly for each control point. To minimize the computation cost of computing $\frac{\delta x_r^\ell}{\delta b_i}$, the computation is carried out in parallel for all mesh nodes on the GPU.

6.3 NURBS Volumes for Turbomachinery

Clearly, the use of a volumetric NURBS free-form deformation technique has many advantages compared to direct boundary ones, especially in terms of complexity and differentiability. However, it is not directly applicable for the parameterization of turbomachinery components. There are two reasons for this.

The first is the presence of surfaces of revolution (hub and shroud surfaces), which should remain so throughout the optimization. This is difficult to ensure with traditional NURBS volumes, since a very complicated and combined displacement of control points is typically needed to maintain axisymmetry in each optimization cycle. In addition, the more complicated the shape of the hub and shroud surfaces the more control points and interaction among them is needed to achieve this effect.

The second reason is the presence of periodic boundaries and geometry/mesh periodicity in general. If the CFD volume mesh is also modified by the NURBS volume, this must be done is such a way that each blade passage be discretized in exactly the same way to avoid introducing numerical non-axisymmetry. This means that mesh nodes whose position differs only by a multiple of the blade row pitch in the circumferential direction, must be displaced in the same way and an analogous periodic displacement is implied for the control points.

The first obstacle can be overcome by defining NURBS volumes that do not include the hub and shroud surfaces at all. However, such an approach is of little engineering value since these areas are very important for 3D flow features arising in flows inside turbomachinery blade rows. Similarly, the second obstacle can be surpassed by defining NURBS volumes that exclude the periodic boundaries. This, in turn, is very restrictive in terms of the shapes that can be generated and the mesh quality after each modification, especially in turbine rows or rows with large blade count, where the periodic boundaries usually lie relatively close to the blade.

The remedy to the axisymmetry problem is sought through an intermediate transformation inspired by the coordinate systems used for the design of turbomachinery bladings. The coordinate system introduced will be referred to as the passage coordinate system (PCS) and the transformation from the 3D Cartesian coordinate system to this one as the 3D Cartesian to passage coordinate transformation. Then, an implicit treatment is introduced as a remedy to the periodicity treatment problem. Both of these solutions are developed in the following subsections.

6.3.1 Cartesian to PCS Transformation

Let the axisymmetric hub and shroud surfaces be represented by a generatrix revolved around the z-axis. The choice of the z-axis is not restrictive at all. More specifically, let $\mathbf{C}^H(\xi)$ be the hub generatrix and $\mathbf{C}^S(\xi)$ the shroud one and let ξ be defined as a normalized parametric coordinate. This curves can be viewed as univariate vector functions mapping a value of the parameter ξ on the meridional plane (z, r), namely

$$\begin{aligned} \mathbf{C}^H(\xi) &: \xi \in [0,1] \rightarrow (z,r) \text{ on hub} \\ \mathbf{C}^S(\xi) &: \xi \in [0,1] \rightarrow (z,r) \text{ on shroud} \end{aligned} \tag{6.3.1.1}$$

Using a linear combination of $\mathbf{C}^H$ and $\mathbf{C}^S$ allows the description of any point on the meridional plane by introducing a single additional parameter, namely the parameter of the linear interpolation η. Thus, any point $\mathbf{r} = (z, r)$ can be expressed as

$$\mathbf{r}(\xi, \eta) = (1-\eta)\,\mathbf{C}^H(\xi) + \eta\,\mathbf{C}^S(\xi) \tag{6.3.1.2}$$

By doing so, a coordinate transformation is defined mapping the (z, r) coordinates of any point between hub and shroud to the newly defined (ξ, η) ones. By introducing the circumferential position of the point θ, a new 3D coordinate system is defined, namely the (ξ, θ, η) system, which is the Passage Coordinate System (PCS). The PCS is not an orthonormal coordinate system. However, it forms a curvilinear system that can be shown to preserve the orientation. Moreover, if the passage does not degenerate (hub and shroud generatrices do not intersect or touch each other and defined both with the same orientation, i.e. upstream to downstream or vice versa) it can be shown that the mapping $(r, \theta, z) \mapsto (\xi, \theta, \eta)$ is one-to-one.

The transformation relationships between the Cartesian, cylindrical systems and PCS are given below.

$$\begin{aligned}
&x = r\cos\theta &&, \; y = r\sin\theta &&, \; z = z \\
&r = \sqrt{x^2+y^2} &&, \; \theta = \tan^{-1}(y/x) &&, \; z = z \\
&r = (1-\eta)\, r^H(\xi) + \eta\, r^S(\xi) &&, \; \theta = \theta &&, \; z = (1-\eta)\, z^H(\xi) + \eta\, z^S(\xi)
\end{aligned} \tag{6.3.1.3}$$

Therefore, if the passage coordinates are known for a certain point, computing its Cartesian or cylindrical coordinates is straight forward. However, if the Cartesian or cylindrical coordinates are known, there is no closed form relation that allows the computation of the passage coordinates.

To circumvent this problem, a Newton-Raphson iteration method is used to solve the following system of non-linear equations for each mesh node ℓ,

$$\begin{bmatrix} r^\ell - r\left(\xi^\ell, \eta^\ell\right) \\ z^\ell - z\left(\xi^\ell, \eta^\ell\right) \end{bmatrix} = 0 \tag{6.3.1.4}$$

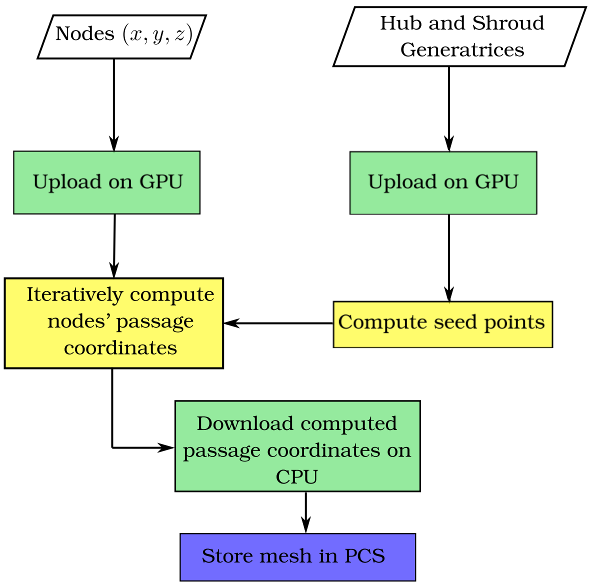

Nodes (x, y, z)
Hub and Shroud Generatrices
Upload on GPU
Upload on GPU
Iteratively compute nodes' passage coordinates
Compute seed points
Download computed passage coordinates on CPU
Store mesh in PCS

Figure 6.3: Flowchart of the process of transforming the computational mesh from Cartesian to passage coordinates. Blue processes are executed on the CPU, while the yellow ones are carried out on the GPU. Intermediate data transfers between CPU and GPU are shown in green.

Again, a good initialization is needed for a robust and fast convergence. Seed points are used again to assign the initial values to the passage coordinates for each mesh node. The process of computing the passage coordinates for each mesh node is described in Fig. 6.3.

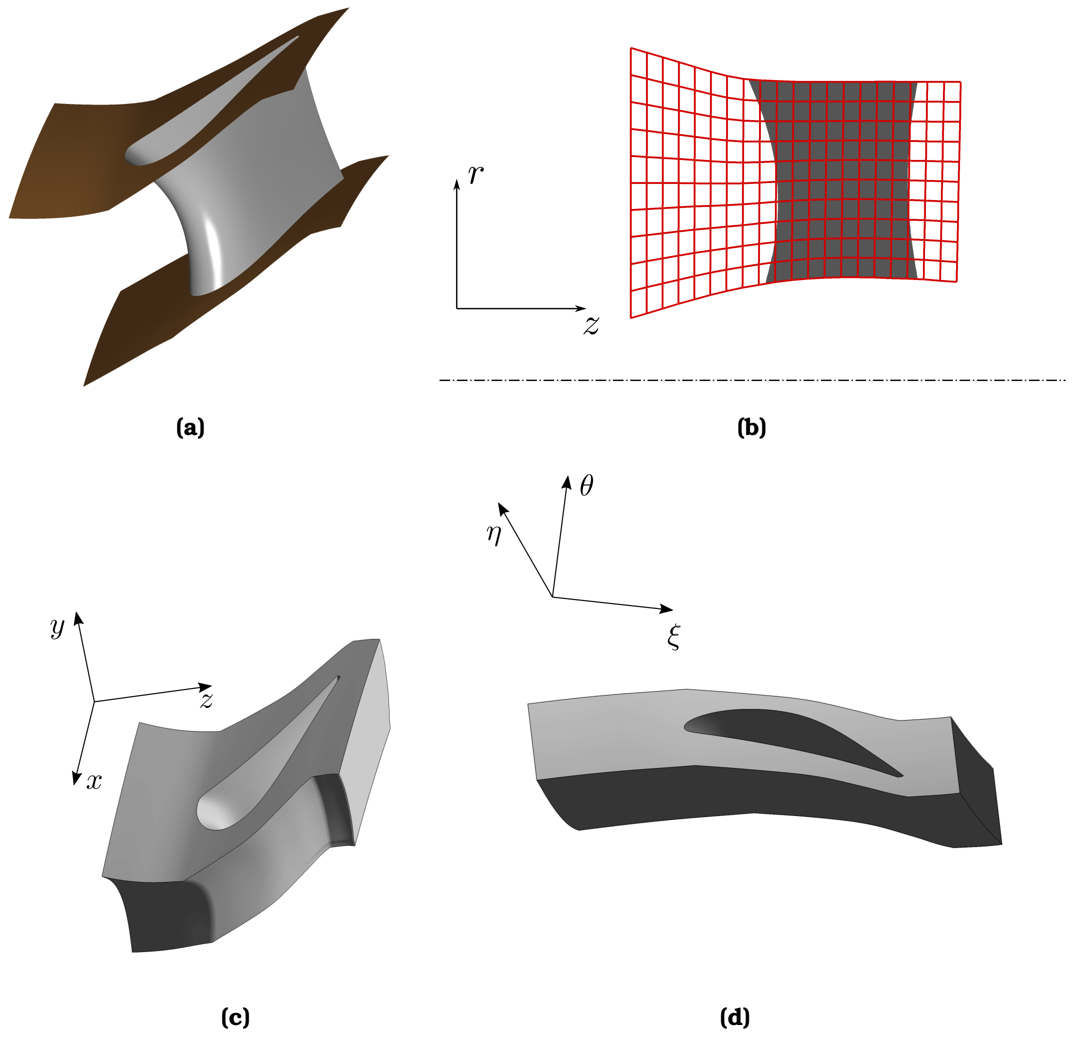


Figure 6.4: (a) The hub and shroud surface generatrices are extracted from the initial 3D geometry. (b) The PCS coordinate system is defined. Iso-ξ and iso-η lines are drawn in red. Solving Eq. 6.3.1.4 for each point on the initial 3D mesh (c) leads to the mapping of the flow domain onto the (ξ, θ, η) space (d).

Having created the PCS, a NURBS volume can be defined in this coordinate system. The advantage of such an approach is that the η coordinate of the boundary control points at hub and shroud can remain fixed. Since the θ and ξ coordinates of these points are not fixed, the surface hub and shroud mesh nodes are allowed to move by sliding along the corresponding surfaces.

6.3.2 Periodicity Treatment

The first choice that must be done to ensure geometry and mesh periodicity concerns the position of the NURBS volume control points along the circumferential direction. The boundary control points in the circumferential direction must be periodically aligned. This means that each control point on the one side of the NURBS volume is paired with a control point on the other periodic side. The points of a pair must have the same ξ and η coordinates, and their θ ones must differ by the angular pitch. By ensuring periodic displacement of the control points as well, periodicity is maintained throughout the optimization process.

Moreover, the NURBS volume may be constructed in a way that intersects the periodic boundary of the CFD mesh. In such a case, some points of the CFD mesh may lie outside the control volume. However, if the image of a mesh node on the (ξ, η) plane lies inside the projection of the control volume on the same plane, then this mesh node must be controlled and thus parametric coordinates (u, v, w) must be computed for this node. This is made possible by rotating these mesh nodes around the axis of revolution by a multiple of the angular pitch, until their transformed counterpart falls inside the NURBS volume. In PCS, this rotation is equivalent to a translation along the θ direction. An value is stored per mesh node corresponding to the times it was rotated by the angular pitch to fall inside the NURBS volume. The treatment of periodicity is shown in the form of a flowchart in Fig. 6.5. After having displaced the control points, the new position for these nodes is computed and the opposite rotation is performed to bring them in place for the new CFD mesh. The process is better explained in Fig. 6.6.

6.3.3 Computation of Geometric Sensitivities

The computation of the geometric sensitivities is partially the same as for the general case of NURBS volumes. More specifically, the relations given is section 6.2 are use to provide the term $\frac{\delta \xi_m^\ell}{\delta \Xi_{s,ijk}}$, where $m, s = 1, \dots, 3$, $(\xi_1\ \xi_2\ \xi_3) \equiv (\xi\ \theta\ \eta)$, ℓ stands for the mesh node index and $\Xi_{s,ijk}$ denotes the passage coordinates of the control point with indices i, j, k in each parametric direction. For brevity the computation of the geometric sensitivities w.r.t. the weights of the control points is omitted, since the process is the same as the one for computing the geometric sensivities w.r.t. the passage coordinates of the control points, with the only difference that the appropriate relation of section 6.2 must be employed. Let $c_p^\ell, p = 1, \dots, 3$ denote the cylindrical coordinates of mesh node ℓ. Then, by applying the chain rule, its geometric sensitivities can be expressed as

$$\frac{\delta x_r^\ell}{\delta \Xi_{s,ijk}} = \frac{\partial x_r^\ell}{\partial c_p^\ell} \frac{\partial c_p^\ell}{\partial \xi_m^\ell} \frac{\partial \xi_m^\ell}{\partial \Xi_{s,ijk}} \tag{6.3.3.1}$$

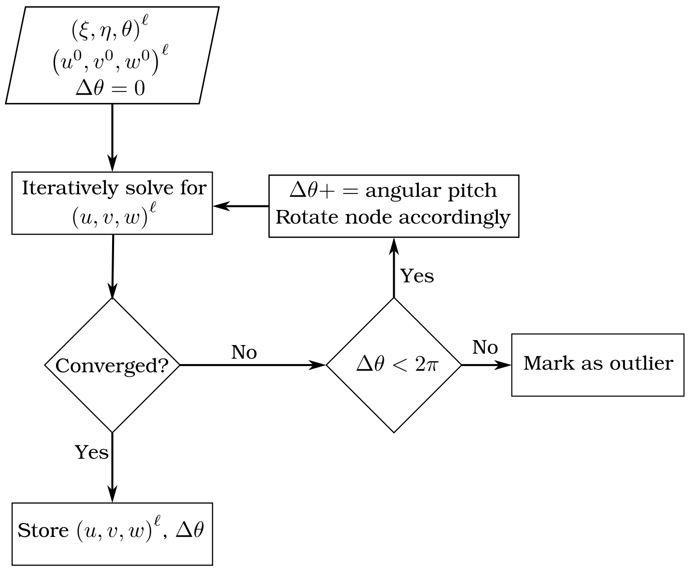


Figure 6.5: Flowchart of the series of processes executed by a single GPU thread in order to compute the parametric coordinates of a mesh node. Periodicity is taken into account.

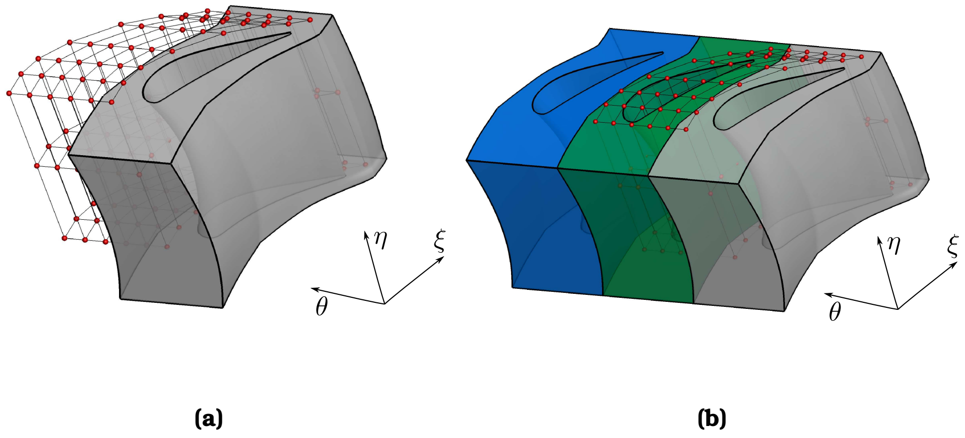


Figure 6.6: Construction of the NURBS volume on the PCS. Control points are shown in red. (a) A large part of the blade and the mesh around the blade initially lies outside the NURBS volume. (b) However, if the mesh is rotated twice by the angular pitch, then the whole blade and its surrounding mesh nodes fall inside the NURBS volume and, thus, parametric coordinates can be computed.

The last term is computed by Eq. 6.2.1. The first two terms are easily computed by the Jacobian matrices of the corresponding transformations given as

$$\frac{\partial x_r^\ell}{\partial c_p^\ell} = \begin{bmatrix} \cos\theta^\ell & -r^\ell \sin\theta^\ell & 0 \\ \sin\theta^\ell & r^\ell \cos\theta^\ell & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3.3.2}$$

and

$$\frac{\partial c_p^\ell}{\partial \xi_m^\ell} = \begin{bmatrix} \left(1-\eta^\ell\right)\dot{r}^H\left(\xi^\ell\right) + \eta^\ell \dot{r}^S\left(\xi^\ell\right) & 0 & -r^H\left(\xi^\ell\right) + r^S\left(\xi^\ell\right) \\ 0 & 1 & 0 \\ \left(1-\eta^\ell\right)\dot{z}^H\left(\xi^\ell\right) + \eta^\ell \dot{z}^S\left(\xi^\ell\right) & 0 & -z^H\left(\xi^\ell\right) + z^S\left(\xi^\ell\right) \end{bmatrix} \tag{6.3.3.3}$$

In Eq. 6.3.3.3 the $\left(z^H, r^H\right)$ and $\left(z^S, r^S\right)$ denote the radial and axial coordinate for the hub and shroud generatrices at the parametric position ξ^ℓ, respectively. Moreover, the dot over them corresponds to differentiation w.r.t. the parameter ξ.

Chapter 7

Shape Optimization of Turbomachinery Components using Evolutionary Algorithms

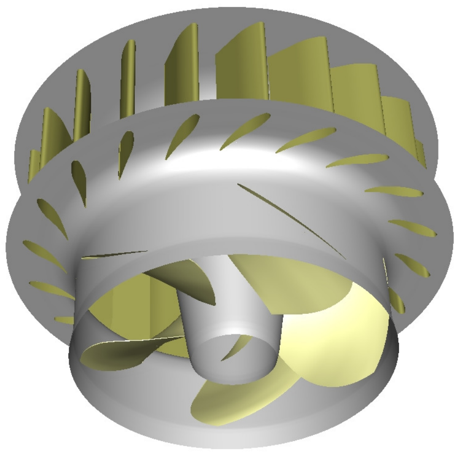
In this chapter a series of turbomachinery component shapes are optimized. The configurations to be optimized are parameterized using the tool GMTurbo described in Chapter 5. The PUMA CFD code running on GPUs described in Chapter 2 will be used to evaluate the performance of each candidate solution during the optimization. The optimization itself will be performed using the Evolutionary Algorithm (EA) software EASY developed by PCOpt/LTT [1].

The use of EAs is preferred at this point instead of the use of gradient-based (GB) optimization methods, since EAs tend to converge to the global optimum (within the design space, as determined by the user-defined bounds of the design variables), while GB methods are more easily trapped to local minima/maxima (i.e. stationary points). Making use of GMTurbo for the parameterization of the geometry leads to a compact representation, thus, reducing the number of design variables and avoiding the curse of dimensionality (a typical problem in EA-based optimization processes), while maintaining a reasonably rich variety of shapes that may arise.

Since the shapes to be optimized are initially available in either neutral CAD formats (as NURBS surfaces) or in the form of a CFD mesh, a process capable of extracting the GMTurbo parameters that reconstruct the given geometry is first needed. Then, the original CFD mesh (if available) is adapted to the reparameterized geometry. Finally, for each shape modification throughout the optimization, the initial CFD mesh is adapted accordingly, using mesh deformation techniques. These three steps are described in appendices A and B.

7.1 Shape Optimization of a Propeller Type Turbine Runner and Guide Vanes

The first case is concerned with the optimization of a propeller type hydraulic turbine runner with guide vanes. The initial geometry and its corresponding CFD mesh are provided by ANDRITZ Hydro. The configuration consists of 20 inlet guide vanes (IGVs) and 5 runner blades. The tip gaps in both the runner and the guide vane bladings are not modeled. Water enters radially and encounters the IGVs, then, the runner blades and exits axially. A steady state solution is sought using the mixing plane technique (circumferential averaging of flow quantities) to account for the interaction between the stationary guide vanes and the rotating rotor blades. An overview of the configuration is presented in Fig. 7.1.


(a)

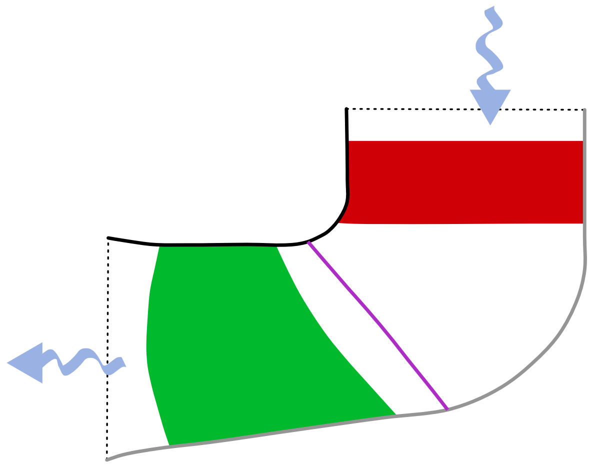
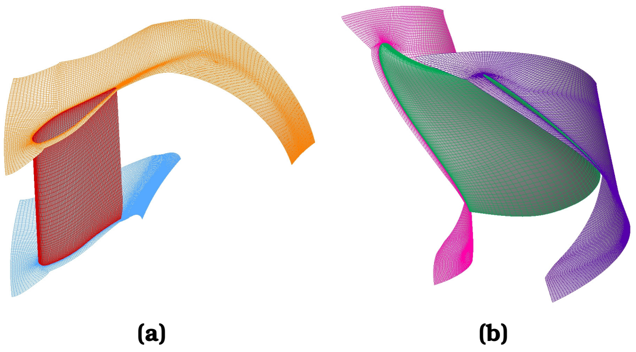

(b)

Figure 7.1: *Propeller Water Turbine*: (a) 3D view of the propeller water turbine configuration. The 20 IGVs and 5 runner blades are visible in gold. The hub and shroud surfaces are drawn in grey. No tip gap is modeled. (b) Meridional section of the IGV and runner configuration. The IGV is drawn in red while the runner blade in green. The grey and black lines correspond to the hub and shroud generatrices, respectively. The position of the interface between the stationary IGVs and the rotating runner is shown in purple. Blue arrows show the direction of the water stream through the turbine.

The baseline mesh, provided by ANDRITZ Hydro is composed by two submeshes. The first, consisting of $\sim 6 \times 10^5$ nodes and $\sim 5.8 \times 10^5$ hexahedra, is used to model a single IGV. The second, used to model the runner, consists of $\sim 4.5 \times 10^5$ nodes and $\sim 4.3 \times 10^5$ hexahedra. The surface mesh for a single IGV and a single runner blade is shown in Fig. 7.2.

Data for the operating conditions of the turbine are presented in Table 7.1.



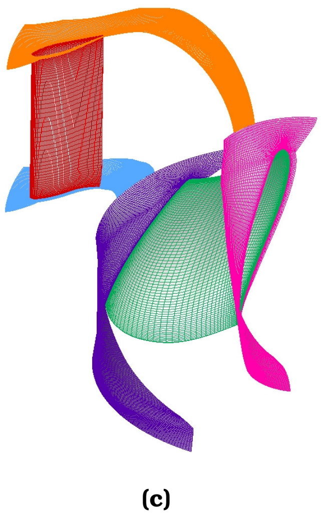


Figure 7.2: *Propeller Water Turbine*: (a) Computational surface mesh for the IGV. The hub is shown in orange and the shroud in blue. (b) Computational surface mesh for the runner. The hub is shown in magenta and the shroud surface in purple. (c) Both the IGV and runner meshes combined. The IGV itself is shown in red, while the runner blade mesh in green.

Quantity	**Symbol**	**Value**
Hydraulic head	H	$13.11\,\mathrm{m}$
Runner angular speed	n	1651.86 rpm
Specific speed	n_s	524.02
Inlet swirl	α_1	$45°$
Inlet Viscosity Ratio	$(\nu_t/\nu)^{\mathrm{in}}$	20.0

Table 7.1: *Propeller Water Turbine*: Operating conditions. The turbine specific speed in computed with the power output in kW.

Before proceeding to the parameterization and optimization, a simulation is carried out and the results of PUMA are compared with those provided by

ANDRITZ Hydro.

The simulation is carried out on the National HPC Infrastructure ARIS of the Greek Research & Technology Network [157]. PUMA ran on two NVIDIA Tesla K40 GPUs on the same computational node, for $\sim 1\,\mathrm{h}$ and $15\,\mathrm{min}$. The pressure coefficient (c_p) is shown in Fig. 7.3 where

$$c_p = \frac{p - p_{\text{vap}}}{\rho g H}$$

where p stands for the static pressure, ρ the constant water density, g the gravitational acceleration, H the hydraulic head of the turbine and p_{vap} the water vapour pressure.

