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

Shape Parameterization and Constrained Aerodynamic Optimization. Applications Including Turbomachines

PhD Thesis

Flavio Gagliardi

Supervisor: Kyriakos C. Giannakoglou
Professor NTUA

Athens, 2020

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

Shape Parameterization and Constrained Aerodynamic Optimization. Applications Including Turbomachines

PhD Thesis

Flavio Gagliardi

Athens, 2020

To my father. To my mother.

If I have seen further,
it is by standing on the shoulders of giants.
Isaac Newton

Abstract

Recently, with the continuous development of analysis tools and the growth of computational power, numerical optimization is increasingly being used to design new shapes with improved (aerodynamic/hydrodynamic) performance using Computational Fluid Dynamics (CFD) simulations. Since CFD simulations for complex problems are expensive, numerical optimization methods should perform as efficiently as possible. Recent progress with CFD adjoint solvers allows computing objective function gradients at a cost which does not scale with the number of Degrees of Freedom (DoFs), making gradient-based optimization a valuable option. At the same time, progress with gradient-free optimization methods (such as evolutionary algorithms) makes them attractive to globally explore design spaces and find improved shapes, sometimes according to more than one criterion. This thesis uses both gradient-based and gradient-free methods for shape optimization in both internal and external aerodynamics, including the design/optimization of blade rows.

A CFD-based shape optimization method involves several tools other than the CFD evaluation software and the search method itself. The parameterization and mesh displacement techniques as well as methods for handling constraints are among them. This thesis focuses on these tools by proposing new methods, as described below.

Shape parameterization is fundamental in all aerodynamic shape optimization problems. It determines the design space and the variety of reachable geometries; thus, it has a leading role in the cost and success of the optimization. In industrial workflows, in particular, shape optimization usually begins with a geometry obtained using Computer-Aided-Design (CAD) software. This starting geometry defines the DoFs of the case and their bounds, should the latter be necessary. In an industrial environment, an evident requirement is that the solutions found by running the optimization loop must also be compatible with the CAD software, at least to enable the manufacturing process. For this reason, this thesis focuses on CAD-compatible parameterization methods that are able to export a Boundary-Representation (B-Rep) model of the shape; B-Rep is a method to represent shapes largely used in CAD software.

Specifically, in the turbomachinery field, this dissertation uses and extends software for modeling turbomachinery blade rows. This is referred to as the Geometric Modeler for Turbomachinery (GMTurbo), and has been developed by and used in the Parallel CFD & Optimization Unit (PCOpt) of the National Technical University of Athens (NTUA); GMTurbo software was the outcome of a recently integrated PhD thesis at NTUA. In the present thesis, new features and tools that allow its integration into automatic, CAD-based, optimization loops are developed and tested. This includes the differentiation of the parameterization procedure to support adjoint-based optimizations, and software to import existing geometries into GMTurbo.

A new method, which will be referred to as B-Rep-Morpher, is proposed to support shape optimization by offering a free-form deformation environment for generic aerodynamic geometries which remain in standard B-Rep format. The proposed technique introduces a small number of "handles", strategically placed around or on the shapes to be optimized. Displacement vectors associated with these handles are used as the DoFs. Using the Radial Basis Function (RBF)-based interpolation method, these displacements are transferred from the handles to the Non-Uniform Rational B-Spline (NURBS) control points of the B-Rep model; the updated surface remains in B-Rep format and is, thus, exportable to a STandard for the Exchange of Product model data (STEP) file. The B-Rep-Morpher is differentiated since this is required in adjoint-based optimization.

Over and above performance criteria, shape optimizations are driven by constraints. Therefore, part of this thesis concerns constraint imposition, including turbomachinery design cases. Two families of constraints are considered. The first one comprises geometric constraints related, for instance, with the minimum thickness of the designed turbomachinery blades or the space necessary for mounting the blades on the casing. Some geometric constraints are merely satisfied by imposing appropriate bounds on the DoFs of the parameterization method, whereas others must be considered as non-linear constraints. For instance, this thesis illustrates a constraint parameterization technique to handle the space into the blade profiles that is necessary to cut the thread to mount it on the casing. The second family comprises performance constraints, such as the total pressure losses in a turbomachinery row. In this dissertation, gradient-based and gradient-free numerical techniques are used to solve constrained shape optimization problems. These are carried out using already available Sequential Quadratic Programming (SQP) methods and Evolutionary Algorithms (EAs). The latter implies the use of the Evolutionary Algorithm System (EASY) software developed by the PCOpt/NTUA.

A CAD-based shape optimization framework is developed, coupling the flow solver of PCOpt/NTUA and its adjoint, GMTurbo or B-Rep-Morpher and gradient-based and -free optimizers. Within this framework, the adaptation of an existing CFD mesh to the new boundaries of the computational domain is also essential. This facilitates the CFD evaluation of new shapes by avoiding costly and hardly automated re-meshing. To this end, this thesis proposes a method to update an existing surface mesh, associated with a shape parameterized as above. The displacement of the surface mesh nodes is used to adapt the CFD volume mesh. Mesh displacement based on RBF interpolation is used for this task; RBF interpolation is known for its ability to preserve the validity and quality of the mesh, even for large displacements, without being affected by mesh connectivity. However, in case of large meshes, such as those used in real-world CFD applications, RBF interpolation, in its standard formulation, becomes excessively expensive. Methods to accelerate the mesh displacement are investigated, and a new, two-step, RBF-based mesh displacement method is proposed. In this method, the Fast Multipole Method (FMM) and the Sparse Approximate Inverse (SPAI) preconditioner are used to reduce the computational cost of the RBF interpolation.