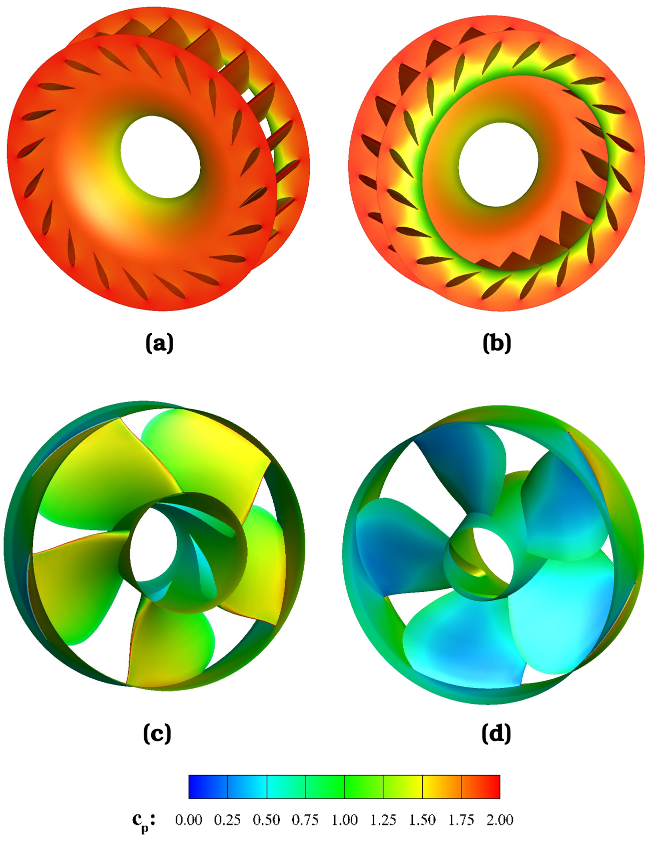


Figure 7.3: *Propeller Water Turbine*: Pressure coefficient on the IGV row wall surfaces. View from (a) the hub and (b) the shroud side. Pressure coefficient on the runner wall surfaces. View of the blades' (c) pressure and (d) suction side.

The comparison between the two CFD solvers is shown in Fig. 7.4. The commercial software used by ANDRITZ Hydro and PUMA are in good agreement with each other, , even though different turbulence models have been employed, namely, the PUMA solver uses the Spalart-Allmaras model, while k-ω SST model is used in the commercial software simulations.

7.1.1 Parameterization with GMTurbo

The process described in appendix A is used to extract the design parameters used by GMTurbo to generate the geometry. Using more design parameters to GMTurbo reduces the error of approximating the geometry as described by the CFD mesh provided but, in turn, has a negative impact on the compactness of the parameterization. In this case, 8 generatrices were used to extract the parameters that generate the IGV and runner blade. Then, the initial mesh was adapted so as to match exactly the reconstructed geometry. Fig. 7.5 presents the error of the geometry reconstruction. The local error is measured by the quantity Δ which corresponds to the displacement of a mesh node as a percentage of the midspan chord c_{mid} (which is different between IGV and runner blade), namely

$$\Delta\% = 100\frac{\mathbf{r}-\mathbf{r}_0}{c_{\text{mid}}}$$

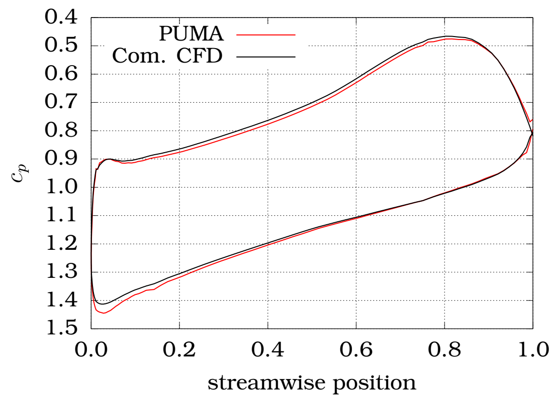
where $\mathbf{r}$ is the position vector of the displaced node and $\mathbf{r}_0$ the initial position vector of the same node.

In order to quantify the effect of the geometry reconstruction on the value of integral quantities of engineering interest, four CFD runs have been performed. The first run corresponds to the baseline geometry as provided in the form of a CFD mesh (the corresponding results are presented in the figures above). The second run is performed on the baseline runner geometry but using the reconstructed IGV one. The third run refers to the evaluation of the baseline IGV geometry coupled with the reconstructed runner and, finally, the fourth run evaluates the fully reconstructed configuration (both runner and IGV). The results of the first run are used as reference to quantify the deviation of each quantity due to reconstruction. The geometry reconstruction effect is summarized in Table 7.2.

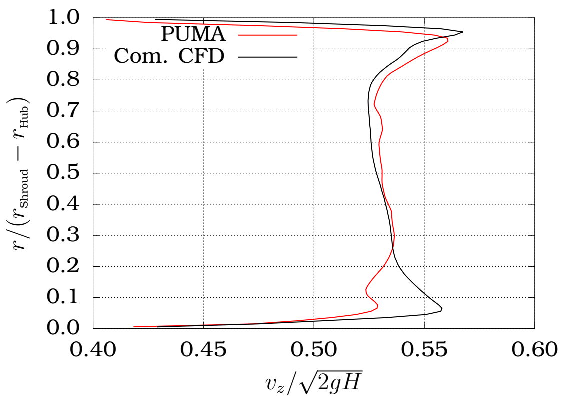
It is evident that the quality of the reconstruction is adequate for starting an optimization loop and considering that the reconstructed geometry performance is representative for the baseline design as well. The maximum error in flow quantities is bound below 0.2%.

7.1.2 Optimizing the IGV with Baseline Runner

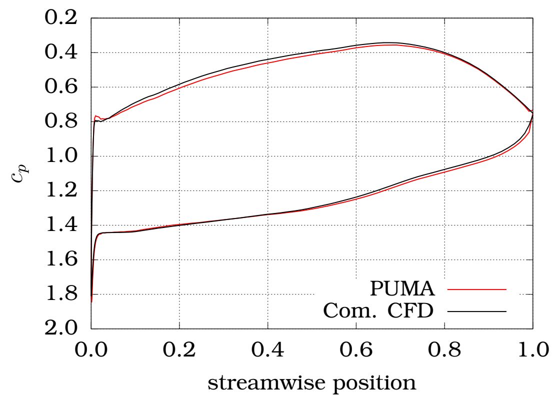
The first loop concerns the optimization of the IGV while keeping the runner blade geometry fixed. The design vector consists of 32 design variables which are summarized in Table 7.3. The design variables together with their bounds (defining the search space of the MAEA) are shown in Fig. 7.6. The maximization of the efficiency will be the optimization objective.



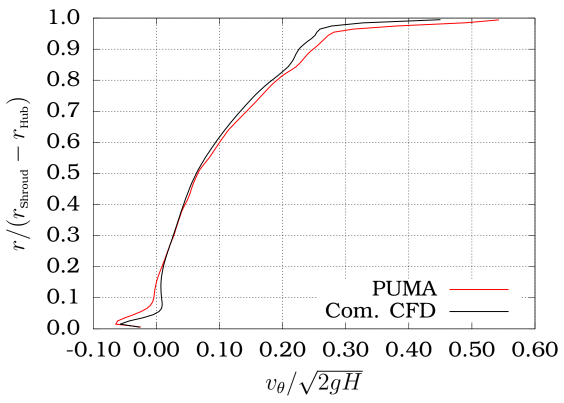
(a)



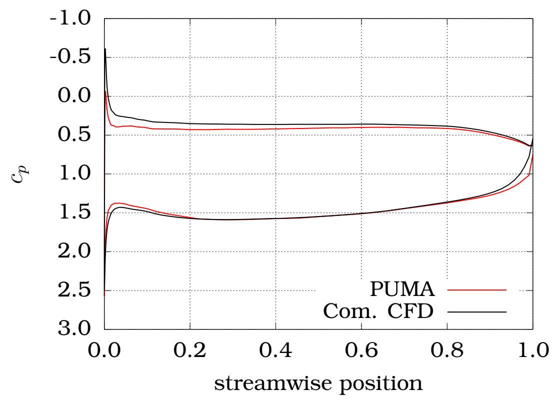
(d)



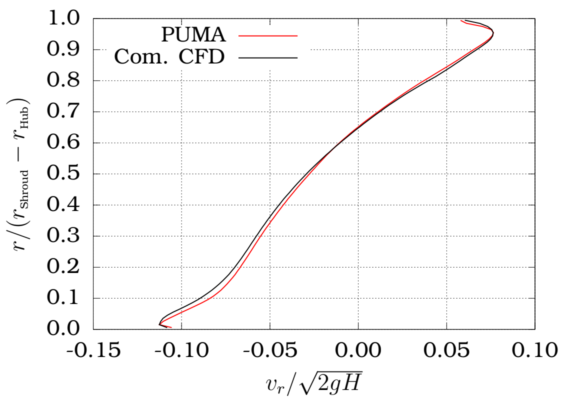
(b)



(e)



(c)



(f)

Figure 7.4: *Propeller Water Turbine*: Comparison between the results obtained by PUMA and those provided by Andritz HYDRO using a commercial CFD solver (Com. CFD). Pressure coefficient distribution on the runner blade at (a) the hub, (b) midspan and (c). The distributions are in good agreement. Outlet spanwise distributions of the non-dimensionalized (d) axial, (e) circumferential and (f) radial velocity components. The distributions are in good agreement apart from the axial velocity one close to the hub region. This may be attributed to the absence of the hub gap of the IGV which, however, is present in the simulation performed by ANDRITZ Hydro with the commercial software.

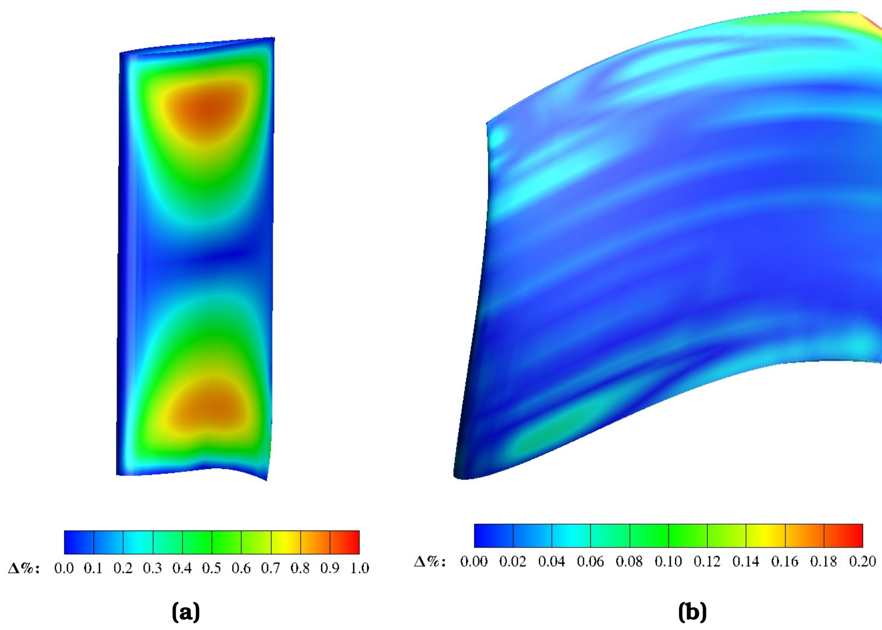

Δ%: 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
Δ%: 0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20
(a)
(b)

Figure 7.5: *Propeller Water Turbine*: Geometry reconstruction error. For the reconstruction, 8 sections of each blade were used. (a) The maximum error in the IGV geometry is $\sim 1\%$ based on the midspan chord length. (b) For the runner blade, the maximum geometry reconstruction error is $\sim 0.2\%$ of the blades midspan chord.

A $(10, 20)$ MAEA (Metamodel-Assisted EA with 10 parents and 20 offspring) is employed for the optimization. RBF networks are used as metamodels to speed-up the optimization process, once 30 candidate solutions have been successfully evaluated. The overall computational budget is restricted to 200 CFD solver calls.

The comparison between the baseline and optimized IGV geometry is shown in Fig. 7.7. The optimization convergence history is shown in Fig. 7.8.

As can be seen in Fig. 7.8, the improvement of the turbine efficiency is minor. This is expected since the runner shape remains the same and the guide vanes are mainly used to control the flow rate. Altering the thickness of the blade would potentially lead greater improvement in efficiency (due to a consequent reduction in viscous losses). However, the blade's thickness is kept constant due to manufacturing constraints.

The runner shape is expected to have a more pronounced effect on the turbine efficiency. This is investigated in the next subsection.

7.1.3 Optimizing the Runner with Baseline IGV

This optimization test case concerns the maximization of the turbine's efficiency by changing only the shape of the runner blade, while keeping the IGV shape fixed.

Error in hydraulic head ϵ_H $(\%)$

IGV Runner	Baseline	Reconstructed
Baseline	0.0000	0.0031
Reconstructed	-0.0272	-0.0240

Error in flow rate ϵ_Q $(\%)$

IGV Runner	Baseline	Reconstructed
Baseline	0.0000	0.0010
Reconstructed	-0.0384	-0.0326

Error in power output ϵ_P $(\%)$

IGV Runner	Baseline	Reconstructed
Baseline	0.0000	0.0146
Reconstructed	-0.1594	-0.1312

Error in efficiency ϵ_η $(\%)$

IGV Runner	Baseline	Reconstructed
Baseline	0.0000	0.0106
Reconstructed	-0.0938	-0.0751

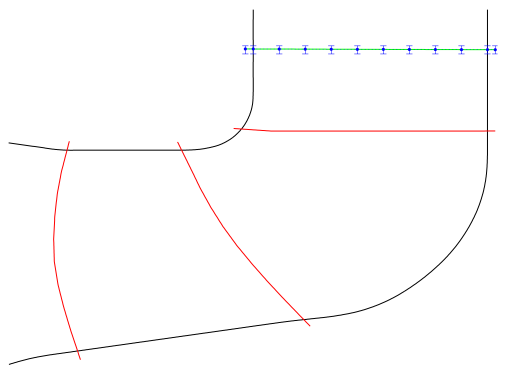
Table 7.2: *Propeller Water Turbine*: Effect of geometry reconstruction error on the head, flow rate, power output and efficiency of the propeller turbine. For all quantities, the configuration consisting of both baseline geometries is used as reference.

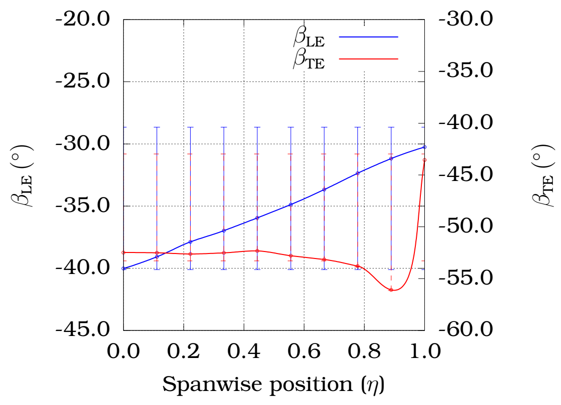
Design Variable Type	Number of Variables
r_{LE}	12
β_{LE}	10
β_{TE}	10

Table 7.3: *Propeller Water Turbine IGV only Optimization*: The 32 design variables used for optimizing the IGV while keeping the runner geometry fixed.

A total of 30 design variables are used. More specifically, 10 controlling the shape of the runner blade LE meridional projection, 10 controlling the β_{LE} spanwise distribution and, another 10 controlling the β_{TE} one. The design variables bounds are shown in Fig. 7.9.

A $(10, 20)$ MAEA is used with the same settings as the one used in the optimization of subsection 7.1.2. The computational budget in this case is restricted to 250 CFD evaluations. The convergence history of the optimization is shown in Fig. 7.10. In this case, the gain is greater compared to optimizing the IGV only.


(a)



(b)

Figure 7.6: *Propeller Water Turbine IGV only Optimization*: Design variables used for optimizing the IGV shape. (a) Design variables controlling the shape of the IGV LE on the meridional plane. The bounds of the design space are shown as bars. (b) Design variables controlling the spanwise distribution of the LE and TE metal angle (β_{LE}, β_{TE}). Bars also show the bounds for each design variable.

This is due to the essential role of the runner in extracting energy from the fluid and transforming it into mechanical energy on the turbine's shaft. In addition, the wetted surface of the runner blades is much greater, thus, generating more viscous losses compared to the IGVs.

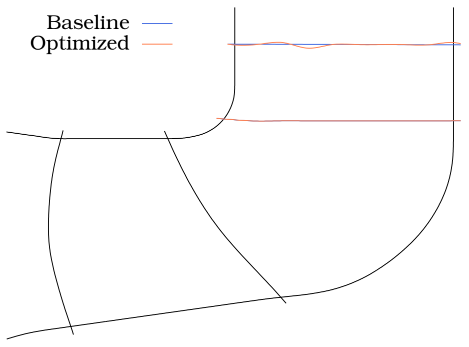
The resulting runner blade shape is presented in Fig. 7.11. It can be seen that the optimization leads to a design with a wavy trailing edge shape. This is shown in Fig. 7.11b as well as in Fig. 7.11d. In contrast to the TE, the optimization leads to a less wavy LE as seen by Fig. 7.11c.

7.1.4 Combined Optimization of the Runner and IGV

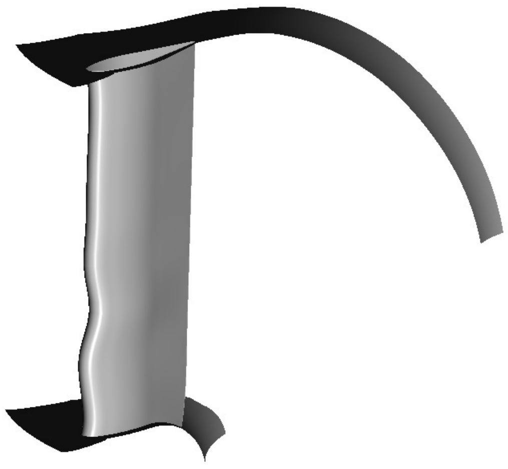
In this optimization problem the turbine IGV and runner shapes are allowed to vary concurrently. The design variables set is the union of the ones used in the optimizations presented in the two previous subsections and, consequently, the dimensionality of the design space is equal to 62. The objective function is still the turbine efficiency, whose maximization is sought.

The convergence history of the optimization is shown in Fig. 7.12.

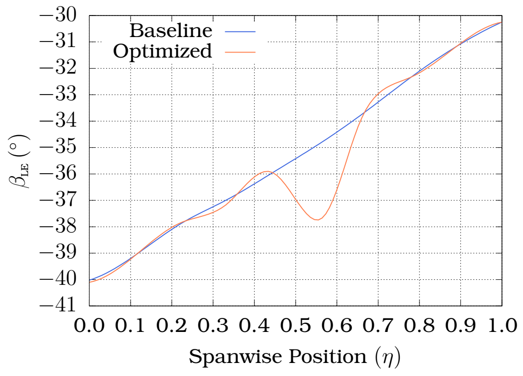
It can be seen that optimizing both the IGV and runner geometries concurrently is beneficial since the optimal solution reached dominates the ones achieved by optimizing each row separately. In addition, combining the geometries resulting from the two first runs (IGV only and runner only optimization) proves worse compared even to the initial one. In fact, the efficiency is by 0.6% lower compared to the baseline one. This gives the message that the Rotor-Stator Interaction



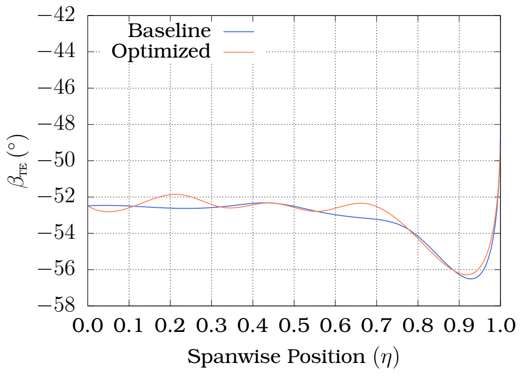
(a)



(b)



(c)



(d)

Figure 7.7: *Propeller Water Turbine IGV only Optimization*: Comparison between the baseline and optimized IGV shape. (a) Comparison of the meridional shape of the IGV leading edge. A much more wavy shape is obtained by the optimization process. (b) The same can be seen in the 3D shape of the IGV. (c) Comparison of the leading edge metal angle (β_{LE}) spanwise distribution. (d) Comparison of the trailing edge metal angle (β_{TE}) spanwise distribution.

must be taken into account during the optimization, otherwise it may lead to non-optimized designs.

In Fig. 7.13, the meridional shapes obtained by the three different optimization runs, as well as the spanwise distributions of the blades metal angles are compared. It can be seen how the IGV shape is adapted to modifications of the runner and vice-versa.

In Fig. 7.14, the 3D shape of the optimized turbine IGV and runner and the pressure coefficient distribution on the solid surfaces are shown. The wavy shape of the optimized blade for both the IGV and the runner is shown.

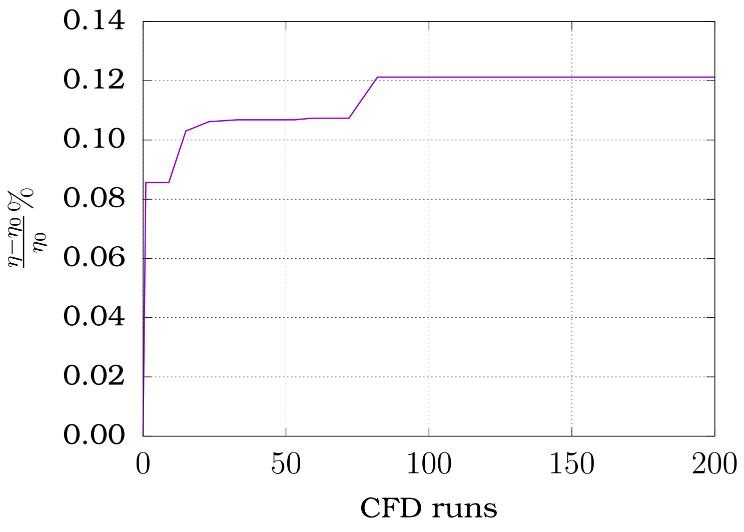
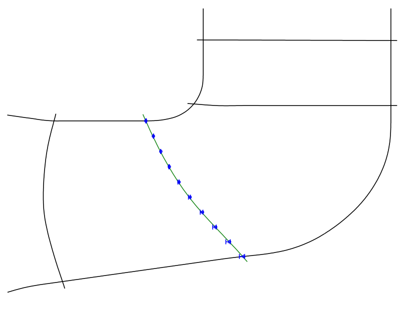
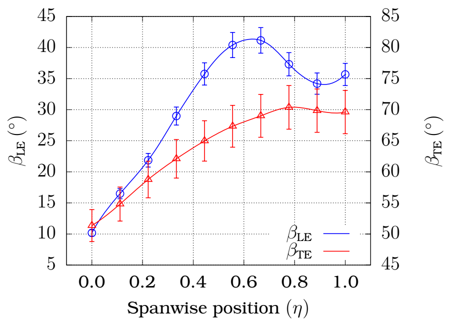


Figure 7.8: *Propeller Water Turbine IGV only Optimization*: Convergence history of the optimization. The objective function improvement is shown as percentage of the baseline performance.



(a)



(b)

Figure 7.9: *Propeller Water Turbine Runner Only Optimization*: (a) Design variables used to control the meridional shape of the runner blade LE. Points are allowed to move only in the axial direction. (b) Design variables controlling the LE and TE metal angle spanwise distribution for the runner blade. The bounds of each design variable (defining the search space of the MAEA) are shown with error bars.

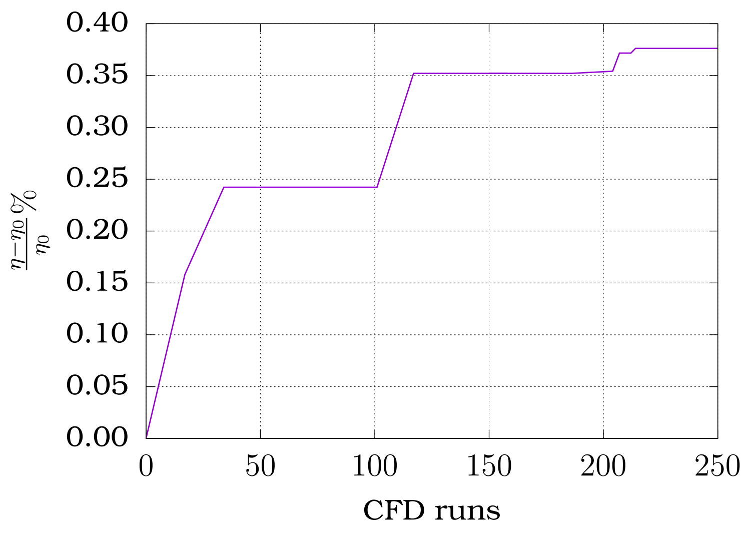
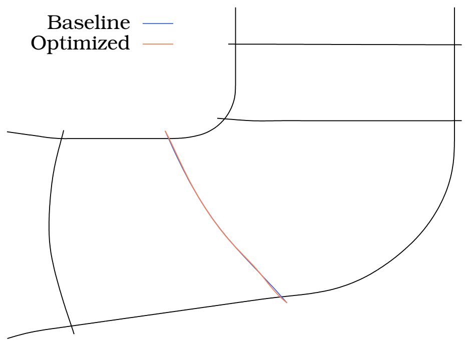
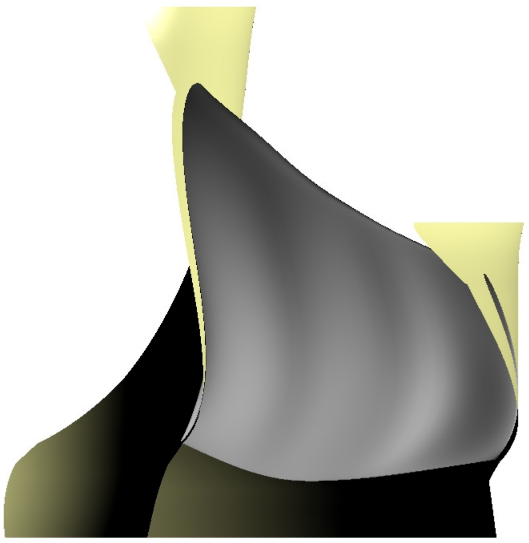


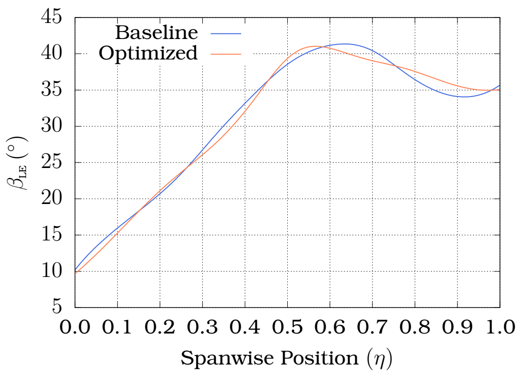
Figure 7.10: *Propeller Water Turbine Runner Only Optimization*: Convergence history of the optimization. The objective function improvement as percentage of the baseline performance reaches approximately 0.37% at a computational cost of ~ 250 CFD runs.



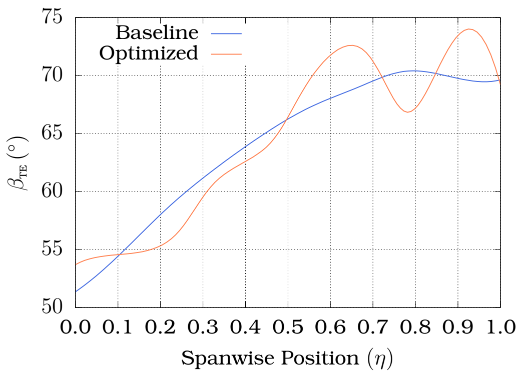
(a)



(b)



(c)



(d)

Figure 7.11: *Propeller Water Turbine Runner Only Optimization*: (a) Comparison of the runner blade LE meridional shape with the baseline one. The modification is minor. (b) 3D shape of the optimized blade. The wavy shape at the trailing edge can be seen. (c) Comparison between the β_{LE} spanwise distribution of the baseline geometry and the optimized one. (d) Same comparison for the β_{TE} spanwise distribution.

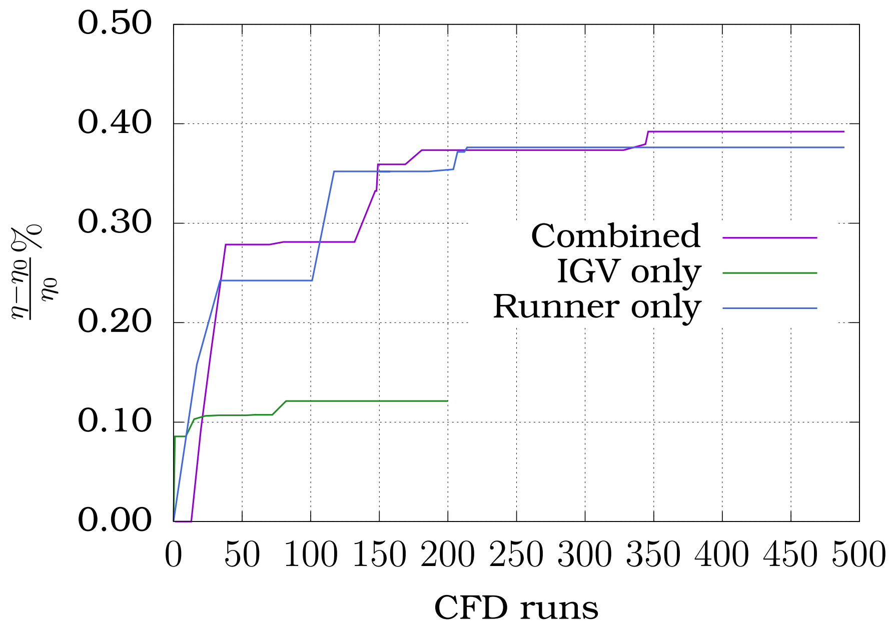
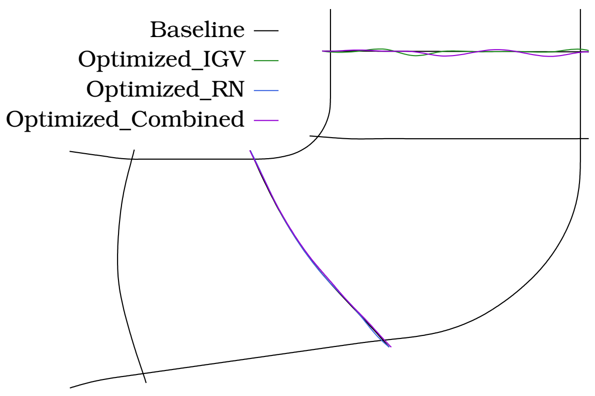
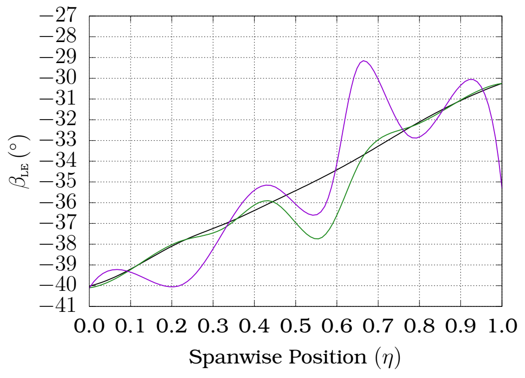


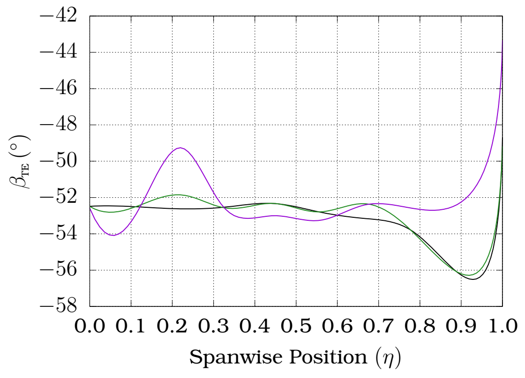
Figure 7.12: *Propeller Water Turbine IGV and Runner Optimization*: Convergence history of the three optimizations performed. It can be seen that simultaneously optimizing the IGV and runner blades, leads to a better solution, than optimizing one row at a time. The optimization for the IGV only case (green line) was stopped earlier, since the improvements without modifications of the runner geometry were minor. The runner only optimization (blue line) could not improve the design's performance during the last half of the optimization (after first 250 CFD evaluations). However, for the simultaneous optimization (purple line), even though more CFD runs are needed to reach a comparable solution to the one optimizing only the runner (due to increased dimensionality of the design space), finally, a better solution has been achieved.



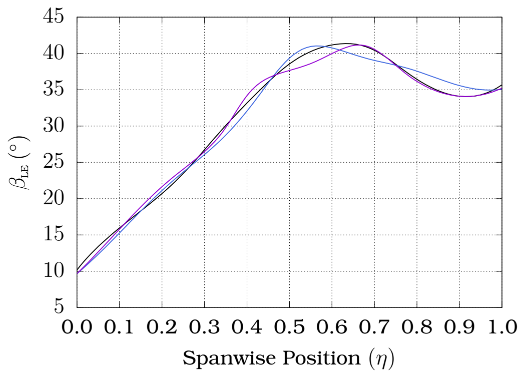
(a)



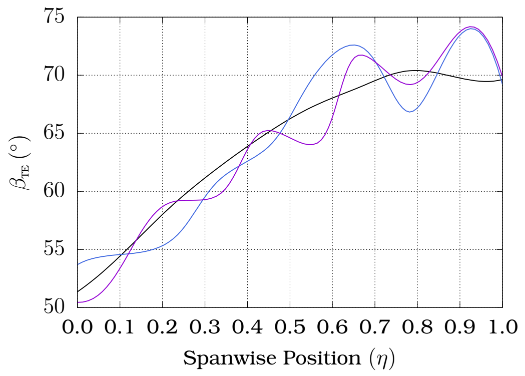
(b)



(c)



(d)



(e)

Figure 7.13: *Propeller Water Turbine IGV and Runner Optimization*: Comparison of the shapes obtained by the three separate optimization runs. The same color code is used in all figures, namely black lines correspond to the baseline geometry, green ones to the IGV only optimized, blue ones to the runner only optimized and purple onces in the simultaneous optimization geometry. (a) Comparison of the meridional shape of the IGV LE. Comparison of the spanwise distributions of (b) β_{LE} of the IGV, (c) β_{TE} of the IGV, (d) β_{LE} of the runner and (e) β_{TE} of the runner.

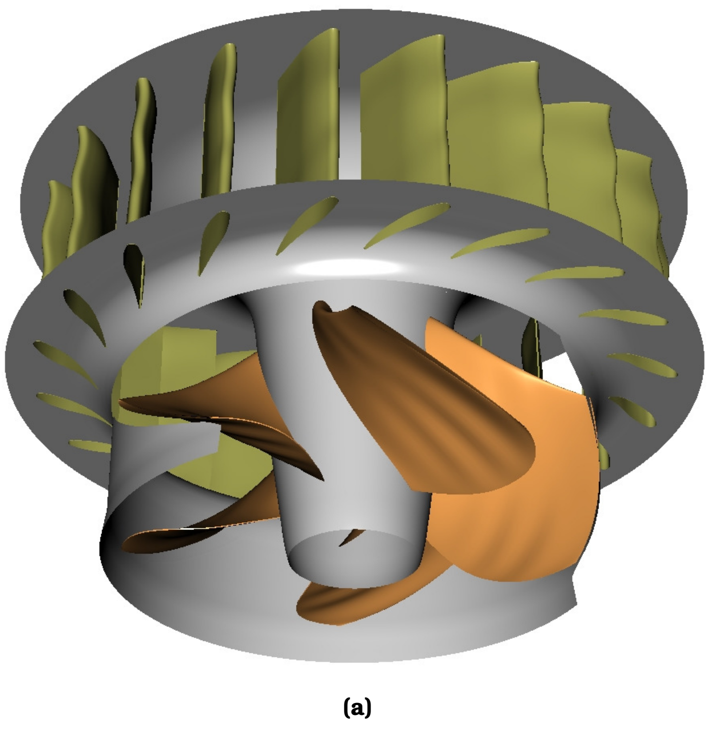
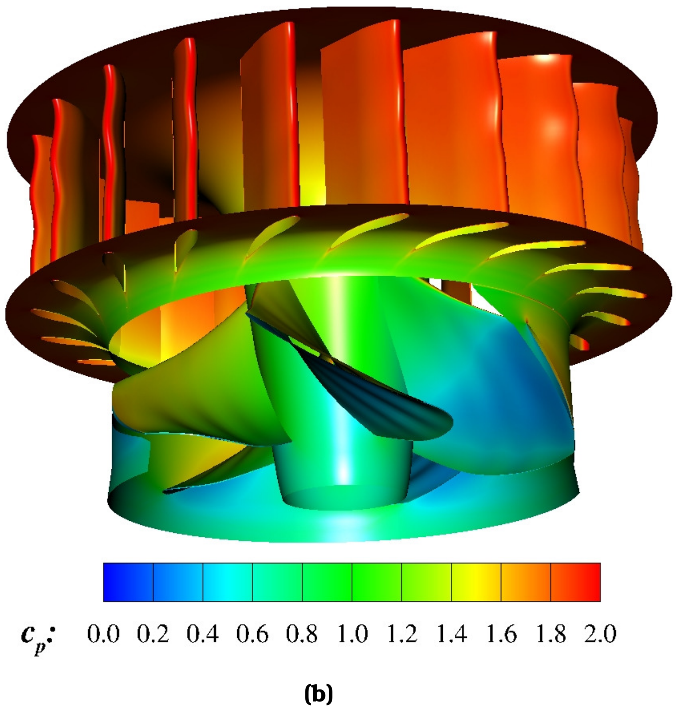



Figure 7.14: *Propeller Water Turbine IGV and Runner Optimization*: (a) Shape arising from the optimization. The large number of control points used to define the spanwise distribution of the metal angles for the IGV and runner blades, allows wavy shapes as the optimized ones to arise during the optimization. (b) Pressure coefficient drawn on the solid surfaces of the water turbine.

Chapter 8

Shape Optimization using Adjoint

In this chapter, the continuous adjoint technique (developed in Chapter 3) is coupled with parameterization methods presented in Chapters 4 and 6 and used to assist gradient-based methods (such as steepest-descent) for the shape optimization of external and internal aerodynamic shapes, including turbomachinery.

In contrast to the parameterization method used for turbomachinery components in the applications of Chapter 7, the parameterization methods used in this chapter are characterized by a much larger number of design variables. This makes the adjoint method an appealing (one might say, an "exclusive") option since the cost of computing the gradient of the objective function is independent of the number of design variables. Even though the free-form parameterization methods, used herein, lack the compactness of the CAD based method used in the applications of Chapter 7, they provide the means to deform the CFD mesh around the geometry under consideration at a much lower cost compared to the mesh deformation methods used to adapt the CFD mesh (described in Appendix B). Furthermore, these methods allow the use of the FI adjoint variant, which provides more accurate sensitivity derivatives, with practically negligible extra cost compared to the SI one.

8.1 Optimization of a Transonic Airfoil

The first application concerns the optimization of a transonic airfoil. The RAE 2822 airfoil is used as the baseline shape to be optimized. A mesh of $\sim 4.8 \times 10^4$ nodes, consisting exclusively of quadrilateral elements, is used for the discretization of the flow equations around the airfoil. The flow is turbulent and the Spalart-Allmaras turbulence model is employed. The flow conditions are presented in Table 8.1. These flow conditions correspond to an infinite flow Mach number $M_\infty = 0.729$ and a Reynolds number based on the airfoil chord of $\mathrm{Re}_c = 6.5 \times 10^6$.

A 8×7 control point NURBS volume is used to parameterize the shape of the

Quantity	**Symbol**	**Value**
Infinite Flow Pressure	p_∞	$1.013\,25 \times 10^5\,\mathrm{Pa}$
Infinite Flow Density	ρ_∞	$1.2\,\mathrm{kg/m^3}$
Infinite Flow Velocity	V_∞	$250.55\,\mathrm{m/\,s}$
Infinite Flow Angle	α_∞	$2.31°$
Air Dynamic Viscosity	μ	$4.627 \times 10^{-5}\,\mathrm{kg/m\,s}$
Infinite Viscosity Ratio	$(\mu_t/\mu)_\infty$	1.0

Table 8.1: *Transonic Airfoil Optimization*: Flow conditions.

airfoil, as well as to deform the mesh around it. The weights of all control points are kept fixed throughout the optimization and only some of them are allowed to move along the y-direction. The mesh around the airfoil and the NURBS control volume are shown in Fig. 8.1. The B-Spline basis functions are of degree 2 in both parametric directions.

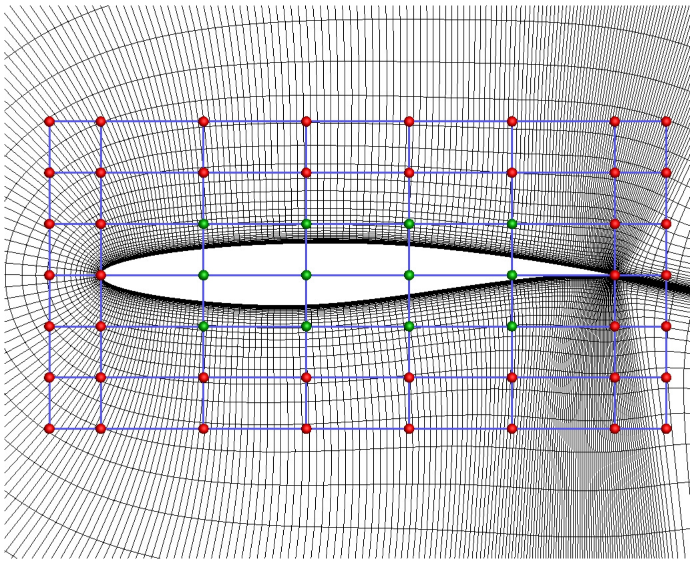


Figure 8.1: *Transonic Airfoil Optimization*: Mesh around the transonic airfoil. The mesh provides a distance of the first node off the wall that leads to a $y^+ < 1$. The airfoil and the surrounding mesh are parameterized by a 8×7 NURBS control volume (blue lines). Control points in green are allowed to move along the y-direction. Two layers of control points in red are kept fixed so as to ensure $\mathcal{C}^1$ continuity with the surrounding, rigid mesh. Control points are not allowed to move in the x-direction constraining the airfoil chord to its initial value.

A flow field solution takes on the average $\sim 1\,\mathrm{min}$ to converge on a singe NVIDIA Tesla K40 GPU. Two separate optimization runs are performed. One for maximizing the lift coefficient (c_L) of the airfoil and one for minimizing its drag coefficient (c_D). Since the airfoil chord remains constant, optimizing for the lift or drag coefficients is the same as optimizing for the force components that are associated with these coefficients, namely the lift (L) and drag forces (D). These are expressed as

$$\begin{aligned} L &= \int\limits_{S_{\text{Obj}}} (p\mathsf{n}_k - \tau_{km}\mathsf{n}_m)\, r_k^L \mathrm{d}S \\ D &= \int\limits_{S_{\text{Obj}}} (p\mathsf{n}_k - \tau_{km}\mathsf{n}_m)\, r_k^D \mathrm{d}S \end{aligned} \tag{8.1.1}$$

where the indices $(k, m = 1, 2)$, $r^L = (-\sin\alpha_\infty, \cos\alpha_\infty)$ and $r^D = (\cos\alpha_\infty, \sin\alpha_\infty)$.

From Eqs. 8.1.1 and taking Eqs. 3.1.4.49-3.1.4.56 into account, the following adjoint boundary conditions arise at the airfoil boundaries

$$\begin{aligned} \Psi_{m+1} &= -r_m, \;\; m = 1, 2 \\ \tilde{\nu}_a &= 0 \\ q_k^{\text{adj}}\mathsf{n}_k &= 0 \end{aligned} \tag{8.1.2}$$

where r_m can be either r_m^L or r_m^D depending on the chosen objective.

For both objective functions, the adjoint field equations converge in approximately the same number of pseudo-time iterations and need $\sim 45\,\mathrm{s}$ each on the same hardware (NVIDIA Tesla K40 GPU). Fig. 8.2 presents the fields of the adjoint momentum $\left(\sqrt{\Psi_{m+1}^2}, m = 1, 2\right)$ and adjoint Spalart-Allmaras variable $(\tilde{\nu}_a)$ for the two objectives.

The steepest-descent method is used to drive the optimization towards the optimum. The convergence history of the two optimization runs is shown in Fig. 8.3.

The optimization for $\max c_L$ stops prematurely (before reaching a stationary point) after 9 cycles, because the mesh morphing led to a mesh with invalid elements. On the other hand, the optimization for $\min c_D$ drag stops after 15 cycles after reaching the available computational budget set for the run. It can be noted that a stationary point (local minimum) is approached. However, it can be seen that the more the optimum is approached the more the optimization appears to stagnate. This is a known disadvantage of the steepest-descent method.

The shapes of the airfoils resulting from the two optimizations are compared to the baseline one in Fig. 8.4. In the same figure, the pressure coefficient distribution along the three different airfoils are compared. It can be seen that the optimization for $\min c_D$ resulted in a slightly flatter suction side reducing significantly the shock intensity and, consequently, shock induced drag. On the other

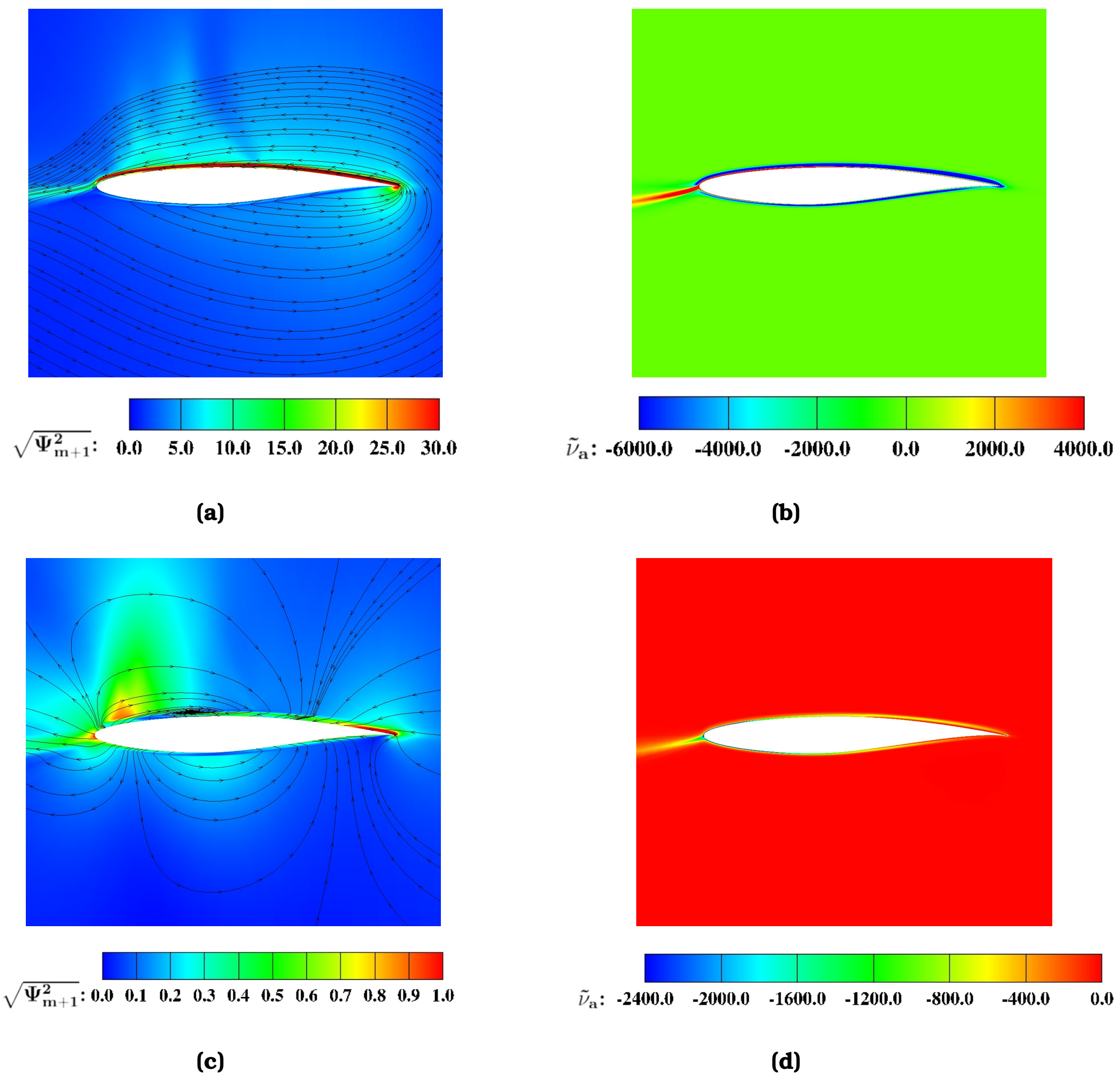
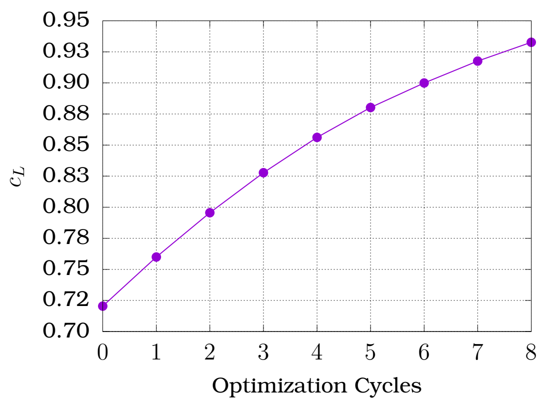


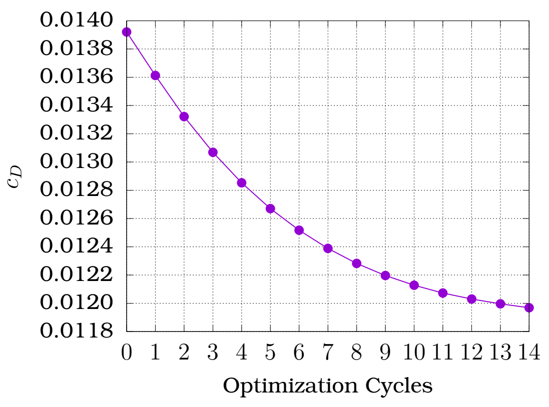
Figure 8.2: *Transonic Airfoil Optimization*: (a), (c) Field of adjoint momentum magnitude for the lift and drag as the objective function, respectively. Streamlines of adjoint momentum are also drawn. (b), (d) Spalart-Allmaras variable field for the lift and drag as the objective function, respectively.

hand, the optimization for $\max c_L$ led to an airfoil shape that accelerated the flow further downstream along the suction side, moving the shock wave towards the trailing edge (TE) and avoiding the shock compression for a longer part of the airfoil, thus, increasing lift.

The Mach number field around the baseline, the $\max c_L$ and $\min c_D$ airfoils are presented in Fig. 8.5.

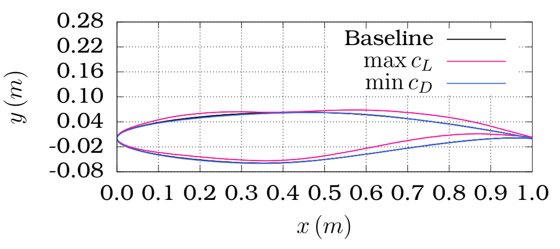


(a)

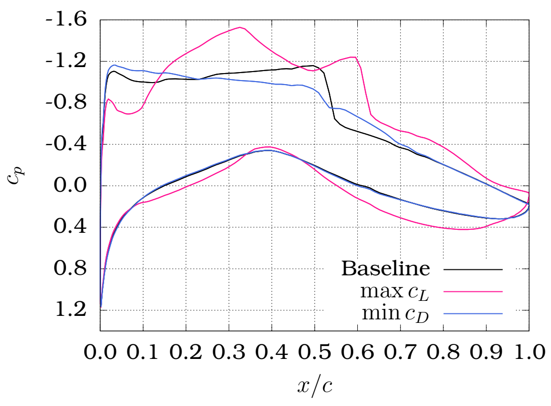


(b)

Figure 8.3: *Transonic Airfoil Optimization*: Optimization history for (a) Maximum lift coefficient $(\max c_L)$ and (b) Minimum drag coefficient $(\min c_D)$.



(a)



(b)

Figure 8.4: *Transonic Airfoil Optimization*: (a) Comparison of the baseline airfoil shape and the ones arising from the optimization for $\max c_L$ and $\min c_D$. The shape of the airfoil for $\min c_D$ is very close to the baseline one but slightly flatter on the suction side. For $\max c_L$ the airfoil is less cambered in its mid-section (delaying the shock formation) but cambers more drastically close to the trailing edge. The effect of shape changes on the distribution of the pressure coefficient (c_p) along the airfoil is shown in (b).

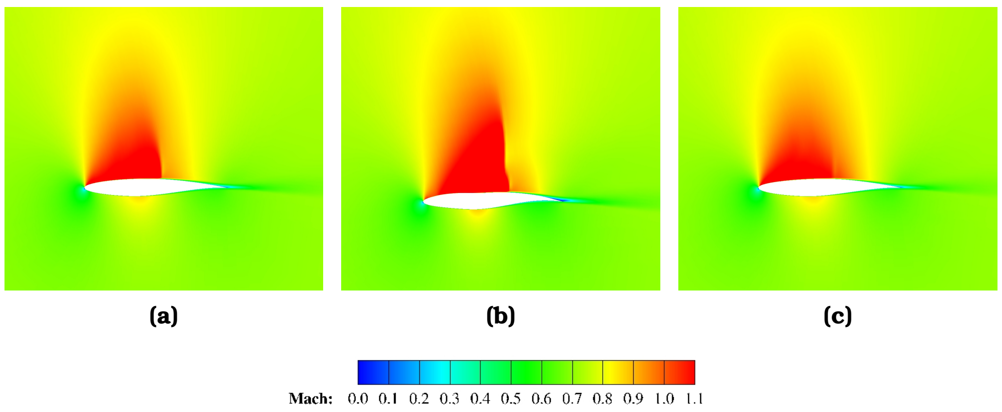


Figure 8.5: *Transonic Airfoil Optimization*: Mach number fields around the (a) baseline, (b) $\max c_L$ and (c) $\min c_D$ airfoils. It can be seen that the shock intensity has increased and the shock is moved further downstream on the $\max c_L$ airfoil. For the $\min c_D$ one, the shock intensity has been reduced.

8.2 Total Pressure Losses Minimization through a Linear 2D Compressor Cascade

In this application, the shape of a linear 2D compressor blade is optimized for $\min$ total pressure losses. A hybrid mesh is generated around the compressor blade, consisting of $\sim 8.8 \times 10^4$ nodes with $\sim 4.7 \times 10^4$ triangles and $\sim 2 \times 10^4$ quadrilaterals. Stretched quadrilateral elements are used to capture the boundary layer close to the blade, while triangular elements cover the remaining CFD domain. The distance of the first node off the wall leads to $y^+ < 1$ and, consequently, the Spalart-Allmaras model resolves the boundary layer down to the wall (the Low-Reynolds number variant of the model). The geometry of the CFD domain and the CFD mesh is presented in Fig. 8.6.

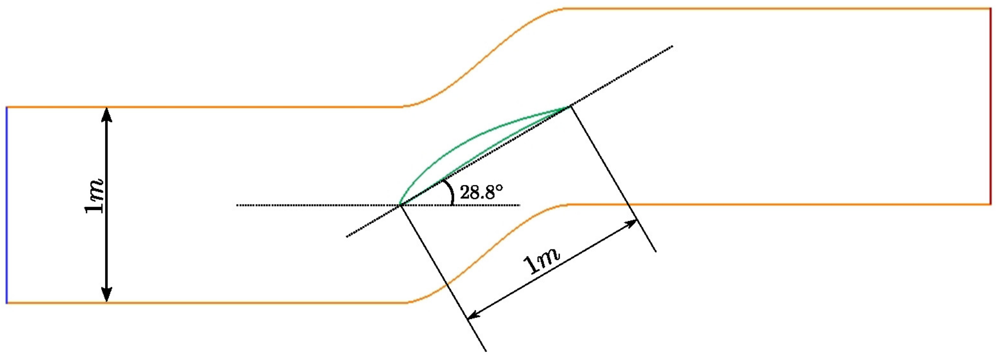
The flow conditions defining the operating point on which the optimized solution will be sought are summarized in Table 8.2.

The objective function is defined as

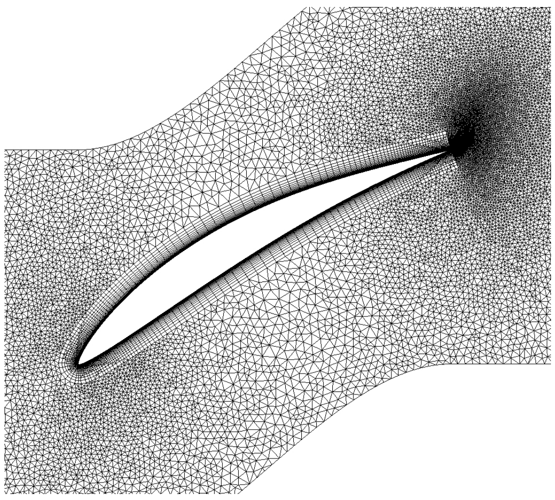
$$F = -\int_{S_I} p_t \rho v_k^A \mathsf{n}_k \mathrm{d}S - \int_{S_O} p_t \rho v_k^A \mathsf{n}_k \mathrm{d}S \qquad (8.2.1)$$

where the negative sign in front of the first term is used to account for the direction of the normal at the inlet boundary (pointing outside the flow domain).

The solution of the flow field takes on the average $\sim 2.5\,\mathrm{min}$ on a single NVIDIA Tesla K40 GPU. The adjoint field is solved on the average in $\sim 2\,\mathrm{min}$ on the same hardware.



(a)



(b)

Figure 8.6: *2D Linear Compressor Cascade Optimization*: (a) Geometry of the CFD domain. The linear pitch between two consecutive blades is 1 m, equal to the chord length. The blade is positioned with a stagger angle $\gamma = 28.8°$. (b) CFD mesh around the blade.

Quantity	**Symbol**	**Value**
Inlet Total Pressure	p_t^{in}	$1.151 \times 10^5\,\text{Pa}$
Inlet Total Temperature	T_t^{in}	$293.0\,\text{K}$
Outlet Static Pressure	p^{out}	$1.084 \times 10^5\,\text{Pa}$
Inlet Flow Angle	α^{in}	42.0°
Molecular Viscosity (constant)	μ	$1.716 \times 10^{-5}\,\text{kg}\,\text{m}^{-1}\,\text{s}^{-1}$
Inlet Viscosity Ratio	$(\mu_t/\mu)^{\text{in}}$	20.0

Table 8.2: *2D Linear Compressor Cascade Optimization*: Flow conditions used for the minimization of the total pressure losses of the flow through a linear compressor cascade. The flow conditions correspond to an outlet isentropic Mach number of $M_{2,\text{is}} = 0.295$.

The geometry of the blade and its surrounding CFD mesh are parameterized using a 13×7 polynomial NURBS control volume (all weights fixed to unity). Two layers of control points are kept fixed to ensure $\mathcal{C}^1$ continuity of the mesh around the blade with the surrounding mesh throughout the optimization. The NURBS control volume is shown in Fig. 8.7. It can be seen that the periodic boundaries are excluded from the control volume, even though they could be included using the method described in Section 6.3.2. The reason is that the pitch of the linear cascade is relatively large, providing enough room for displacement of the blade throughout the optimization process, without the need for periodic boundary modification.

The steepest-descent method is employed. The optimization process ran for 15 cycles and led to a reduction of $\sim 5\%$ in the objective function (Fig. 8.8).

Let the quantity f stand for the mass-weighted flow rate total pressure at a certain axial position, namely

$$f = \int\limits_{S_{\text{pos}}} p_t \rho v_k^A \mathsf{n}_k \mathrm{d}S \tag{8.2.2}$$

where S_{pos} stands for the surface obtained by transverse cuts of the flow field. Fig. 8.9 presents the distribution of $\frac{f - f^{\text{in}}}{f^{\text{in}}}$ along the axial direction for the baseline and the optimized geometry. In can be seen that the reduction on the objective function results mainly from the reduction in the total pressure losses generated by the wake of the blade.

8.3 Drag Minimization of a Transonic Wing

This application concerns the optimization of an isolated wing in transonic flow conditions for drag minimization. The ONERA M6 wing is used as the baseline

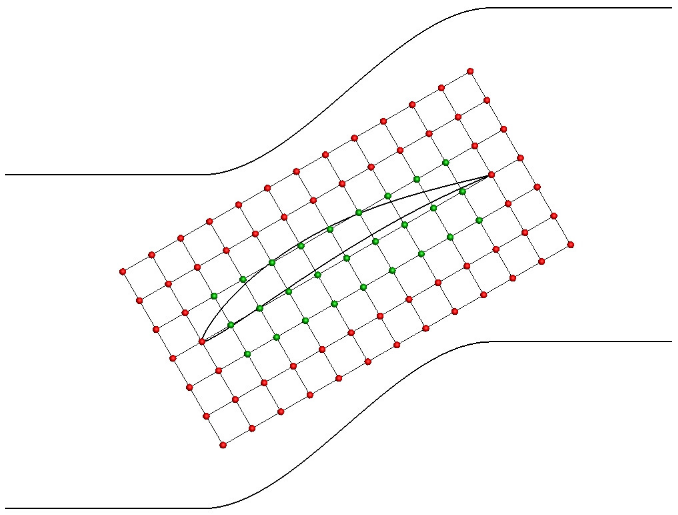


Figure 8.7: *2D Linear Compressor Cascade Optimization*: NURBS control volume defined around the blade. The control volume is used to deform the blade as well as the surrounding CFD mesh. Red control points are kept fixed to ensure $\mathcal{C}^1$ continuity with the non-deformed mesh outside the control volume. Green control points are allowed to move along the y-direction leading to a total of 27 design variables.

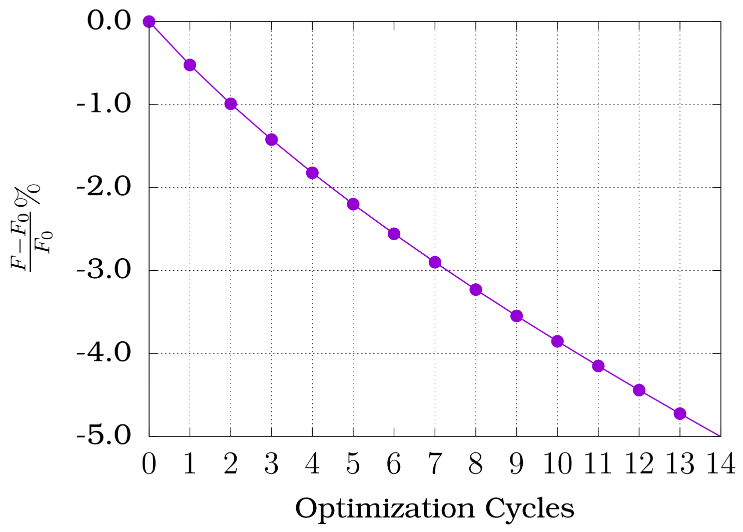
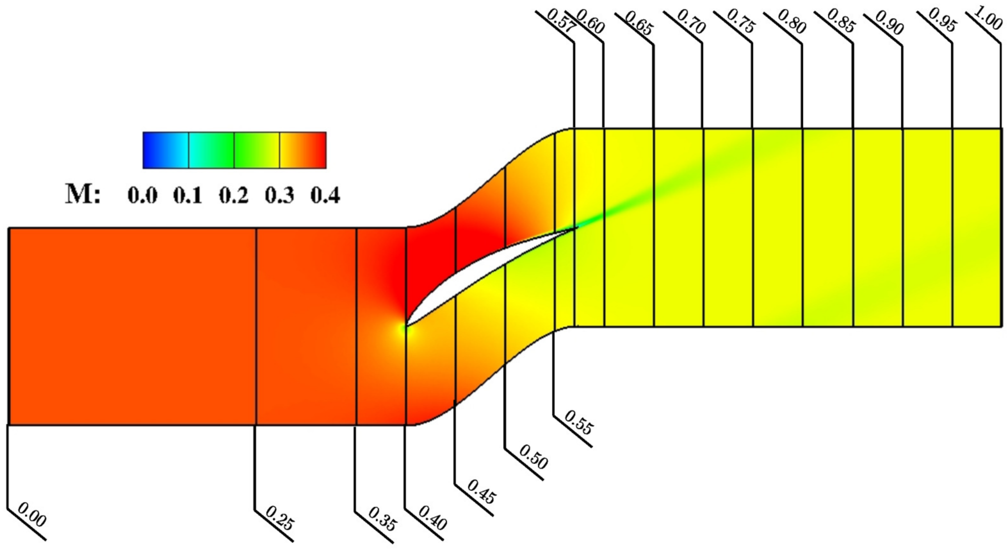
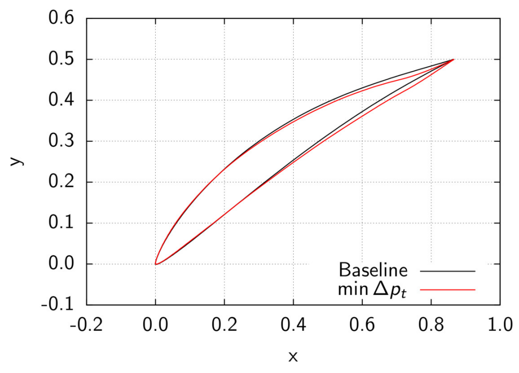
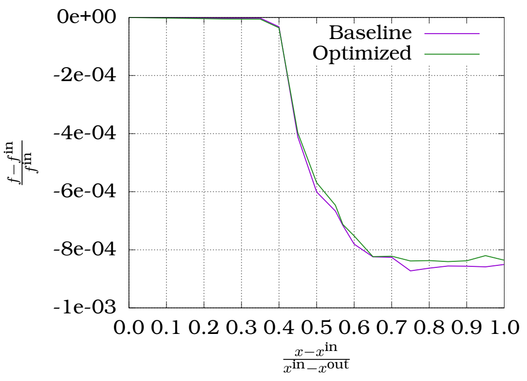


Figure 8.8: *2D Linear Compressor Cascade Optimization*: Optimization history. The optimization ran for 15 cycles and a reduction of $\sim 5\%$ of the objective function is achieved.



(a)

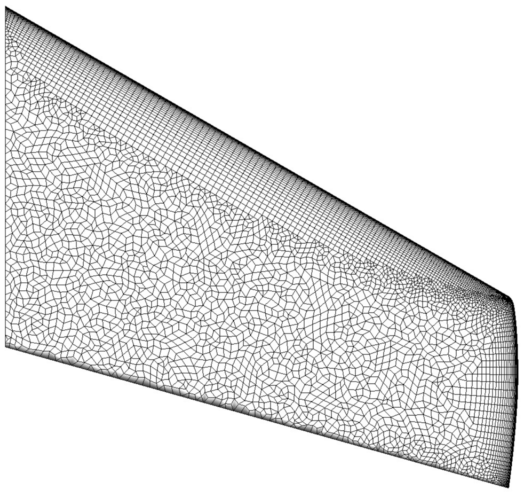




(b) (c)

Figure 8.9: *2D Linear Compressor Cascade Optimization*: (a) Mach number field through the cascade for the baseline geometry. The flow is subsonic. However, compressibility effects are not negligible. Several positions are chosen from inlet to outlet to compute the quantity f defined in Eq. 8.2.2 . (b) Comparison between the baseline and the optimized blade shape. The modification is more evident close to the trailing edge, in order to enhance diffusion control. (c) Total pressure losses integrated at several positions along the axial direction. The function f defined by Eq. 8.2.2 is used. The total pressure losses are presented as a percentage of f^{in}. It is seen that the main effect of the optimization appears in the wake region.

shape. A hybrid mesh of tetrahedra, pyramids, prisms and hexahedra is generated around the wing consisting of $\sim 1.3 \times 10^6$ nodes. The distance of the first nodes off the wall is chosen to allow for resolution of the turbulent boundary layer down to the wall $(y^+ < 1)$. The Spalart-Allamaras turbulence model is used and is also differentiated in the adjoint formulation (as in all adjoint-based optimizations performed within this thesis). The CFD mesh is shown in Fig. 8.10. The flow conditions are shown in Table 8.3.


(a)

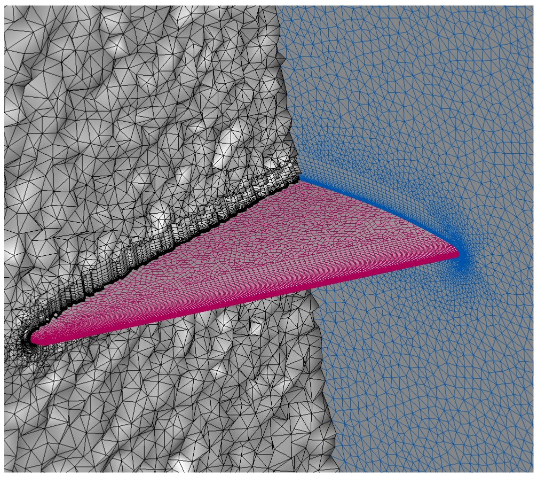

(b)

Figure 8.10: *Transonic Wing Optimization*: (a) Hybrid surface mesh on the wing surface. Stretched quadrilateral elements are used close to the LE and tip regions, to help properly capturing the surface curvature at these areas. (b) Hybrid volume mesh around the wing. The wing surface is shown in pink and the symmetry boundary in blue. Stretched hexahedra and prism elements are used close to the wing surface to obtain proper boundary layer resolution.

Quantity	**Symbol**	**Value**
Infinite Flow Pressure	p_∞	1.01325×10^5 Pa
Infinite Flow Density	ρ_∞	1.2 kg/m^3
Infinite Flow Velocity	V_∞	2.886373×10^2 m/ s
Infinite Flow Angle	α_∞	$3.06°$
Air Dynamic Viscosity	μ	1.90935×10^{-5} kg/m s
Infinite Viscosity Ratio	$(\mu_t/\mu)_\infty$	1.0

Table 8.3: *Transonic Wing Optimization*: Flow conditions.

Before proceeding to the optimization of the wing, a CFD analysis is performed and results are compared to experimental ones and other CFD results from the literature [163]. The isentropic Mach number field on the wing surface is shown in Fig. 8.11.

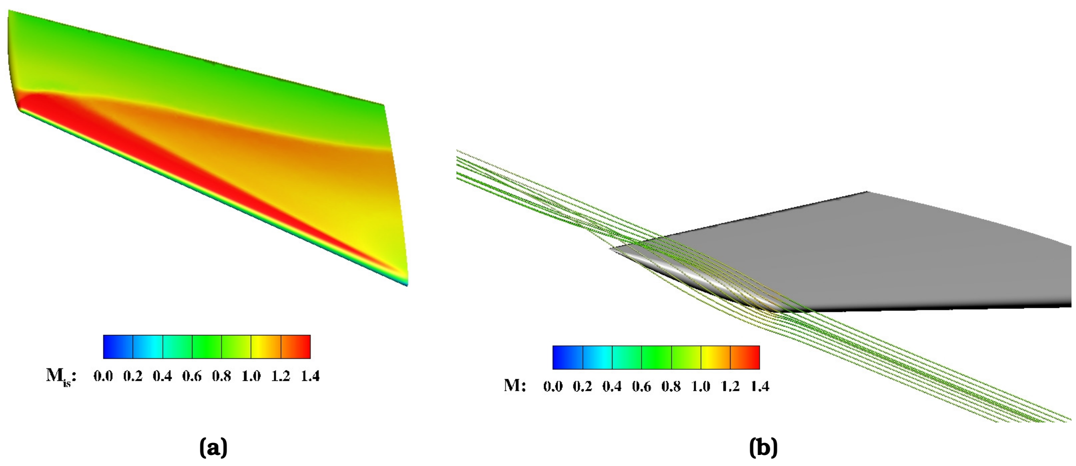

M_{is}: 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
M: 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
(a)
(b)

Figure 8.11: *Transonic Wing Optimization*: (a) Isentropic Mach number field drawn on the suction side of the baseline wing surface. The λ-shock pattern can be seen. (b) Streamlines close to the wing tip for the baseline geometry. The tip vortex structure is visible. Streamlines are coloured w.r.t. the Mach number value.

In Fig. 8.12, the pressure coefficient at different positions along the span of the wing is presented and compared to other CFD, as well as experimental results [163]. Each flow solution takes $\sim$ 38 min on a single NVIDIA Tesla K40 GPU.

The wing and its surrounding CFD mesh are parameterized by a $7 \times 9 \times 5$ NURBS control volume, with B-Spline basis functions of 2nd degree along each parametric direction. All weights of the control points are kept fixed and equal to unity throughout the optimization. The NURBS control volume built around the wing is shown in Fig. 8.13.

The FI continuous adjoint method is used to compute the sensitivity derivatives of the drag force w.r.t. each NURBS control point position. Each adjoint field solution takes $\sim$ 30 min on a single NVIDIA Tesla K40 GPU. Together with the solution of the flow and adjoint field each optimization cycle takes $\sim$ 70 min. It is clear that the time required for computing the sensitivity derivatives, adapting the CFD mesh and re-computing the necessary geometric data is very small (2 min) compared with the 70 min needed for each optimization cycle. A total of 15 optimization cycles ran. The drag is reduced by $\sim$ 30 drag counts (Fig. 8.14).

The change on the wing's shape is presented in Fig. 8.15 together with a comparison of the isentropic Mach number on the suction side, while Fig. 8.16 presents the pressure coefficients of the optimized wing in comparison with the ones of the baseline shape. It is seen that the drag is reduced mainly by decreasing the intensity of the shock wave on the suction side of the wing. In addition, the

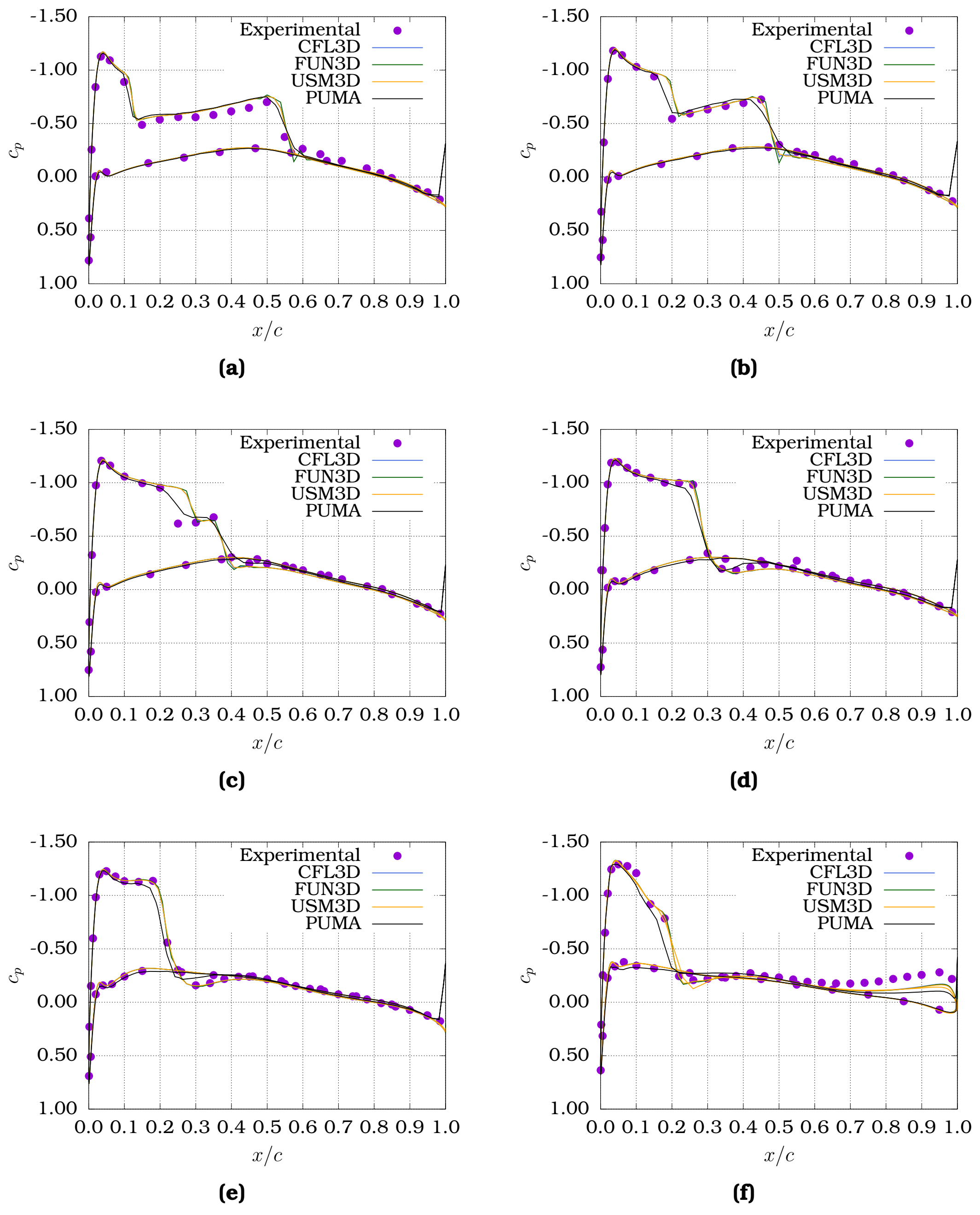


Figure 8.12: *Transonic Wing Optimization*: Comparison of the pressure coefficient distributions at (a) 44% (b) 65% (c) 80% (d) 90% (e) 96% (f) 99% of the wing span, for the baseline shape. The results obtained by PUMA are compared to results from three other CFD codes, namely CFL3D[109], FUN3D[110] and USM3D[111], as well as experimental results from wind tunnel testing [163].

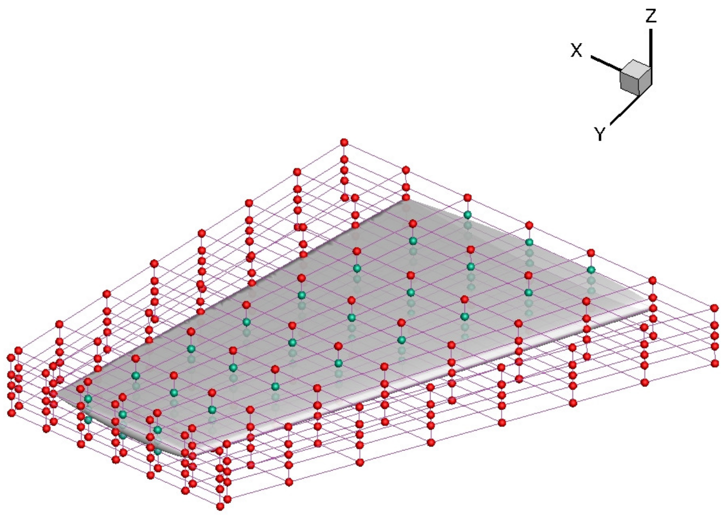


Figure 8.13: *Transonic Wing Optimization*: NURBS control volume built around the wing for the purpose of controlling its shape throughout the optimization and deforming the surrounding CFD mesh accordingly. Red control points are kept fixed to ensure continuity with the non-deformed outer mesh, while green control points are allowed to move along the z-direction. The wing's spanwise chord distribution is kept fixed.

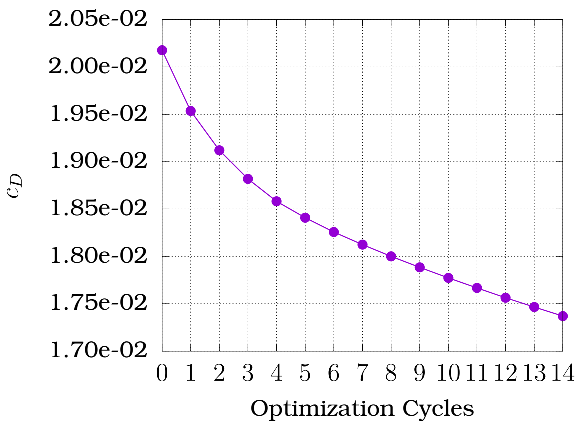
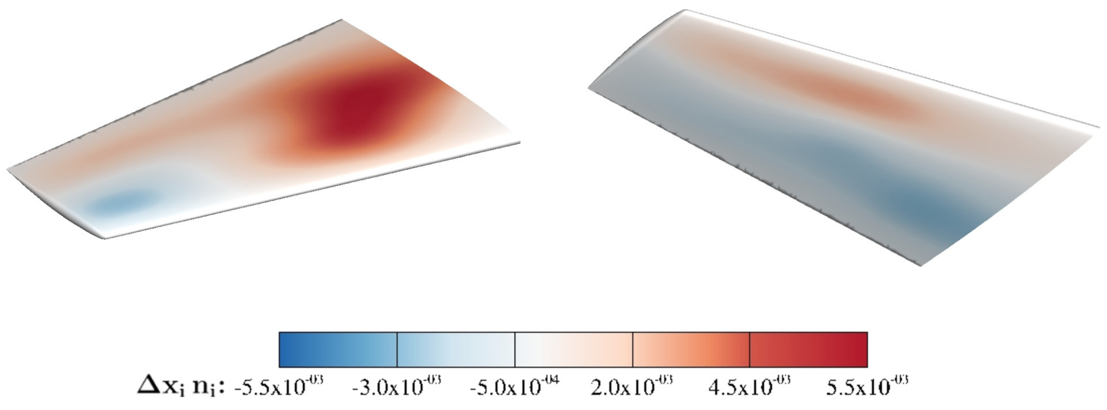
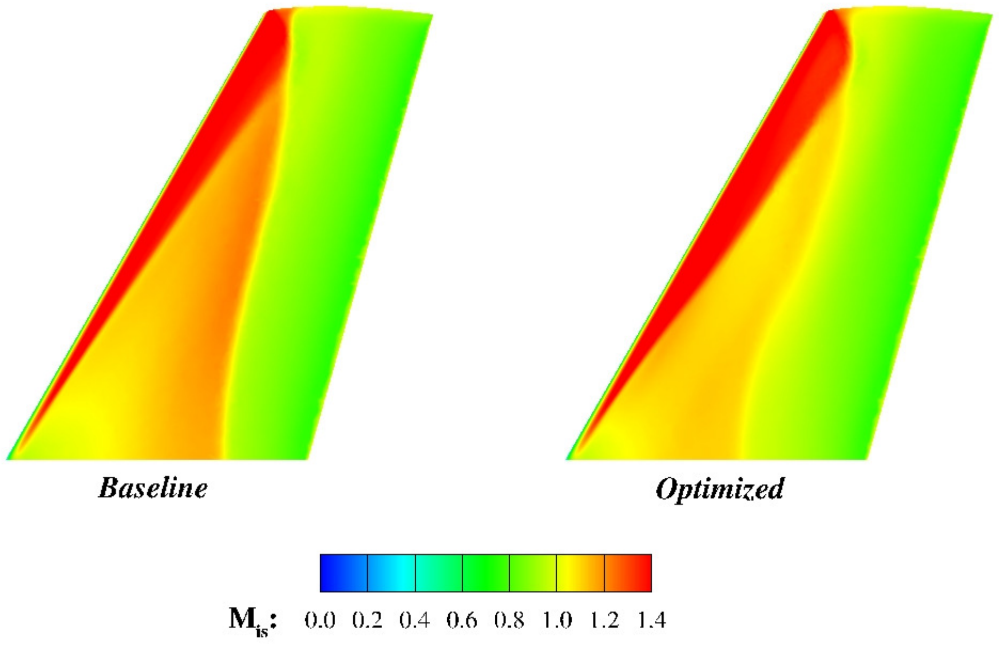


Figure 8.14: *Transonic Wing Optimization*: Optimization history. A reduction of ~ 30 drag counts is obtained after 15 optimization cycles. The overall wall clock time is $17\,\mathrm{h}$ and $30\,\mathrm{min}$ on a single NVIDIA Tesla K40 GPU. Even though a stationary point has not been reached yet, the optimization was stopped due to a limitation set on the computational budget.

shape of the suction side close to the tip is modified in an effort to control the tip vortex and reduce the drag induced by it.

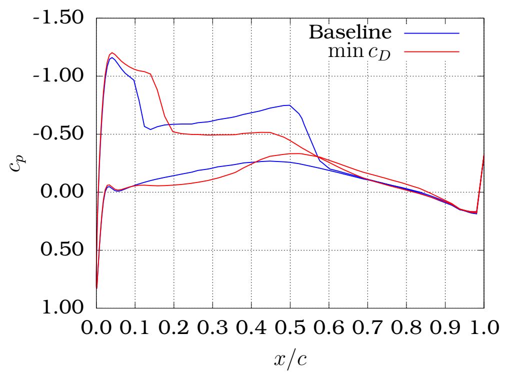

$\Delta x_i\, n_i$: -5.5x10^{-03} -3.0x10^{-03} -5.0x10^{-04} 2.0x10^{-03} 4.5x10^{-03} 5.5x10^{-03}

(a)

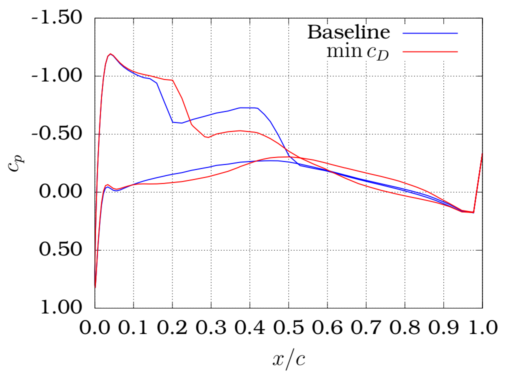


(b)

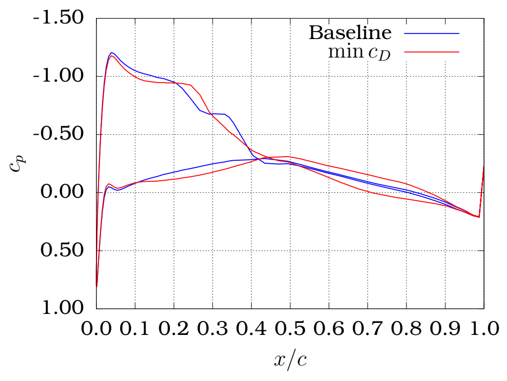
Figure 8.15: *Transonic Wing Optimization*: (a) Normal displacement of the suction (left) and pressure side surface of the optimized wing. Positive normal displacement corresponds to the geometry being pushed towards the solid, while negative means that the wing is inflated towards the fluid region. (b) Comparison of the isentropic Mach number field on the suction side of the wing between the baseline and optimized shape. It can be seen that the shock intensity is reduced on the optimized wing.



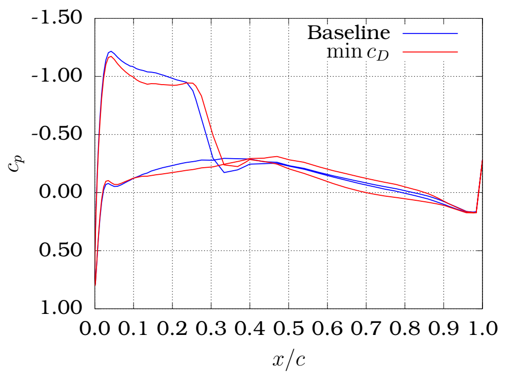
(a)



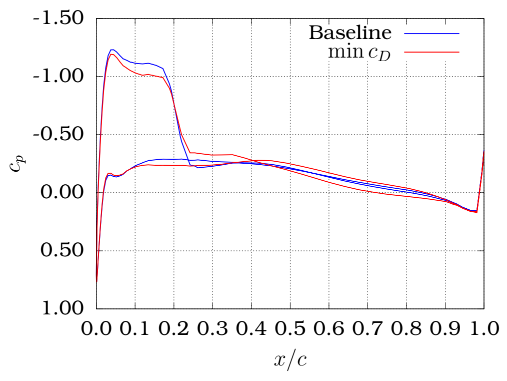
(b)



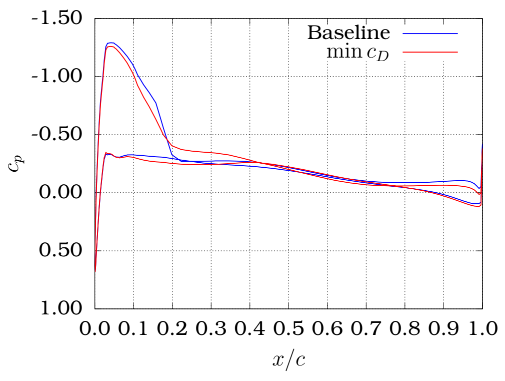
(c)



(d)



(e)

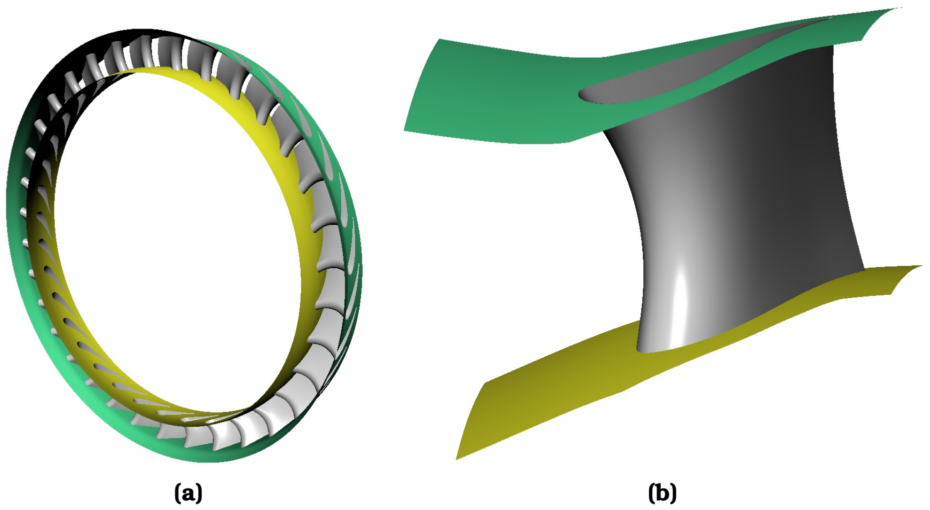


(f)

Figure 8.16: *Transonic Wing Optimization*: Comparison of the pressure coefficient distributions at (a) 44% (b) 65% (c) 80% (d) 90% (e) 96% (f) 99% of the optimized wing span with the baseline one. The shock intensity on the suction side of the wing is greatly reduced in the inner sections of the wing leading to lower drag coefficient.

8.4 Total Pressure Losses Minimization of a Turbine Nozzle Guide Vane

This application concerns the optimization of turbine nozzle guide vane (NGV) row for min. total pressure losses of the flow between inlet and outlet. The baseline turbine geometry is provided by Rolls-Royce Deutschland. The row consists of 34 blades and is presented in Fig. 8.17.



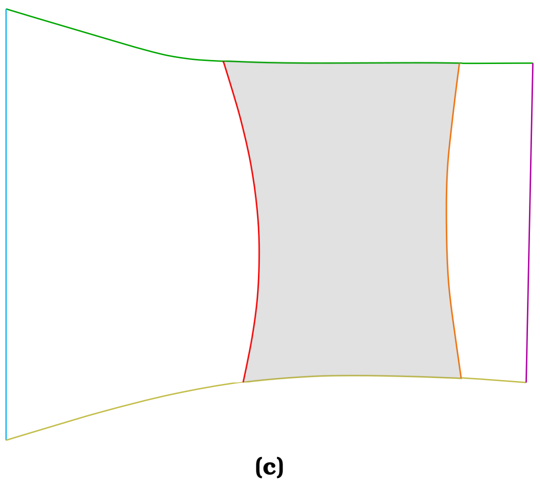


Figure 8.17: *Turbine NGV Optimization*: (a) Full annulus geometry of the turbine NGV row, consisting of 34 blades. The hub surface is shown in yellow and the shroud one in green. (b) Close-up view of the geometry on a single blade. The blade is shrouded and no tip gap exists between the blade and the hub. (c) Meridional projection of the turbine NGV row geometry. The hub and shroud generatrices are shown in yellow and green color, respectively. The blue line is the inlet to and the purple one the outlet from the row. Finally, the projection of the leading edge on the meridional plane is shown in red, while the one of the trailing edge in orange.

A multi-block structured mesh is generated around the blade, consisting of $\sim 1.5 \times 10^6$ nodes. Even though the mesh is multi-block structured, it is treated by PUMA as unstructured consisting solely of hexahedral elements. The Spalart-Allmaras model is used to integrate the governing equations down to the wall

$(y^+ < 1)$. A perspective view of the boundary mesh is presented in Fig. 8.18.

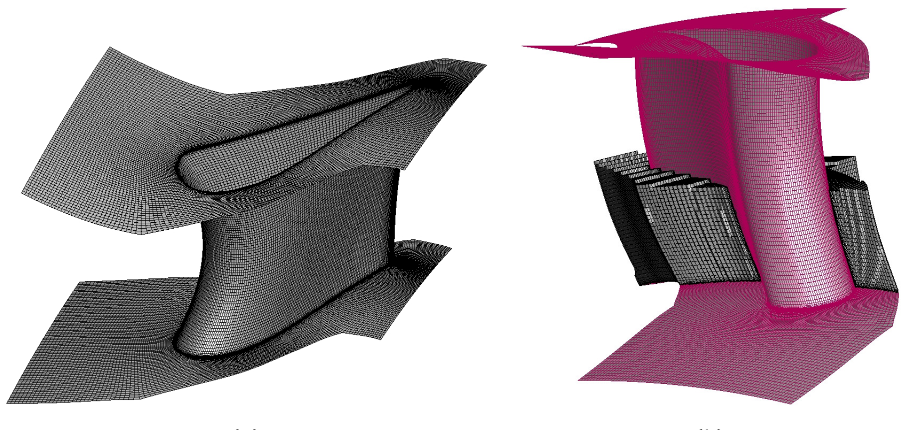


Figure 8.18: *Turbine NGV Optimization*: (a) Surface mesh used for the analysis and optimization of the turbine NGV. The mesh is shown only on the solid wall surfaces. (b) Volume mesh cut at a certain radial and axial position. The surface mesh on the blade is shown in pink color.

Before proceeding with the optimization, a flow analysis is performed with PUMA and the results are compared with those from a computation performed by the provider of the test case with its in-house CFD software. The flow conditions are shown in Table 8.4. Some of the conditions are specified as inlet or outlet radial profiles, as presented in Fig. 8.19.

Quantity	Symbol	Value
Inlet Total Temperature	T_t^{in}	565.0 K
Molecular Viscosity(constant)	μ	1.785×10^{-5} kg/m s
Specific Heat Ratio	γ	1.379 047

Table 8.4: *Turbine NGV Optimization*: Flow conditions. The data presented in this table are supplemented with the radial profiles provided in Fig. 8.19.

Comparison of the radial distribution of the outlet velocity components predicted by PUMA with those predicted by the in-house CFD software of the test case provider are presented in Fig. 8.20. In the same figure, the radial distributions of the loss coefficient are also compared.

The CFD run takes $\sim$ 1 h and 45 min on a single NVIDIA Tesla K40 GPU.

In order to proceed to the optimization of the NGV, a NURBS control volume is defined around the blade on the PCS. The control volume consists of $8 \times 6 \times 3$ control points. The basis functions are of 3rd degree in the u-direction (roughly

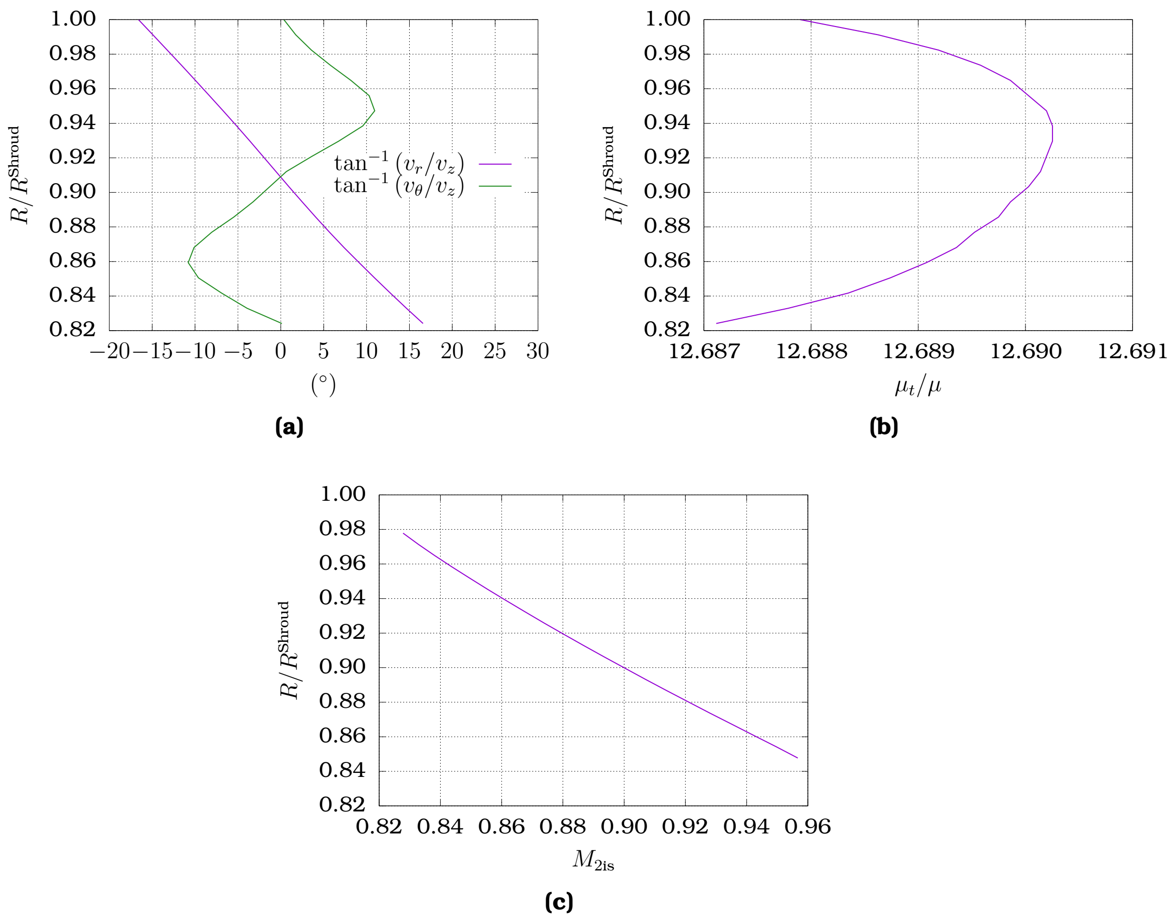


Figure 8.19: *Turbine NGV Optimization*: Flow conditions which supplement the ones shown in Table 8.4. (a) Radial profile of the inlet flow angles, where v_r and v_θ stand for the circumferential and radial velocity components. (b) Radial profile of the inlet viscosity ratio. (c) Radial profile of the isentropic Mach number at the stator outlet.

associated with the streamwise direction) and of 2nd degree in the other two parametric directions. The control volume is presented in Fig. 8.21. The blade has been axially duplicated as many times as needed to ensure periodicity. The control volume is presented both on the PCS, as well as on 3D Cartesian space. It must be noted, however, that even though the coordinates of the control points in the 3D Cartesian space are exact, the lines of the control polygon are not, in the sense that a straight line defined on the PCS does not map to a straight line in the 3D Cartesian space.

The objective function to be minimized is the mass-averaged total pressure losses between inlet and outlet. Mathematically, these are expressed as

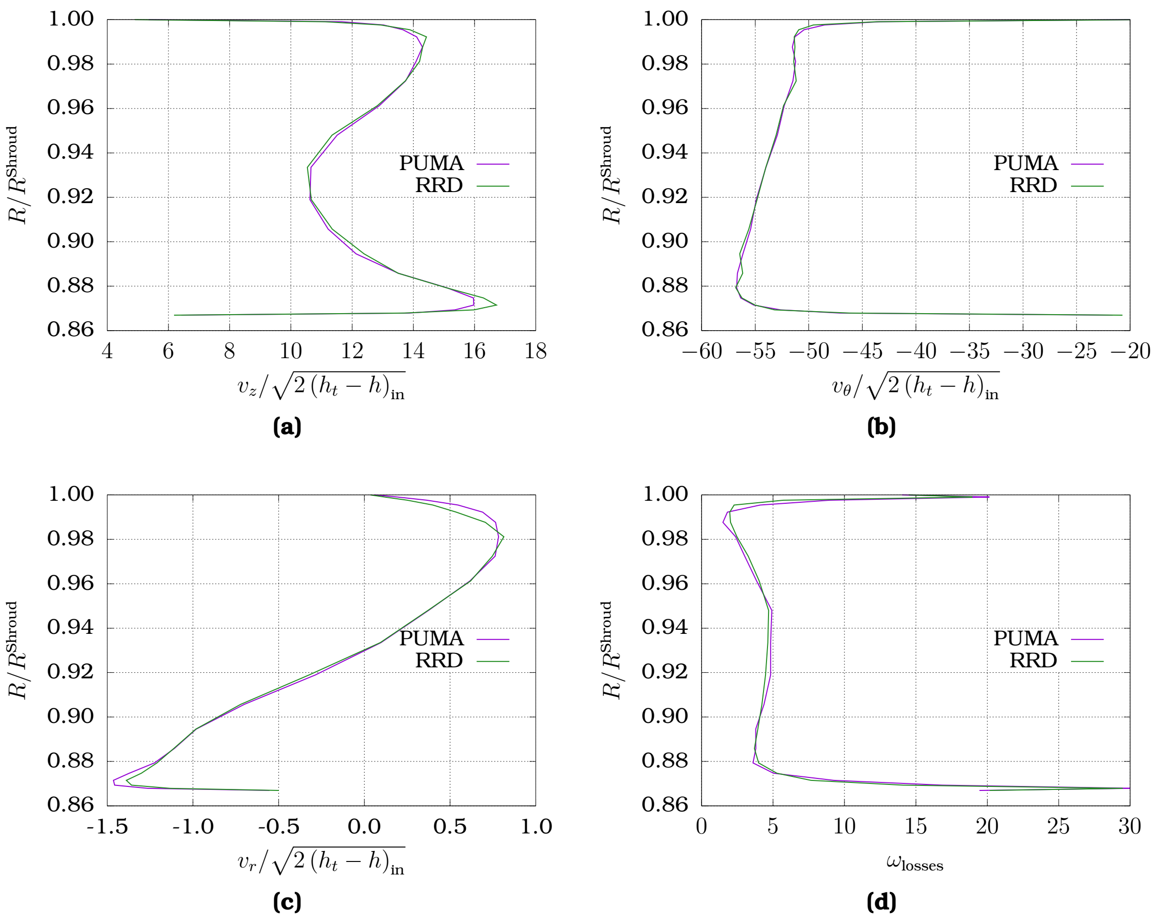


Figure 8.20: *Turbine NGV Optimization*: Comparison of the outlet flow characteristics predicted by PUMA, with those predicted by the CFD software of the test case provider. (a) Normalized axial velocity component, (b) normalized peripheral velocity component, (c) normalized radial velocity component and (d) normalized total pressure losses computed as $\omega_{\text{losses}} = \frac{p_t^{\text{in}}-p_t}{p_t^{\text{in}}-p^{\text{in}}}$. The results compare very well with each other.

$$F = f^{S_I} - f^{S_O}$$
$$\text{where } f^S = \frac{\int\limits_S p_t \rho v_i^A \mathsf{n}_i \mathrm{d}S}{\int\limits_S \rho v_i^A \mathsf{n}_i \mathrm{d}S} \tag{8.4.1}$$

The optimization runs for 10 cycles and an $\sim 20\%$ reduction in the objective function is achieved. The wall clock time of the overall optimization process is $\sim$

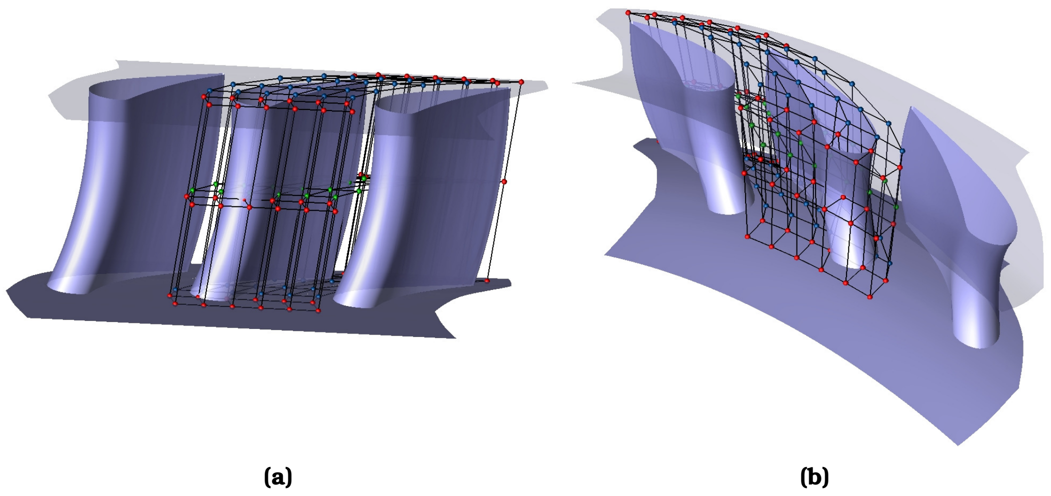
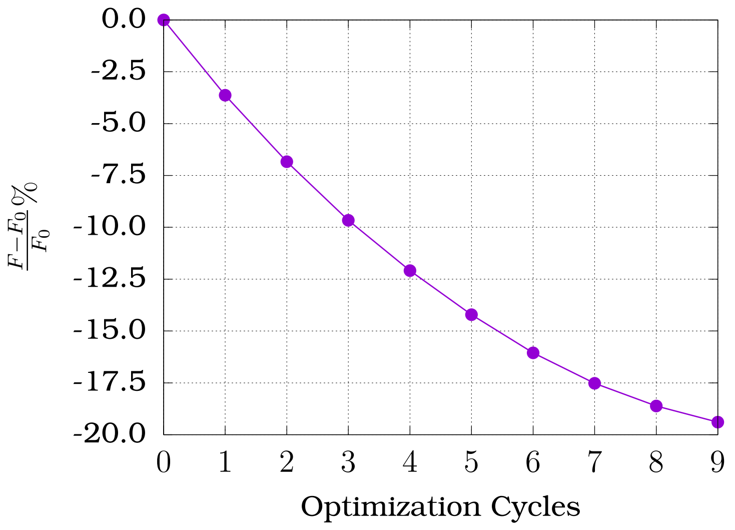


Figure 8.21: *Turbine NGV Optimization*: (a) NURBS control volume defined around the blade in PCS. Green control points are allowed to move in all three directions (streamwise, circumferential and spanwise). Red points are kept fixed to ensure continuity of the deforming part of the mesh with the non-deformed one. Blue control points are allowed to move only along the streamwise (ξ) and pitchwise (θ) directions. Periodic movement is ensured by the method described in 6.3.2. (b) NURBS control volume mapped in the Cartesian space. It must be noted that the lines connecting the control points are not actual lines of the control grid, since these are only defined in PCS. Keeping the spanwise (η) coordinate of the blue control points fixed ensures that the CFD mesh at the hub and shroud will slide along the corresponding surfaces.

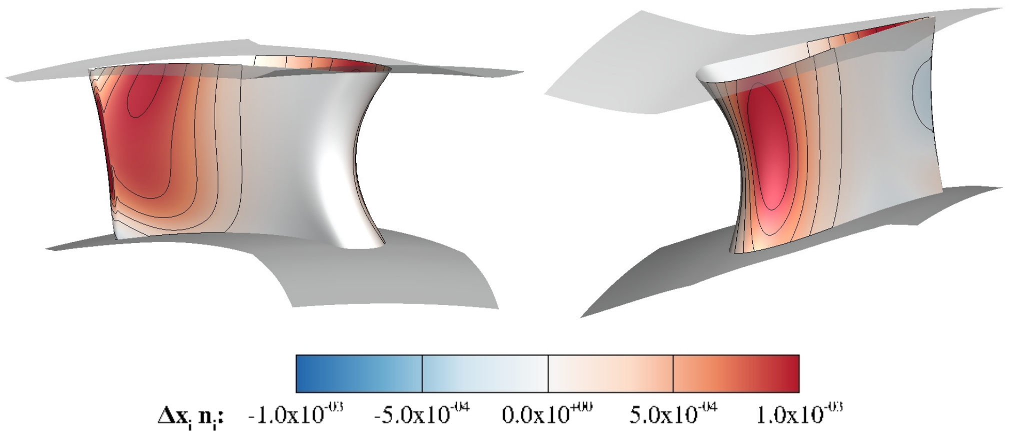
34 h on a single NVIDIA Tesla K40 GPU. The solution of the primal field consumes 51.2% of the overall optimization wall clock time. Another 46.3% is consumed by the adjoint solver and, finally, 2.5% of the optimization wall clock time is consumed by mesh morphing, computation of sensitivities and mesh related operations. The objective function convergence throughout the optimization is shown in Fig. 8.22 together with the field of normal displacement on the wall boundaries. It can be seen that the normal displacement on the hub and shroud is zero, due to the constrained displacement of the blade airfoil and surrounding mesh along these surfaces.

Airfoils of the blade at different spanwise positions are presented in Fig. 8.23 where the shape of the airfoils is compared to the baseline one. The shape modification is more pronounced at midspan, compared to the hub and shroud airfoils.

Fig. 8.24 presents the field of the isentropic Mach number on the blade suction side for the baseline and optimized blades. It can be seen that a small shock starts to appear on the suction side of the baseline shape, while for the optimized one this shock is greatly diffused.



(a)


$\Delta x_i n_i$: -1.0x10^{-03} -5.0x10^{-04} 0.0x10^{+00} 5.0x10^{-04} 1.0x10^{-03}

(b)

Figure 8.22: *Turbine NGV Optimization*: (a) Objective function history throughout the optimization. After 10 optimization cycles the objective function (massflow averaged total pressure losses) has been reduced by $\sim 20\%$ of the initial value. (b) Normal displacement field on the wall boundaries for the optimized NGV shape. The major modification takes place at the suction side (close to the position of the maximum camber/maximum flow acceleration) in an attempt to reduce the total pressure losses induced by the presence of a shock. Red color is associated with displacement of the blade's surface towards the solid area. In addition, the effect of sliding along the hub and shroud surfaces (zero normal displacement), as a result of the TMVNURBS parameterization, can be seen.

The effect of minimizing the mass-averaged total pressure losses on the loss coefficient radial distribution at the NGV outlet is shown in Fig. 8.25. The same

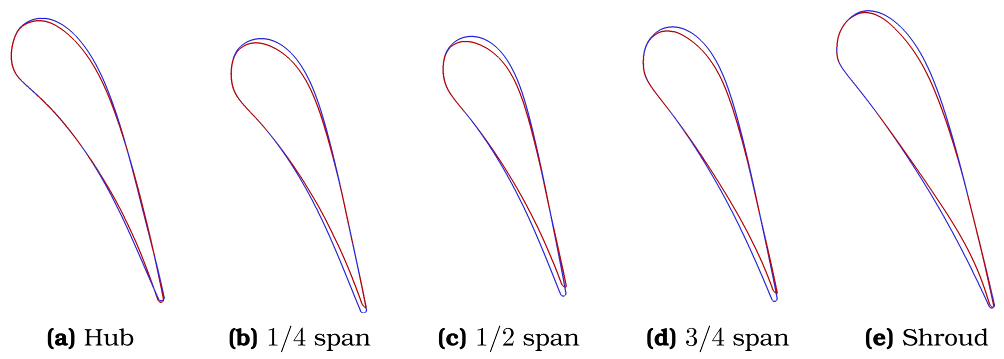


Figure 8.23: *Turbine NGV Optimization*: Comparison of the optimized (red) and baseline (blue) NGV airfoils at different positions along the blade span. It can be seen that the modification is more pronounced at the sections away from hub and shroud. Also, the trailing edge appears to guide the flow downstream with smaller turning, mainly due to modifications in the pressure side shape. In addition, the blade also appears shortened in the 1/4, mid-span and 3/4 sections, which leads to smaller viscous losses.

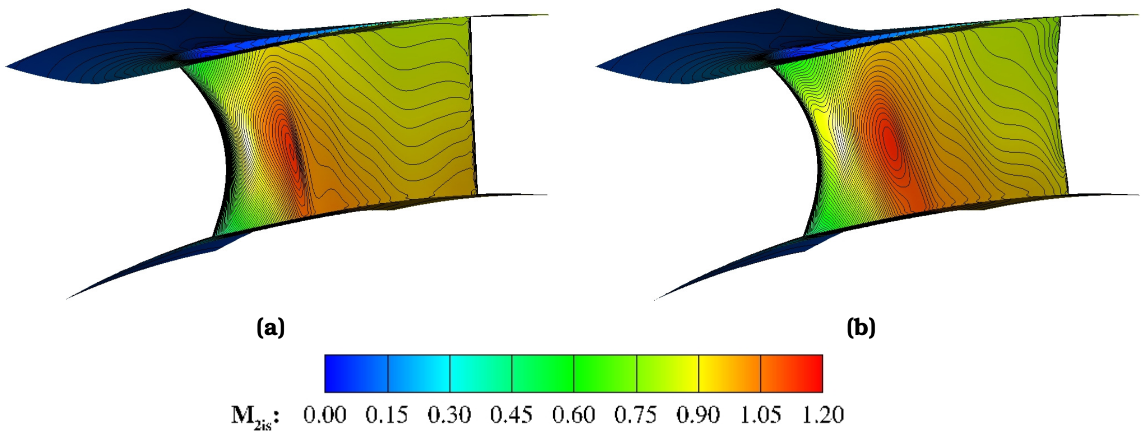
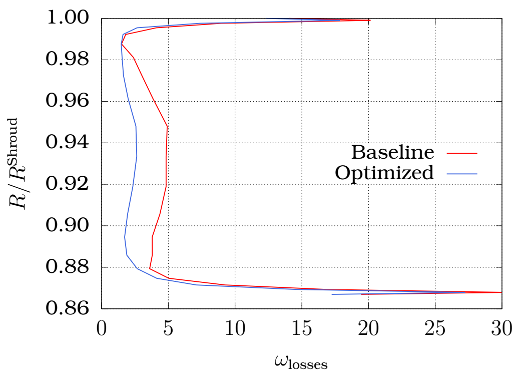
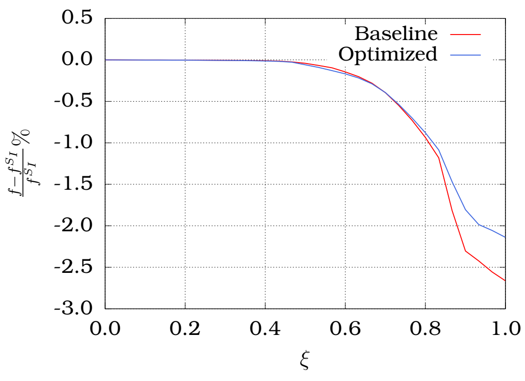


Figure 8.24: *Turbine NGV Optimization*: Isentropic Mach number field on the suction side surface of (a) the baseline and (b) the optimized blade. The weak shock that appears on the baseline blade is diffused on the optimized one.

figure shows the distribution of the mass-averaged total pressure (f of Eq. 8.4.1) along the flow path. It can be seen that the greatest reduction in generated losses is obtained by increasing the mass-averaged total pressure right after the position of the weak shock.





(a) (b)

Figure 8.25: *Turbine NGV Optimization*: (a) Radial distribution of ω_{losses} coefficient at the outlet of the baseline and optimized NGV. It can be seen that the losses are significantly lower for the optimized geometry, especially around the midspan position. (b) Streamwise distribution of mass-averaged total pressure losses. In the optimized geometry, the mass-averaged total pressure losses are significantly lower around the weak shock position and remain lower up to the outlet.

Chapter 9

Closure

9.1 Summary-Conclusions

The target of this thesis was the development of an integrated workflow for aerodynamic shape optimization, versatile and efficient enough to be incorporated in the design of real-life aerodynamic components. To this end, acceleration of the CFD using GPUs, development of the continuous adjoint method for both compressible and incompressible flows and development of compact and efficient shape parameterization methods have been presented.

More specifically, the development of the compressible and incompressible flow (U)RANS equations solver on GPUs proved that GPU-enabled CFD codes are advantageous compared to CPU ones, in terms of computational cost and hardware cost efficiency (lower purchase cost and memory consumption for the same computational performance). Specifically, the GPU-enabled code proved to be $\sim 40\times$ faster than the corresponding CPU one, although this figure may vary depending on the CPU and GPU hardware specifications involved in the comparison. Even though, developing software running on a GPU cluster is stil more complicated than developing one for a CPU cluster, the ground laid on GPU programming of the PUMA code in [182] and exploited/upgraded in this thesis, proved to be solid enough to allow for new, more complex, features to be fast and reliably implemented on the GPU. This, in turn, eliminates the additional cost of CPU to GPU code translation and restructuring for the development of new methods or ideas in the field of CFD. Additionally, the faster execution of the GPU implementation of the PUMA code facilitates parametric studies and validation/verification of new features.

The artificial compressibility (AC) method has been developed for 3D incompressible flows in rotating frames of reference using the multiple reference frame (MRF) approach. The method proved to be accurate, robust and reliable in a series of flow simulations, from the field of wind turbines to that of hydraulic turbomachinery. Even for low-speed compressible fluid flows ($M < 0.3$) the AC method is preferable, compared to the use of the compressible solver variant with Low-Mach number preconditioning. The reason is that the AC method for isothermal flows employs one equation less (no energy equation solved) and, consequently, consumes less memory and computational resources. Compared to other approaches for solving the incompressible flow equations such as the pressure-correction method, the AC method is easier to maintain in combination with a solver for hyperbolic PDEs using time–marching techniques (such as a compressible flow solver), since the discretization and numerical schemes are the same between the AC method and the latter. As an example, the compressible and incompressible flow (based on AC) solver of PUMA share the majority of source code files.

Concerning the optimization methods, the thesis targeted mostly at developing and extending the applicability of the continuous adjoint approach for use in a wide range of aerodynamic shape optimization problems. Following the works presented in [201, 143, 96] the use of adjoint to turbulence models has been extended to compressible flows. More specifically, the Spalart-Allmaras model has been developed for compressible and incompressible flows. Earlier works in the PCOpt/LTT on the continuous adjoint method for compressible flows [136, 6] neglected the effects of variations on turbulent quantities. Incorporating the adjoint turbulence model by also taking into account variations of the distance from the wall, a complete and more accurate adjoint formulation for compressible flows is now available. The differentiation of variable bulk viscosity laws (such as the Sutherland law) aimed also in extending the applicability of the continuous adjoint method in flows where the constant bulk viscosity assumption is inappropriate (e.g. flows through high-pressure turbine blade rows e.t.c.).

Both the Field Integral (FI) and Surface Integral (SI) adjoint approaches have been developed, resulting in different expressions for the sensitivity derivatives. The SI approach results in expressions containing exclusively surface integrals, while the FI one results in expressions with volume integrals as well. When the severed-SI approach is used, the accuracy of the computed gradients is potentially different between the two formulations, depending mainly on the flow physics and discretization accuracy. The development of both approaches has been carried out in a way that facilitates the incorporation of different objective functions and/or flow related constraints, without the need of reformulating the adjoint solver from scratch. The use of object-oriented programming proved to be of paramount importance towards this direction.

The idea of the AC method has been extended to the continuous adjoint equations for incompressible flow problems, resulting in improved convergence characteristics of the adjoint solver without loss in the computed gradient accuracy. The availability of the adjoint AC method, combined with the adjoint for Low-Mach Number Preconditioning and a fully differentiated compressible solver resulted in an adjoint-based optimization tool for all flow regimes. In addition, the GPU implementation of the adjoint solver proved to bear great computational savings, even greater than the transfer of the flow solver from CPUs to GPUs. This allowed gradient-based optimization techniques to be applied in industrial scale applications with reduced optimization turnaround times.

A geometric modeler for the design and shape parameterization of turbomachinery components has been developed, namely the GMTurbo software. This software has proved valuable in designing and optimizing a wide range of turbomachinery shapes ranging from axial to radial and mixed flow ones. The same software helped in maintaining CAD compatibility throughout the optimization, providing the optimization results in an appropriate format for subsequent stages of a product's development and production chain. Furthermore, the compactness of the parameterization resulting from the GMTurbo, results in a representation with directly controllable number of design variables, rendering EA-based optimization a viable option for the design of rotating machinery blade rows. Additionally, the reduced evaluation cost due to the use of single or multiple GPUs extended the applicability of EAs as a competitive tool for industrial scale aerodynamic shape design and optimization.

The volumetric NURBS approach for parameterizing aerodynamic shapes and deforming CFD meshes built around them has been presented. This enabled the optimization of geometries whose CAD representation is not available from the start and are given only in a discrete manner (CFD mesh or collection of points). At the same time, the same parameterization technique allowed the inexpensive use (consuming approximately 3% of the optimization total wall-clock time) of the otherwise increased cost FI continuous adjoint approach in optimization loops. With such an approach mesh morphing occurs simultaneously with the shape modification, further reducing the optimization run time. Extension of the volumetric NURBS approach for turbomachinery geometries facilitated the optimization setup for blade rows with unavailable initial CAD representation.

Finally, the ensemble of the aforementioned developments led to an integrated optimization loop with improved robustness and a wide range of applicability for computationally expensive CFD based optimization problems.

9.2 Novel Contributions

The novel contributions of this PhD thesis are summarized below:

- The artificial compressibility (AC) method for incompressible flows solved in a rotating frame of reference with the absolute velocity components as unknown variables (MRF approach) is presented for the first time in this thesis. In addition, the GPU implementation of this solver may be considered novel.

- The extension of the AC method to the continuous adjoint equations for incompressible flows is novel. This is not to be confused with the approach proposed in [102], where the adjoint equations are derived by the flow equations after the artificial compressibility has been applied on them. The latter can be considered as an alternative approach.

- The FI continuous adjoint formulation for turbulent compressible fluid flows with variable bulk viscosity is presented for the first time.

- To the authors knowledge, the PUMA code is the only GPU implementation of the continuous adjoint method.

- The extension of the volumetric NURBS technique for turbomachinery applications. Especially, the ability of the method to undertake the mesh deformation by maintaining axisymmetry and periodicity is novel.

Publications and Conference Presentations

- D. Kapsoulis, K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. 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, 2018.

- D. Kapsoulis, K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. PCA-Assisted Hybrid Algorithm Combining EAs and Adjoint Methods for CFD-based Optimization. *Applied Soft Computing*, 73:520-529, 2018.

- K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. Design- Optimization of a Compressor Blading on a GPU Cluster. In EUROGEN 2013, *10th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Las Palmas de Gran Canaria, Spain, October 7 - October 9, 2013. (Chapter in book)

- X. Trompoukis, K. Tsiakas, M. Ghavami Nejad, V. Asouti, and K. Giannakoglou. The Continuous Adjoint Method on Graphics Processing Units for Compressible Flows. In OPT-i, *International Conference on Engineering and Applied Sciences Optimization*, Kos Island, Greece, June 4–June 6, 2014.

- K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. Shape Optimization Using the Continuous Adjoint Method and Volumetric NURBS on a Many GPU System. In 8th GRACM *International Congress on Computational Mechanics*, Volos, Greece, July 12–15, 2015

- K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. Shape Optimization of Wind Turbine Blades using the Continuous Adjoint Method and Volumetric NURBS on a GPU Cluster. In *EUROGEN 2015, 11th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Glasgow, UK, September 14–September16, 2015. (Chapter in book)

- K. Tsiakas, F. Gagliardi, X. Trompoukis, and K. Giannakoglou. Shape Optimization of Turbomachinery Rows using a Parametric Blade Modeller and the Continuous Adjoint Method running on GPUS. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, Crete island, Greece, June 5–June 10, 2016.

- D. Kapsoulis, K. Tsiakas, V. Asouti, and K. Giannakoglou. The use of Kernel PCA in Evolutionary Optimization for Computationally Demanding Engineering Applications. In *2016 IEEE Symposium Series on Computational In- telligence (IEEE SSCI 2016)*, Athens, Greece, December 6–December 9, 2016.

- F. Gagliardi, K. Tsiakas, and K. Giannakoglou. A Two-Step Mesh Adaptation Tool Based on RBF with Application to Turbomachinery Optimization Loops. In *EUROGEN 2017, International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Madrid, Spain, September 13–September 15, 2017. (Chapter in book)

9.3 Future Work - Suggestions

This PhD thesis was based on previous works by the PCOpt/LTT and enriched them with new and novel developments. Hopefully, the work presented within this thesis will be extended and further developed in future works by PCOpt/LTT. Some ongoing developments and suggestions concerning future work are exposed in the following list.

- The continuous adjoint method developed for both compressible and incompressible flows may be extended to multi-disciplinary optimization problems (MDO), namely aerostructural, aerothermal and aeroacoustic optimization. Some steps concerning the aerostructural optimization of wind turbine blades have been presented by the author in the scope of the integrated research project WIND-FSI, funded by the General Secretariat of Research and Technology [53]. PCOpt/LTT is actively developing the coupling of the adjoint solver of PUMA with commercial adjoint structural solvers for the aerostructural optimization of aircrafts. Concerning the aerothermal optimization, there is already a PhD thesis under development in PCOpt/LTT [57] on the development of the adjoint method for Conjugate Heat Transfer problems. This work is based on the open source software OpenFOAM, but transfer of recent developments on the PUMA software are already in progress. Finally, concerning aeroacoustic optimization, Computational Aeroacoustics (CAA) tools are developed in PUMA, together with their adjoint counterparts, in the framework of the PhD thesis [127].

- The continuous adjoint in PUMA can be extended for unsteady flows. An interesting field of research is related on the cost reduction of storing/recomputing intermediate flow solutions, needed to march the adjoint solution backwards in time and, especially, the GPU implementation of such cost reduction techniques. Related work is in progress in two PhD theses [190, 159].

- Concerning the turbomachinery modeler GMTurbo, developed in this PhD thesis, its differentiation and coupling with the continuous adjoint method is done within another PhD thesis [51] running in PCOpt/LTT. GMTurbo is, also, undergoing constant development with the inclusion of more advanced technological features, common in modern turbomachinery blades, such as cooling holes, cooling slots, squealers, non-axisymmetric hub and shroud and many more. The GMTurbo software can be extended to account for manufacturing imperfections and coupled with Uncertainty Quantification techniques provide a framework for robust design optimization.

- The method of volumetric NURBS may serve as a framework for Fluid-Structure Interaction (FSI) problems. The NURBS basis functions can be used to build an interpolation matrix for transferring information between non-matching interfaces.

- Another possibility is the extension of the volumetric NURBS technique to be based upon T-Splines [165, 86] or the latest development of U-Splines [180]. Such techniques, will allow more complicated constraints to be applied of the movement without the need to introduce a large number of additional control points.

- An interesting possible development concerns Isogeometric Analysis (IGA) methods that have attracted a lot of interest for Finite Element applications [71]. Such methods, use volumetric NURBS or T-Splines for both the geometry description and analysis phase. These are higher-order methods that have achieved noticeable reduction in computational cost for FEA applications. There exist a handful of works on the development of IGA methods for CFD and the Finite Volume technique (e.g. [67]), but the field may well be the focus in the next years. The development of the adjoint method for IGA codes will be even more appealing, since it will maintain CAD compatibility throughout the optimization, eliminating, at the same time, the need for mesh generation and deformation overall.

- The current PhD thesis is devoted to the subject of aerodynamic shape optimization. Topology optimization is another important field where the adjoint method has been used extensively. The extension of the adjoint solver of PUMA for topology optimization is, consequently, a natural next step.

Appendix A

Extracting GMTurbo parameters from a CAD or CFD mesh

In this appendix, the process of computing the parameters needed by GMTurbo to reconstruct a given blade row geometry is presented. It is assumed that the geometry is given in the form of either a CFD mesh or a CAD model describing it as a set of NURBS surfaces.

A CFD mesh is comprised of all the nodal coordinates, the connectivity among the nodes and the description of mesh boundaries through the identification of the boundary patches of the mesh. Given all these pieces of information, a series of topological data such as the edges of the mesh connecting the nodes, the edges of each boundary patch e.t.c. can be extracted. This is a useful piece of information to be used in the process of reconstructing the row geometry. Without loss of generality, it is assumed that the boundary of the CFD mesh is described by the following boundary patches:

- Hub and shroud patches (S_H and S_S respectively),
- Inlet and outlet patches (S_I and S_O),
- Two Periodic patches (S_P, S_{P^*}),
- Pressure and suction side patches on the blade surface (S_{PS}, S_{SS}) and
- optionally, leading (LE) and trailing edge (TE) patches (S_{LE}, S_{TE}) in case one of these edges is either of blunt, wedge or dovetail shape.

In what follows, surface patches will be denoted by S. Nodes belonging to two patches (i.e. lying on shared edges) are denoted by $E_{A,B}$, where A and B are the two patches. Similarly, corner nodes shared by three boundary patches are denoted as $C_{A,B,D}$. Consequently, given three patches A, B and D the following relations hold

$$
\begin{gathered}
E_{A,B} = S_A \cap S_B \\
E_{A,B} \subset S_A, \;\; E_{A,B} \subset S_B \\
C_{A,B,D} = E_{A,B} \cap S_D \\
C_{A,B,D} \subset E_{A,B}, \;\; C_{A,B,D} \subset E_{A,D}, \;\; C_{A,B,D} \subset E_{B,D}
\end{gathered} \tag{A.1}
$$

In order to reconstruct the given geometry, the following data need to be extracted either from the CFD mesh or the given CAD model:

- meridional contour of the hub and shroud,
- a series of camber lines of the blade at different spanwise positions and
- thickness profiles and spanwise thickness distributions of the blade.

The steps to extract these data are described in the following sections.

A.1 Extraction of the Meridional Contour of the Geometry

In case a CFD mesh is given, the nodal description of the edge $E_{H,P}$ between the hub and the a periodic patch are identified. Then, these nodes are arranged, using topological information from the mesh connectivity, starting from $C_{H,P,I}$ and ending at $C_{H,P,O}$. The resulting set of nodes is mapped from the 3D Cartesian space onto the meridional (z, r) plane and, then, is either interpolated or fitted by a NURBS curve. This process results in the construction of the meridional projection of the hub generatrix. A similar process is followed for the projection of the shroud generatrix. In case a tip gap exists, the same process is folllowed using the nodes share by S_{PS} and S_T or S_{SS} and S_T, with S_T denoting the tip patch. In this case, the generated meridional curve is extended up to the inlet and down to the outlet of the CFD mesh.

If a CAD model is given and the hub (or shroud) geometry is described as a NURBS surface, then an isoparametric curve (at any parameter in the parametric direction not associated with the direction of revolution) is extracted, points are created on the this curve, mapped onto the meridional plane and a NURBS generatrix is created. In many cases, the original CAD model of a row already contains the hub and shroud meridional generatrices which are, hence, retained in the GMTurbo description.

A.2 Extraction of Spanwise Blade Sections

Let the generatrices of the hub and shroud in the meridional plane be denoted as $\mathbf{C}_H$ and $\mathbf{C}_S$, respectively. These two curves are brought to common knot

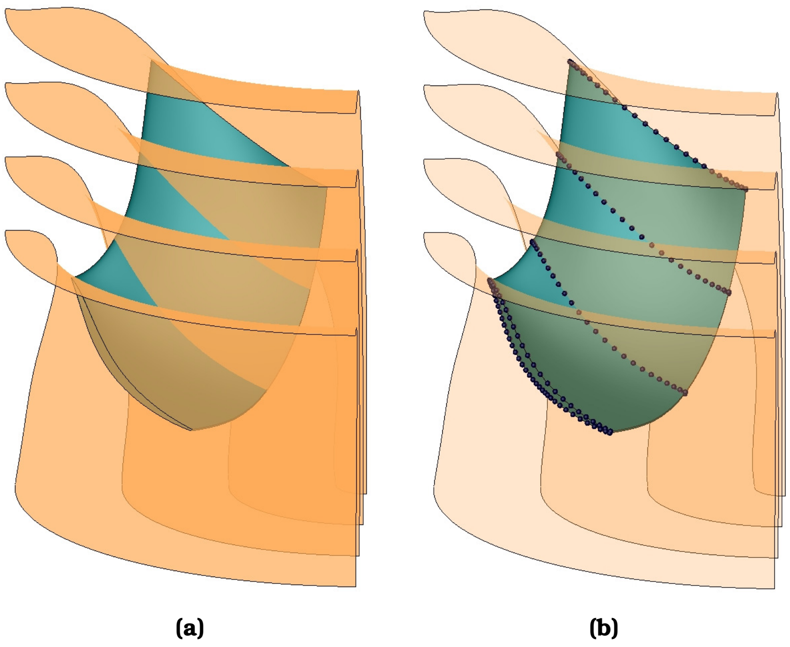


Figure A.1: (a) Intermediate surfaces of revolution at fixed span spositions along the blade are generated, in order to extract data from each span blade section. (b) The points where each surface of revolution intersects the blade surfaces (or patches S_{PS} and S_{SS}) form airfoils at different span positions.

vectors and degrees using knot insertion and degree elevation algorithms, leading to curves $\hat{\mathbf{C}}_H$ and $\hat{\mathbf{C}}_S$. Intermediate generatrices are generated between $\hat{\mathbf{C}}_H$ and $\hat{\mathbf{C}}_S$ by linear interpolation along the the spanwise direction and are, afterwards, used as the basis to construct isospan surfaces of revolution. Let $\mathbf{R}^i$ denote the ith out of these surfaces.

For a given $\mathbf{R}^i$, in case a CFD mesh is available, points on the blade pressure side surface are generated by intersecting S_{PS} with $\mathbf{R}^i$. The points are arranged starting from the intersection of $\mathbf{R}^i$ with $E_{PS,SS}$ at the LE and ending at the intersection of $\mathbf{R}^i$ with $E_{PS,SS}$ at the TE. If a blunt LE or TE is used, then the corresponding end-point is identified as the intersection of $\mathbf{R}^i$ with $E_{PS,TE}$ or $E_{PS,LE}$, respectively (Fig. A.1). The same process is repeated for the suction side of the blade. The computed points are mapped from 3D Cartesian space onto the (m, θ)-plane using Eq. 5.2.1, effectively mapping the ith blade section onto (m, θ).

The points along the airfoil are interpolated using a NURBS curve with an appropriate level of continuity at the LE and TE positions. If any of the LE and TE is of circular arc type, the radius of the circular arc is computed using the curvature of the NURBS curve at the corresponding position, taking also into account Eq. 5.2.3.

Repeating this process for all $\mathbf{R}_i$ from hub to shroud, a number of airfoils are generated and mapped onto (m, θ)-plane. Since points are already computed on the LE and TE in the 3D space, these can be mapped onto the meridional plane and fitted by NURBS curves, yielding the meridional projection of the blade.

A.3 Extraction of Blade Metal and Auxiliary Angles

Having computed the airfoils of the blade at several spanwise positions, the camber line of these airfoils must be computed, so that spanwise distributions of the blade's metal angles can be extracted.

Let M_0 be the LE point. A series of N_1 points are generated on one side of the blade (the choice of the side is irrelevant) denoted as A_i, $(i = 1, \dots, N_1)$. For each A_i, a point B_i must be identified on the opposite blade side, so that $\mathbf{A}_i\mathbf{B}_i \perp \mathbf{t}$, where $\mathbf{t}$ is the vector tangent to the camber line computed as $\mathbf{t} = \mathbf{M}_{i-1}\tilde{\mathbf{M}}_i$. $\tilde{M}_i$ is the midpoint of $\mathbf{A}_i\mathbf{B}_i$. If such a point B_i is identified, then a new point on the camber line $M_i = \tilde{M}_i$ results. The process is illustrated in Fig. A.2.

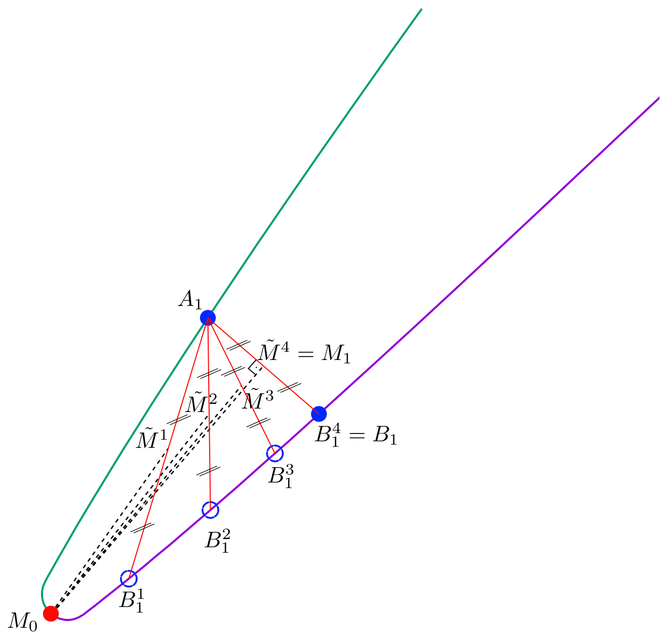


Figure A.2: Illustration of the algorithm employed to compute the camber line of an airfoil, given its two sides and the LE position. For each point A_i on the one side, a series of points B_i^j are generated on the opposite side, until the one that matches the prescribed criteria is encountered. Consequently, several trial points $\tilde{M}^j$ are generated until the new camber line point M_i is found.

The computed camber line points M_i, $i = 0, \dots, N_1$ are then fitted by a cubic Bezier curve. If the control points of this Bezier curve are denoted as P_0, P_1, P_2 and P_3, respectively, the θ_{LE}, θ_{TE}, β_{LE}, β_{TE}, δ_{LE} and δ_{TE} angles can be computed as

$$
\begin{aligned}
\theta_{\text{LE}} &= \theta_{P_0} \\
\theta_{\text{TE}} &= \theta_{P_3} \\
\beta_{\text{LE}} &= \arctan\left(\frac{\theta_{P_1}-\theta_{P_0}}{m_{P_1}-m_{P_0}}\right) \\
\beta_{\text{TE}} &= \arctan\left(\frac{\theta_{P_3}-\theta_{P_2}}{m_{P_3}-m_{P_2}}\right) \\
\delta_{\text{LE}} &= \arctan\left(\frac{\theta_{P_1}-\theta_{CM}}{m_{P_1}-m_{CM}}\right)-\gamma \\
\delta_{\text{TE}} &= \arctan\left(\frac{\theta_{P_2}-\theta_{CM}}{m_{P_2}-m_{CM}}\right)-\gamma \\
\gamma &= \arctan\left(\frac{\theta_{P_3}-\theta_{P_0}}{m_{P_3}-m_{P_0}}\right)
\end{aligned}
\tag{A.3.1}
$$

where CM stands for the midpoint of the chord $\mathbf{P}_0\mathbf{P}_3$.

Once the blade angles become available at several spanwise positions, they are fitted with NURBS curves that define their spanwise distribution.

A.4 Extraction of Thickness Data

Having computed the camber lines at several spanwise positions, the thickness data can also be extracted. Points are generated along the camber line based on normalized arc-length positions computed via a stretching function. Common stretching functions are the hyperbolic trigonometric functions such as the hyperbolic tangent. From each point on the camber line, a straigth line is generated perpendicular to it, intersecting both airfoil sides. The distance from the camber line point to each intersection point corresponds to the half-thickness value at this normalized arc-length position, scaled appropriately (Eq. 5.2.3). Among the τ values extracted, the maximum is found and assigned to τ_{max}. All half-thickness values are then normalized by τ_{max}, yielding pairs of $(s, \hat{\tau})$. Fitting these pairs results in the normalized thickness profile curve at this spanwise position. Finally, all values of τ_{max} at different spanwise positions are fitted with a NURBS curve to yield the spanwise distribution of the half-thickness for each blade side. At this point it is worth repeating that the half-thickness need not be symmetric for the two sides of the blade.

Appendix B

Adapting turbomachinery mesh to GMTurbo generated geometry

In this appendix, the process of adapting an existing CFD mesh to a geometry generated by GMTurbo is described. If the GMTurbo parameters have been extracted by a pre-existing CFD mesh, this process is also used to adapt the CFD mesh to the reconstructed geometry. In such a case, some additional work is needed, to obtain the parametric coordinates of the nodes on the corresponding surfaces. Since the process of adapting a 3D CFD mesh to a reconstructed geometry is more general than the process of adapting it to a modified one later on, only the former is presented. At the end, it will be clear which steps of the process are omitted when deforming the geometry.

The CFD mesh is assumed to be described by the boundary patches listed in Appendix A. In addition, the following surfaces are made available by GMTurbo:

- a surface for each blade side ($\mathbf{G}_{PS}$ and $\mathbf{G}_{SS}$),
- one surface for the hub and one for the shroud ($\mathbf{G}_{H}$ and $\mathbf{G}_{S}$),
- a surface for the LE and/or TE if a blunt, wedge or dovetail type edge is used ($\mathbf{G}_{LE}$ and/or $\mathbf{G}_{TE}$), and
- a surface for the tip of the blade, if such a feature is present ($\mathbf{G}_{T}$).

Since $\mathbf{G}_{H}$, $\mathbf{G}_{S}$ and $\mathbf{G}_{T}$ are surfaces of revolution, representing them as functions of m and θ proves more handy, especially when it comes to periodicity treatment (periodic nodes retain the same m coordinate and θ differing by the blade row pitch). The surfaces describing the blade itself retain their initial representation using the (u, v) parametric coordinate system.

The process of adapting the CFD mesh to the reconstructed geometry is split into 5 steps which are developed in the following sections.

B.1 Projecting the CFD Surface Nodes onto the Geometry

The first step concerns the projection of the nodes of each boundary patch of the mesh on its corresponding surface. Herein, the term projection refers to the process of finding the closest point $\mathbf{P}^*$ lying on a parametric surface, given a point $\mathbf{P}$ in the 3D space. The algorithm for implementing such a process can be found in [149]. A by-product of the projection process is that a pair of parametric coordinates (u, v) or (m, θ) is assigned to $\mathbf{P}$, being the pair that reproduces point $\mathbf{P}^*$. If $\mathbf{P}$ lies on the surface under consideration, then the projection process is equivalent to inverting the vector valued function representing the surface.

The pair of parametric coordinates assigned to each surface node is a very useful piece of information, since it provides the means of treating the surface mesh as a planar 2D one in the parametric space. However, since the reconstructed geometry does not match exactly the CFD mesh, simply projecting the surface nodes may potentially create gaps or folds in areas where surfaces intersect. For this reason, these areas are treated in a special way described in the next section.

B.2 Adapting Mesh Nodes onto Surface Intersections

The process of adapting the nodes on the surface intersections will be described for the intersection of the blade's pressure side $\mathbf{G}_{PS}$ with the hub surface $\mathbf{G}_H$. The same process is repeated for the rest of the surface intersections of the mesh.

Initially, the length of $E_{H,PS}$ is computed from the CFD mesh and a normalized arc-length value $\hat{s}$ is assigned to each node. Next, a NURBS surface–NURBS surface intersection algorithm is employed to compute the intersection curves between $\mathbf{G}_H$ and $\mathbf{G}_{PS}$. In fact, no unique intersection curve can easily be computed between to general NURBS surfaces. The intersection curves result as NURBS curves in the form

$$\mathbf{G}_H \cap \mathbf{G}_{PS} \Rightarrow \begin{cases} \mathbf{C}_H^{PS}(t) : [0,1] \rightarrow (m,\theta)_H \\ \mathbf{C}_{PS}^{H}(t) : [0,1] \rightarrow (u,v)_{PS} \end{cases} \tag{B.2.1}$$

where parameter t is common between the two intersection curves. Using the $\hat{s}$ values computed from the CFD mesh, corresponding values of t are computed and the mesh nodes in $E_{H,PS}$ are assigned parametric coordinates using the parametric form of the two intersection curves (Eq. B.2.1).

After adapting the mesh nodes to the surface intersection, folds and gaps are avoided in these areas. However, since the parametric coordinates of these mesh nodes may differ from the ones computed by the projection step, each surface mesh in parametric space needs to be adapted to this repositioning as well. This is done by employing 2D mesh deformation techniques developed in the past and used by the PCOpt/NTUA [182], [96]. Indicatively, some of them are the linear or torsional spring analogy methods [44, 36], the Laplace PDE-based mesh deformation [120] and the elastic medium analogy [39]. These methods propagate the displacement of the boundary nodes (of the surface mesh mapped onto the parametric space) into the interior of the mesh. In addition, they allow the treatment of flexible periodic boundaries, free to move in a periodic manner in the θ direction (applicable for the hub and shroud surface meshes).

This last step, that of retrofitting the parametric mesh of each surface on the adapted surface intersection nodes, leads to a pair of parametric coordinates (u, v) or (m, θ) for each surface mesh node, that is used as the non-deformed state for subsequent mesh adaptations that may be carried out (e.g. during a parametric study or an optimization process).

B.3 Finalizing the 3D Surface Mesh and Adapting the Volume Mesh

The final step is the computation of the 3D Cartesian coordinates for each surface mesh node and the adaptation of the volume mesh. The former is performed by simply evaluating the points on each NURBS surface (for blade surfaces) for given (u, v) values previously computed, or evaluating the points on surfaces of revolution for given (m, θ) values. Using algorithms described in Chapter 4 and Chapter 5 this process is straightforward.

The final adaptation of the 3D volume mesh to the newly computed 3D surface boundaries is carried out using the 3D variants of the mesh deformation techniques used for adapting the 2D parametric meshes. Again, periodic boundaries are allowed to move in a periodic manner.

In case the initial CFD mesh is generated directly on the geometry generated by the GMTurbo, the projection step is redundant, since the parametric coordinates of the surface nodes are known by the mesh generation process. This is also the case when adapting a mesh to any geometry arising during an optimization loop. The rest of the mesh adaptation steps are still necessary.

Bibliography

[1] The EASY (Evolutionary Algorithms SYstem) software, http://velos0.ltt.mech.ntua.gr/EASY, 2008.

[2] T. Akenine-Moller, E. Haines, and N. Hoffman. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 3rd edition, 2008.

[3] W. Anderson, J. Newmann, D. Whitfield, and E. Nielsen. Sensitivity Analysis for Navier-Stokes Equations on Unstructured Meshes using Complex Variables. *AIAA Journal*, 39(1):56–63, 2001.

[4] W. Anderson and V. Venkatakrishnan. Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation. *Computers & Fluids*, 28(4):443 – 480, 1999.

[5] L. Armijo. Minimization of Functions having Lipschitz Continuous First Partial Derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[6] V.G. Asouti. *Aerodynamic analysis and design methods at high and low speed flows, on multiprocessor platforms*. PhD thesis, National Technical University of Athens, 2009.

[7] V.G. Asouti, D.I. Papadimitriou, D.G. Koubogiannis, and K.C. Giannakoglou. Low Mach Number Preconditioning for 2D and 3D Upwind Flow Solution Schemes on Unstructured Grids. In *5th GRACM International Congress on Computational Mechanics*, Limassol, Cyprus, June 29 - July 1 2005.

[8] V.G. Asouti, X.S. Trompoukis, I.C. Kampolis, and K.C. Giannakoglou. Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units. *International Journal for Numerical Methods in Fluids*, 67(2):232–246, 2011.

[9] V.G. Asouti, A.S. Zymaris, D.I. Papadimitriou, and K.C. Giannakoglou. Continuous and discrete adjoint approaches for aerodynamic shape optimization with low Mach number preconditioning. *International Journal for Numerical Methods in Fluids*, 57(10):1485–1504, 2008.

[10] A. Barr. Global and Local Deformations of Solid Primitives. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 21–30, New York, NY, USA, 1984.

[11] T. Barth and C. Jespersen. The Design and Application of Upwind Schemes on Unstructured Meshes. *AIAA paper*, 0366, 02 1989.

[12] O. Baysal and M. Eleshaky. Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations. *Journal of Fluids Engineering*, 113(4):681–688, 1991.

[13] R. Bellman. *Dynamic Programming*. Dover Publications, Inc., New York, NY, USA, 2003.

[14] HG. Beyer and HP Schwefel. Evolution strategies { a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[15] M. Bloor and M. Wilson. Efficient Parametrization of Generic Aircraft Geometry. *Journal of Aircraft*, 32(6):1269–1275, 1995.

[16] T. Bogar, M. Sajben, and J. Kroutil. Characteristic Frequencies of Transonic Diffuser Flow Oscillations. *AIAA Journal*, 21(9):1232–1240, 1983.

[17] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder. Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid. *ACM Transactions on Graphics*, 22(3):917–924, 2003.

[18] JF. Bonnans, JC. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization, Theoretical and Practical Aspects*. Springer, Berlin, Heidelberg, Germany, 2006.

[19] K. Bootpathy and M. Rumpfkeil. A Multivariate Interpolation and Regression Enhanced Kriging Surrogate Model. In *21st AIAA Computational Fluid Dynamics Conference*, San Diego, CA, USA, June 24 - June 27 2013.

[20] T. Brandvik and G. Pullan. Acceleration of a 3D Euler Solver Using Commodity Graphics Hardware. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, 7 January–10 January, 2008.

[21] T. Brandvik and G. Pullan. An Accelerated 3D Navier-Stokes Solver for Flows in Turbomachines. *Journal of Turbomachinery*, 133(2), 2011.

[22] D. Büche, N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(2):183–194, 2005.

[23] A. Bueno-Orovio, C. Castro, F. Palacios, and E. Zuazua. Continuous Adjoint Approach for the Spalart-Allmaras Model in Aerodynamic Optimization. *AIAA Journal*, 50(3):631–646, 2012.

[24] K. R. Butterfield. The Computation of all the Derivatives of a B-spline Basis. *IMA Journal of Applied Mathematics*, 17(1):15–25, 1976.

[25] R. Byrd, HF. Khalfan, and R. Schnabel. Analysis of a Symmetric Rank-One Trust Region Method. *SIAM Journal on Optimization*, 6(4):1025–1039, 1996.

[26] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal of Scientific Computing*, 16(5):1190–1208, 1995.

[27] J-R. Carlson, V. Vatsa, and J. White. Validation of a Node-Centered Wall Function Model for the Unstructured Flow code FUN3D. In *22nd AIAA Computational Fluid Dynamics Conference*, Dallas, TX., June 22 - June 26 2015.

[28] C. Chen, M. Sajben, and J. Kroutil. Shock Wave Oscillations in a Transonic Diffuser Flow. *AIAA Journal*, 17(10):1076–1083, 1979.

[29] A. Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems. *Journal of Computational Physics*, 2(1):12 – 26, 1967.

[30] S. Coquillart. Extended Free-form Deformation: A Sculpturing Tool for 3D Geometric Modeling. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 187–196, Dallas, TX, USA, 1990.

[31] NVIDIA Corporation. NVIDIA CUDA Homepage. `developer.nvidia.com/cuda-zone`, 2019.

[32] R. Corral, F. Gisbert, and J. Pueblas. Execution of a Parallel Edge-Based Navier-Stokes Solver on Commodity Graphics Processor Units. *International Journal of Computational Fluid Dynamics*, 31(2):93–108, 2017.

[33] J. Crosby. *Computer Simulation in Genetics*. John Wiley & Sons, London, UK, 1973.

[34] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favored Races in the Struggle for Life*. Murray, London, UK, 1859.

[35] J. Deardorff. A Numerical Study of Three-dimensional Turbulent Channel Flow at Large Reynolds Numbers. *Journal of Fluid Mechanics*, 41(2):453{480, 1970.

[36] C. Degand and C. Farhat. A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes. *Computers & Structures*, 80(3):305 – 316, 2002.

[37] J.-A. Désidéri and A. Janka. Hierarchical Parametrization for Multilevel Evolutionary Shape Optimization with Application to Aerodynamics. In *EUROGEN 2003, Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, Barcelona,Spain, 2003.

[38] M. Duta, S. Shahpar, and M. Giles. Turbomachinery Design Optimization using Automatic Differentiated Adjoint Code. In *ASME Turbo Expo 2007: Power for Land, Sea and Air*, Montreal, Canada, May 14 - May 17 2007.

[39] R. Dwight. Robust Mesh Deformation using the Linear Elasticity Equations. In *Computational Fluid Dynamics 2006*, pages 401–406, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[40] H. Elbanna and L. Carlson. Determination of Aerodynamic Sensitivity Coefficients in the Transonic and Supersonic Regimes. In *27th Aerospace Sciences Meeting*, Reno, NV, USA, 9 January - 12 January 1989.

[41] J. Elliott and J. Peraire. Practical Three-Dimensional Aerodynamic Design and Optimization using Unstructured Meshes. *AIAA Journal*, 35(9):1479–1485, 1997.

[42] E. Elsen, P. LeGresley, and E. Darve. Large Calculation of the Flow over a Hypersonic Vehicle using a GPU. *Journal of Computational Physics*, 227(24):10148 – 10161, 2008.

[43] L.E. Eriksson. A Preconditioned Navier-Stokes solver for low Mach number flows. *Computational Fluid Dynamics*, 1996.

[44] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional Springs for Two-Dimensional Dynamic Unstructured Fluid Meshes. *Computer Methods in Applied Mechanics and Engineering*, 163(1):231 – 245, 1998.

[45] G. Farin. *Curves and Surfaces for CAGD*. Morgan Kauffmann, 2001.

[46] R. Fletcher. *Practical Methods of Optimization; (2nd Ed.)*. New York, NY, USA.

[47] N. Foster and G. Dulikravich. Three-Dimensional Aerodynamic Shape Optimization Using Genetic and Gradient Search Algorithms.

[48] A. Fraser and D. Burnell. *Computer Models in Genetics*. McGraw-Hill, New York, USA, 1970.

[49] J. Fung and S. Mann. Using Multiple Graphics Cards as a General Purpose Parallel Computer: Applications to Computer Vision. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 1, pages 805–808, 2004.

[50] P. Gage, I. Kroo, and P. Sobieski. Variable-complexity Genetic Algorithm for Topological Design. *AIAA Journal*, 33(11):2212–2217, 1995.

[51] F. Gagliardi. *Shape Parameterization and Constrained Aerodynamic Optimization. Applications including Turbomachines*. PhD thesis, National Technical University of Athens, (in progress).

[52] F. Gagliardi, K. Tsiakas, and K. Giannakoglou. A Two-Step Mesh Adaptation Tool Based on RBF with Application to Turbomachinery Optimization Loops. In *EUROGEN 2017, International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Madrid, Spain, September 13–September 15, 2017.

[53] General Secretariat of Research and Technology. General Secretariat of Research and Technology. www.gsrt.gr.

[54] U. Ghia, K. Ghia, and C. Shin. High-Re Solutions for Incompressible Flow using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, 48:387–411, 1982.

[55] M. Giles and N. Pierce. Adjoint Equations in CFD - Duality, Boundary Conditions and Solution Behaviour. In *13th Computational Fluid Dynamics Conference*, Snowmass Village, CO, USA, 29 June - 2 July 1997.

[56] F. Gisbert, R. Corral, and J. Pueblas. Computation of Turbomachinery Flows with a Parallel Unstructured Mesh Navier-Stokes Equations Solver on GPUs. In *21st AIAA Computational Fluid Dynamics Conference*, San Diego, CA, USA, 24 June–27 June, 2013.

[57] K. Gkaragkounis. *The Continuous Adjoint Method with Emphasis to Turbulence Models and Applications in Aerodynamic Shape Optimization*. PhD thesis, National Technical University of Athens, (in progress).

[58] NASA Glenn Research Center. WIND-US documentation. www.grc.nasa.gov/www/winddocs.

[59] N. Goodnight, C. Woolley, G. Lewin, D. Luebke, and G. Humphreys. A Multigrid Solver for Boundary Value Problems Using Programmable Graphics Hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 102–111, 2003.

[60] J. Gräsel, A. Keskin, M. Swoboda, H. Przewozny, and A. Saxer. A Full Parametric Model for Turbomachinery Blade Desing and Optimisation. In *ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Salt Lake City, Utah, USA, September 28 - October 2 2004.

[61] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2008.

[62] OpenCFD Ltd. (ESI Group). OpenFOAM the Open Source CFD Toolbox. www.openfoam.com.

[63] T. Hagen, K-A. Lie, and J. Natvig. Solving the Euler Equations on Graphics Processing Units. In *Computational Science - ICCS 2006*, pages 220–227. Springer Berlin Heidelberg, 2006.

[64] J. Hager and K. Lee. A Multi-point Optimization for Transonic Airfoil Design. In *4th Symposium on Multidisciplinary Analysis and Optimization*, Cleveland,OH,USA, 21 September - 23 September 1992.

[65] M. Harris, W. Baxter, T. Scheuermann, and A. Lastra. Simulation of Cloud Dynamics on Graphics Hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 92–101, 2003.

[66] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[67] C. Heinrich, B. Simeon, and S. Boschert. A Finite Volume Method on NURBS Geometries and its Application in Isogeometric Fluid{Structure Interaction. *Mathematics and Computers in Simulation*, 82(9):1645 – 1666, 2012.

[68] R. Hicks and P. Henne. Wing Design by Numerical Optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[69] C. Hirsch. *Numerical Computation of Internal and External Flows*, volume 1: Fundamentals of Numerical Discretization. 1988.

[70] W. Hsu, J. Hughes, and H. Kaufman. Direct Manipulation of Free-form Deformations. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, New York, NY, USA, 1992.

[71] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[72] ANSYS Inc. ANSYS Fluent webpage. www.ansys.com/products/fluids/ansys-fluent.

[73] Khronos Group Inc. OpenCL Homepage. www.khronos.org/opencl/, 2019.

[74] NUMECA International. NUMECA webpage. www.numeca.com/home.

[75] P. Miller IV, J. Oliver, D. Miller, and D. Tweedt. BladeCAD: An Interactive Geometric Design Tool for Turbomachinery Blades. In *ASME 1996 International Gas Turbine and Aeroengine Congress and Exhibition*, Birmingham, UK, June 10 - June 13 1996.

[76] A. Jameson. Aerodynamic Design via Control Theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[77] A. Jameson, T. Baker, and N. Weatherill. Calculation of Inviscid Transonic Flow over a Complete Aircraft. In *24th Aerospace Sciences Meeting*, Reno, NV, USA, 6 January–9 January, 1986.

[78] A. Jameson and J. Reuther. Control Theory based Airfoil Design using the Euler Equations. In *5th Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, USA, 7 September - 9 September 1994.

[79] A. Jameson, W. Schmidt, and E. Turkel. Numerical Solution of the Euler Equations by Finite Volume Methods using Runge-Kutta Time Stepping Schemes. In *14th Fluid and Plasma Dynamics Conference*, Palo Alto, CA, USA, 23 June–25 June, 1981.

[80] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1:61–70, 2011.

[81] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic Coordinates for Character Articulation. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, San Diego, California, 2007.

[82] Y. Kallinderis. A 3-d Finite-Vvolume Method for the navier-stokes Equations with Adaptive Hybrid Grids. *Applied Numerical Mathematics*, 20(4):387 – 406, 1996. Adaptive mesh refinement methods for CFD applications.

[83] I. Kampolis and K. Giannakoglou. A Multilevel Approach to Single- and Multiobjective Aerodynamic Optimization. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2963–2975, 2008.

[84] I.C. Kampolis and K.C. Giannakoglou. Distributed evolutionary algorithms with hierarchical evaluation, engineering optimization. *Engineering Optimization*, 41(11):1037–1049, 2009.

[85] I.C. Kampolis, X.S. Trompoukis, V.G Asouti, and K.C. Giannakoglou. CFD-based analysis and two-level aerodynamic optimization on graphics processing units. *Computer Methods in Applied Mechanics and Engineering*, 199(9-12):712–722, 2009.

[86] H. Kang, F. Chen, and J. Deng. Modified T-Splines. *Computer Aided Geometric Design*, 30:827–843, 2013.

[87] D. Kapsoulis. *Development of prediction models for use in the low-cost evolutionary algorithm based optimization*. PhD thesis, National Technical University of Athens, (in progress).

[88] D. Kapsoulis, K. Tsiakas, V. Asouti, and K. Giannakoglou. The use of Kernel PCA in Evolutionary Optimization for Computationally Demanding Engineering Applications. In *2016 IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2016)*, Athens, Greece, December 6–December9, 2016.

[89] D. Kapsoulis, K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. 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, 2018.

[90] D. Kapsoulis, K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. PCA-Assisted Hybrid Algorithm Combining EAs and Adjoint Methods for CFD-based Optimization. *Applied Soft Computing*, 73:520–529, 2018.

[91] M.K. Karakasis and K.C. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8):941–957, 2006.

[92] M.K. Karakasis, A.P. Giotis, and K.C. Giannakoglou. Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. *International Journal for Numerical Methods in Fluids*, 43(10-11):1149–1166, 2003.

[93] M.K. Karakasis, D.G. Koubogiannis, and K.C. Giannakoglou. Hierarchical distributed evolutionary algorithms in shape optimization. *International Journal for Numerical Methods in Fluids*, 53(3):455–469, 2007.

[94] I. Kavvadias, E. Papoutsis-Kiachagias, G. Dimitrakopoulos, and K. Giannakoglou. The Continuous Adjoint Approach to the k–ω SST Turbulence Model with Applications in Shape Optimization. *Engineering Optimization*, 47(11):1523–1542, 2015.

[95] I. Kavvadias, E. Papoutsis-Kiachagias, and K. Giannakoglou. On the Proper Treatment of Grid Sensitivities in Continuous Adjoint Methods for Shape Optimization. *Journal of Computational Physics*, 301:1–18, 2015.

[96] I.S. Kavvadias. *Continuous adjoint methods for steady and unsteady turbulent flows with emphasis on the accuracy of sensitivity derivatives.* PhD thesis, National Technical University of Athens, 2016.

[97] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, 1995.

[98] G. Kenway and J. Martins. Aerostructural Shape Optimization of Wind Turbine Blades Considering Site-Specific Winds. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, British Columbia, Canada, September 10 - September 12 2008.

[99] J-H. Kim, B. Ovgor, K-H. Cha, J-H. Kim, S. Lee, and K-Y Kim. Optimization of the Aerodynamic and Aeroacoustic Performance of an Axial-Flow Fan. *AIAA Journal*, 52(9):2032–2044, 2014.

[100] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–880, 1983.

[101] G. Koini, S. Sarakinos, and I. Nikolos. A Software Tool for Parametric Design of Turbomachinery Blades. *Advances in Engineering Software*, 40:41–51, 2009.

[102] E.A. Kontoleontos. *Designing Thermo-Fluid Systems using Gradient-based Optimization Methods and Evolutionary Algorithms.* PhD thesis, National Technical University of Athens, 2012.

[103] E.A Kontoleontos, V.G. Asouti, and K.C. Giannakoglou. An asynchronous metamodel-assisted memetic algorithm for cfd-based shape optimization. *Engineering Optimization*, 44(2):157–173, 2012.

[104] D. Koubogiannis. *Numerical Solution of the Navier-Stokes Equations on Unstructured Grids in a Paralle Processig Environment.* PhD thesis, National Technical University of Athens, 1998.

[105] J. Krüger and R. Westermann. Linear Algebra Operators for GPU Implementation of Numerical Algorithms. *ACM Transactions on Graphics*, 22(3):908–916, 2003.

[106] A Kumar, Andy Keane, P.B. Nair, and Shahrokh Shahpar. Efficient robust design for manufacturing process capability. In *Proceedings of the 6th ASMO UK Conference on Engineering Design Optimization*, 2006.

[107] S. Kyriacou. *Evolutionary Algorithm-based Design-Optimization Methods in Turbomachinery*. PhD thesis, National Technical University of Athens, 2013.

[108] N. Lambropoulos. *Multigrid Techniques and Parallel Processing for the Numerical Prediction of Flow Fields through Thermal Turbomachines, using Unstructured Grids*. PhD thesis, National Technical University of Athens, 2005.

[109] NASA Langley Research Center. CFL3D Home Page. https://cfl3d.larc.nasa.gov/.

[110] NASA Langley Research Center. FUN3D Home Page. https://fun3d.larc.nasa.gov/.

[111] NASA Langley Research Center. USM3D Home Page. https://tetruss.larc.nasa.gov/usm3d/.

[112] B. Launder and D. Spalding. The Numerical Computation of Turbulent Flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269 – 289, 1974.

[113] D. Lee. *Local Preconditioning of the Euler and Navier-Stokes Equations*. PhD thesis, Aerospace Engineering, University of Michigan, 1996.

[114] J. Leiva and B. Watson. Automatic Generation of Basis Vectors for Shape Optimization in the GENESIS Program. In *7th AIAA/USAF/-NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis,MO,USA, 2 September - 4 September 1998.

[115] M. Lesoinne and C. Farhat. Geometric Conservation Laws for Flow Problems with Moving Boundaries and Deformable meshes, and their Impact on Aeroelastic Computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 07 1996.

[116] W. Li, X. We, and A. Kaufman. Implementing Lattice Boltzmann Computation on Graphics Hardware. *The Visual Computer*, 19(7):444–456, Dec 2003.

[117] J. Lions. *Optimal Control of Systems Governed by Partial Differential Equations*. Springer-Verlang, New York, NY, USA, 1971.

[118] Y. Liu, X. Liu, and E. Wu. Real-Time 3D Fluid Simulation on GPU with Complex Obstacles. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 247–256, 2004.

[119] C. Lozano, E. Andrés, M. Martin, and P. Bitrián. Domain versus boundary computation of flow sensitivities with the continuous adjoint method for aerodynamic shape optimization problems. *Numerical Methods in Fluids*, 70(10):1305–1323, 2012.

[120] R. Löhner and C. Yang. Improved ALE Mesh Velocities for Moving Bodies. *Communications in Numerical Methods in Engineering*, 12(10):599–608, 1996.

[121] M. Martin, E. Andres, C. Lozano, and E. Valero. Volumetric B-Splines shape parameterization for aerodynamic shape design. *Aerospace Science and Technology*, 37:26–36, 2014.

[122] J. Martins, P. Sturdza, and J. Alonso. The Complex-step Derivative Approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003.

[123] Mathematics, Argonne National Laboratory Computer Science Division, and Rice University Center for Research on Parallel Computation. https://www.mcs.anl.gov/research/projects/adifor/.

[124] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlang, Berlin Heidelberg, Germany, 1996.

[125] Z. Michalewicz. *How to Solve It: Modern Heuristics, 2nd Edition*. Springer-Verlag, Berlin, Heidelberg, 2010.

[126] A. Milli and S. Shahpar. PADRAM: Parametric Design and Rapid Meshing System for Complex Turbomachinery Configurations.

[127] M. Monfaredi. *CFD-CAA Analysis and Optimization Methods with Industrial Application*. PhD thesis, National Technical University of Athens, (in progress).

[128] D. Montgomery. *Design and Analysis of Experiments*. John Wiley &#38; Sons, Inc., USA, 2006.

[129] O. Navarro-Hinojosa, S. Ruiz-Loza, and M. Alencastre-Miranda. Physically Based Visual Simulation of the Lattice Boltzmann Method on the GPU: A Survey. *Journal of Supercomputing*, 74(7):3441–3467, 2018.

[130] F. Neri and C. Cotta. Memetic Algorithms and Memetic Computing Optimization: A Literature Review. *Swarm and Evolutionary Computation*, 2:1 – 14, 2012.

[131] J. Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlang, New York, NY, USA, 2006.

[132] S. Orszag. Analytical Theories of Turbulence. *Journal of Fluid Mechanics*, 41(2):363{386, 1970.

[133] J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krueger, A. Lefohn, and T. Purcell. A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[134] P. Cook, M. McDonald and M. Firmin. Aerofoil RAE 2822 - Pressure Distributions, and Boundary Layer and Wake Measurements. Report AR 13, AGARD, 1979. Experimental Data Base for Computer Program Assessment.

[135] G. Papadakis, S. Voutsinas, G. Sieros, and T. Chaviaropoulos. CFD Aerodynamic Analysis of Non-Conventional Airfoil Sections for Very Large Rotor Blades. *Journal of Physics: Conference Series*, 555(1):012104, 2014.

[136] D.I. Papadimitriou. *Adjoint formulations for the analysis and design of turbomachinery cascades and optimal grid adaptation using a posteriori error analysis.* PhD thesis, National Technical University of Athens, 2007.

[137] D.I. Papadimitriou and K.C. Giannakoglou. A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows. *Computers & Fluids*, 36(2):325–341, 2007.

[138] D.I. Papadimitriou and K.C. Giannakoglou. A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows. *Journal of Computers & Fluids*, 36(2):325–341, 2007.

[139] D.I. Papadimitriou and K.C. Giannakoglou. Total pressure loss minimization in turbomachinery cascades using a new continuous adjoint formulation. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 221(6):865–872, 2007.

[140] E. Papoutsis-Kiachagias, V. Asouti, K. Giannakoglou, K. Gkagkas, S. Shimokawa, and E. Itakura. Multi-Point Aerodynamic Shape Optimization of Cars based on Continuous Adjoint. *Structural and Multidisciplinary Optimization*, (to appear).

[141] E. Papoutsis-Kiachagias and K. Giannakoglou. Continuous Adjoint Methods for Turbulent Flows, Applied to Shape and Topology Optimization: Industrial Applications. *Archives of Computational Methods in Engineering*, 23(2):255–299, 2016.

[142] E. Papoutsis-Kiachagias, A. Zymaris, I. Kavvadias, D. Papadimitriou, and K. Giannakoglou. The Continuous Adjoint Approach to the k–ϵ Turbulence Model for Shape Optimization and Optimal Active Control of Turbulent Flows. *Engineering Optimization*, 47(3):370–389, 2015.

[143] E.M. Papoutsis-Kiachagias. *Adjoint Methods for Turbulent Flows, Applied to Shape or Topology Optimization and Robust Design.* PhD thesis, National Technical University of Athens, 2013.

[144] E.M. Papoutsis-Kiachagias and K.C. Giannakoglou. Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Archives of Computational Methods in Engineering*, pages 1–45, 2014.

[145] J. Peter and R. Dwight. Numerical Sensitivity Analysis for Aerodynamic Optimization: A Survey of Approaches. *Computers & Fluids*, 39(3):373 – 391, 2010.

[146] R. Picket, M. Rubinstein, and R. Nelson. Automated Structural Synthesis using a Reduced Number of Design Coordinates. In *AIAA/ASME/SAE 14th Structures, Structural, Dynamics, and Materials Conference*, Williamsburg, Virginia, USA, March 20 - March 22 1973.

[147] L. Piegl. *Fundamental Developments in Computer Aided Geometric Modelling*. Academic Press Inc, 1993.

[148] L. Piegl and W. Tiller. A Menagerie of Rational B-Spline Circles. *IEEE Comput. Graph. Appl.*, 9(5):48–56, September 1989.

[149] L. Piegl and W. Tiller. *The NURBS book*. Springer, 1997.

[150] O. Pironneau. Optimal Shape Design for Elliptic Systems. In *System Modeling and Optimization*, pages 42–66, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[151] Pointwise Inc. Pointwise Homepage. `www.pointwise.com`.

[152] D. Poole, C. Allen, and T. Rendall. Optimal Domain Element Shapes for Free-Form Aerodynamic Shape Control. Kissimmee, Florida, USA, 5 January - 9 January 2015.

[153] J. Reuther and A. Jameson. Control Theory based Airfoil Design for Potential Flow and a Finite Volume Discretization. In *32nd Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, 10 January - 13 January 1994.

[154] P. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43(2):357 – 372, 1981.

[155] M. Rossgatterer, B. Jüttler, M. Kapl, and G. Della Vecchia. Medial design of blades for hydroelectric turbines and ship propellers. *Computers & Graphics*, 36:434–44, 2012.

[156] M. Rumpf and R. Strzodka. Nonlinear Diffusion in Graphics Hardware. In *Data Visualization 2001*, pages 75–84, Vienna, 2001. Springer Vienna.

[157] GRNET S.A. GRNET Homepage. `hpc.grnet.gr`.

[158] J. Samareh. Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization. *AIAA Journal*, 39(5):877–884, 2001.

[159] K. Samouchos. *The continuous adjoint method to immersed boundary methods for turbomachinery applications*. PhD thesis, National Technical University of Athens, (in progress).

[160] J. Schepers and K. Boorsma et al. Final Report of IEA Task 29, Mexnext (Phase 1): Analysis of MEXICO Wind Tunnel Measurements. Technical Report ECN-E-12-004, Energy Research Center of the Netherlands, February 2012.

[161] J. Schepers, L. Pascal, and H. Snel. First Results from Mexnext: Analysis of Detailed Aerodynamic Measurements on a 4.5m Diameter Rotor placed in the Large German Dutch Wind Tunell DNW. In *European Wind Energy Conference, EWEC*, Warsaw, Poland, 20 April–23 April, 2010.

[162] J. Schepers and H. Snel. MEXICO, Model Experiments in Controlled Conditions. Technical Report ECN-E-07-042, Energy Research Center of the Netherlands, 2007.

[163] V. Schmitt and F. Charpin. Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers. Technical report, AGARD.

[164] T. Sederberg and S. Parry. Free-form Deformation of Solid Geometric Models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 151–160, New York, NY, USA, 1986.

[165] T. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-Splines and T-NURCCs. *ACM Transactions on Graphics*, 22(3):477–484, 2003.

[166] R. Shang, J. Wang, L. Jiao, and Y. Wang.

[167] H. Sharp and L. Sirovich. Constructing a Continuous Parameter Range of Computational Flows. *AIAA Journal*, 27(10):1326–1331, 1989.

[168] K. Siddappaji, M. Turner, and A. Merchant. General Capability of Parametric 3D Blade Design Tool for Turbomachinery. In *ASME Turbo Expo 2012: Turbine Technical Conference and Exposition*, Copenhagen, Denmark, June 11 - June 15 2012.

[169] S. Skinner and H. Zare-Behtash. State-of-the-art in Aerodynamic Shape Optimisation Methods. *Applied Soft Computing*, 62:933 – 962, 2018.

[170] J. Smagorinsky. General Circulation Experiments with the Primitive Equations. *Monthly Weather Review*, 91:99, 1963.

[171] Siemens PLM Software. STAR-CCM+ webpage. mdx.plm.automation.siemens.com/star-ccm-plus.

[172] INRIA Sophia-Antipolis. TAPENADE. https://www-sop.inria.fr/tropics/tapenade.html.

[173] D. Sorensen. Newton's Method with a Model Trust Region Modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, 1982.

[174] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. *Recherche Aerospatiale*, (1):5–21, 1994.

[175] J. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons Inc., New York, NY, USA, 2003.

[176] J. Stam. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 121–128, 1999.

[177] the Open-Source CFD Code 2018 SU2. SU2, The Open-Source CFD Code. su2code.github.io/.

[178] W. Sutherland. The Viscosity of Gases and Molecular Force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(223):507–531, 1893.

[179] J. Thibault and I. Senocak. CUDA Implementation of a Navier-Stokes Solver on Multi-GPU Desktop Platforms for Incompressible Flows. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Orlando, FL, USA, 5 January–8 January, 2009.

[180] D. Thomas, L. Engvall, S. Schmidt, K. Tew, and M. Scott. U-Splines: Splines over Unstructured Meshes. Technical report, Coreform LLC, 2018.

[181] X. Trompoukis, K. Tsiakas, M. Ghavami Nejad, V. Asouti, and K. Giannakoglou. The Continuous Adjoint Method on Graphics Processing Units for Compressible Flows. In *OPT-i, International Conference on Engineering and Applied Sciences Optimization*, Kos Island, Greece, June 4–June 6 2014.

[182] X.S. Trompoukis. *Solving aerodynamic-aeroelastic problems on Graphics Processing Units*. PhD thesis, National Technical University of Athens, 2012.

[183] K. Tsiakas, F. Gagliardi, X. Trompoukis, and K. Giannakoglou. Shape Optimization of Turbomachinery Rows using a Parametric Blade Modeller and the Continuous Adjoint Method running on GPUS. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, Crete island, Greece, June 5- June 10, 2016.

[184] K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. Design-Optimization of a Compressor Blading on a GPU Cluster. In *EUROGEN 2013, 10th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Las Palmas de Gran Canaria, Spain, October 7–October 9 2013.

[185] K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. Shape Optimization of Wind Turbine Blades using the Continuous Adjoint Method and Volumetric NURBS on a GPU Cluster. In *EUROGEN 2015, 11th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Glasgow, UK, September 14–September16, 2015.

[186] K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. Shape Optimization Using the Continuous Adjoint Method and Volumetric NURBS on a Many GPU System. In *8th GRACM International Congress on Computational Mechanics*, Volos, Greece, July 12–15, 2015.

[187] E. Turkel. Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations. *Journal of Computational Physics*, 72(2):277 – 298, 1987.

[188] B. van Leer. Flux-Vector Splitting for the Euler Equations. In *Eighth International Conference on Numerical Methods in Fluid Dynamics*, pages 507–512, 1982.

[189] V. Venkatakrishnan. On the Accuracy of Limiters and Convergence to Steady State Solutions. *AIAA paper*, 31(08), 02 1993.

[190] C. Vezyris. *OpenFOAM-based continuous adjoint method for aerodynamic optimization of unsteady turbulent flows*. PhD thesis, National Technical University of Athens, (in progress).

[191] A. Walther and A. Griewank. Getting started with ADOL-C. In *Combinatorial Scientific Computing*, pages 181–202. Chapman-Hall CRC Computational Science, 2012.

[192] J. Weiss, J. Maruszewski, and W. Smith. Implicit Solution of Preconditioned Navier-Stokes Equations using Algebraic Multigrid. *AIAA Journal*, 25:29–36, 1999.

[193] D. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, Incorporated, 1993.

[194] W. Yamazaki, K. Matsushima, and K. Nakahashi. Aerodynamic Design Optimization Using the Drag-Decomposition Method. *AIAA Journal*, 46(5):1096–1106, 2008.

[195] L. Yang and S. Nadarajah. Data Compression Algorithms for Adjoint-Based Sensitivity Studies of Unsteady Flows. Montreal, Quebec, Canada, July 15 - July 20 2018.

[196] Y-X. Yuan. Recent Advances in Trust Region Algorithms. *Mathematical Programming, Series A and B*, 151(1):249–281.

[197] T. Zervogiannis. *Optimization methods in aerodynamics and turbomachinery based on the adjoint technique, hybrid grids and the exact Hessian matrix.* PhD thesis, National Technical University of Athens, 2011.

[198] J. Zhang, LW. Xu, and RZ. Gao. Multi-island Genetic Algorithm Optimization of Suspension System. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 10, 11 2012.

[199] D. Zhao, Y. Wang, W. Cao, and P. Zhou. Optimization of Suction Control on an Airfoil Using Multi-island Genetic Algorithm. *Procedia Engineering*, 99:696 – 702, 2015.

[200] D. Zingg, M. Nemec, and T. Pulliam. A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization. *European Journal of Computational Mechanics*, 17(1-2):103–126, 2008.

[201] A.S. Zymaris. *Adjoint methods for the design of shapes with optimal aerodynamic performance in laminar and turbulent flows*. PhD thesis, National Technical University of Athens, 2010.

[202] A.S. Zymaris, D.I. Papadimitriou, K.C. Giannakoglou, and C. Othmer. Continuous adjoint approach to the spalart-allmaras turbulence model for incompressible flows. *Computers & Fluids*, 38(8):1528–1538, 2009.

[203] A.S. Zymaris, D.I. Papadimitriou, K.C. Giannakoglou, and C. Othmer. Adjoint wall functions: A new concept for use in aerodynamic shape optimization. *Journal of Computational Physics*, 229(13):5228–5245, 2010.