The programmed framework is applied to shape optimization cases to assess its performance. These cases include the optimization of the following shapes: *(a)* a double elbow duct, to minimize total pressure losses, using SQP; *(b)* a compressor stator blade, to minimize the deviation of the exit flow from the axial direction and the total pressure losses of the flow while imposing geometric constraints, using SQP and EAs; *(c)* a turbine stator to maximize the capacity and minimize the total pressure losses, using EAs; *(d)* an aircraft, to minimize the drag and maximize the lift, using EAs.

Journal and conference papers that reflect work done in this PhD thesis are listed below:

- Journal papers:
  - F. Gagliardi and K. C. Giannakoglou. ‘A Two–Step Radial Basis Function-Based CFD Mesh Displacement Tool’. In: *Advances in Engineering Software* 128 (2019), pp. 86–97.
  - F. Gagliardi and K. C. Giannakoglou. ‘RBF-Based Morphing of B-Rep Models for use in Aerodynamic Shape Optimization’. In: *Advances in Engineering Software* 138 (2019).
- Conference papers:
  - F. Gagliardi, K. T. Tsiakas and K. C. Giannakoglou. ‘A Two-Step Mesh Adaptation Tool Based on RBF with Application to Turbomachinery Optimization Loops’. In: *Evolutionary and Deterministic Methods for Design Optimization*

and Control with Applications to Industrial and Societal Problems. Springer, 2019, pp. 127–141.

- K. T. Tsiakas, F. Gagliardi, X. S. Trompoukis and K. C. Giannakoglou. ‘Shape Optimization of Turbomachinery Rows using a Parametric Blade Modeller and the Continuous Adjoint Method Running on GPUs’. In: *7th ECCOMAS Conference Proceedings*. 2016.

The research illustrated in this thesis was funded by the People Programme (ITN Marie Curie Actions) of the European Union’s H2020 Framework Programme (MSCA-ITN-2014-ETN) under REA Grant Agreement no. 642959 (IODA project). The author was an IODA (Industrial Optimal Design using Adjoint CFD) Early Stage Researcher (ESR).

Acknowledgments

I want to express gratitude to my supervisor, professor Kyriakos C. Giannakoglou, for guiding me during my doctoral studies. His passion and dedication to the work are a source of inspiration and his investment in time and effort has been essential for creating this work.

I am sincerely thankful to all my colleagues and friends at the PCOpt/LTT who created a cheerful and welcoming environment. Particularly to Konstantinos Samouchos, James Koch, Kostantinos Tsiakas, Xenofon Trompoukis, Varvara Asouti, Dimitrios Kapsoulis, Konstantinos Gkaragkounis, Ioannis Vryonis and Morteza Monfaredi. From them, I've received excellent support and mentoring. In detail, I wish to express appreciation to V. Asouti for the support regarding the EASY software and K. Tsiakas for the help with the PUMA and GMTurbo software and his feedback on research ideas. A special thank goes to the fellow PhD students J. Koch, K. Samouchos and C. Kapellos for the endless valuable conversations.

I also thank the fellow PhD students of the IODA project. By collaborating on the same research topics, we had the opportunity to share knowledge.

Another thank goes to peers who supported and inspired me even before starting the doctoral studies. Above all, I am grateful to my family and girlfriend, who made this journey possible.

Contents

Chapter 1

Introduction

Shape optimization requires coordinating a significant number of computational tools that must be developed, if non-existent, and integrated efficiently. This chapter focuses on shape parameterization and deformation methods, which are fundamental in the shape optimization process, as well as the integration of constraints into the same process. Tools needed by any shape optimization method are identified by distinguishing those used as a black-box in this thesis and those extended or developed. A more thoughtful discussion and literature survey regarding each of these tools are delegated to the following chapters.

1.1 Background

During the last decades, computer-based simulation techniques have been shaping product development. In fact, the aerodynamic performance of objects is determined by their shapes, and the exponential growth of computational power has led to the evolution and exploitation of computer-based shape optimization methods [1, §1]. Simulations allow evaluating performance, such as aerodynamic performance, of many design prototypes at an early stage of the development workflow, thus reducing the need to perform experiments and offering the possibility to identify potential problems with shapes in advance. Design optimization aims at computing alternative designs with improved performance.

Aerodynamic (hydrodynamic, too) simulations are based on CFD techniques and tools. The first computer applications modeling fluid flows appeared in the late 60s; since then, the scientific community and industry proceeded with the development of solvers for accurately modeling increasingly more complex flow features. The use of simulations is an alternative to the pure experimental way and allows new designs to be tested rapidly, at a reduced cost.

The design improvement process is iterative; CFD simulations are used repetitively based on the gradually obtained knowledge, to achieve an improved design. In detail, to improve the aerodynamic performance of shapes, complex flow features have to be considered, which may not be intuitive for a human. Besides, manual shape changes are labor-intensive and require deep expertise. It must also be taken into account that modern engineering design must handle increasing requirements such as performance, environmental impact and cost, which require large design spaces, namely a high number of DoFs. Satisfying these needs calls for design spaces to be explored systematically, which can only be accomplished through numerical optimization and computer-aided design of shapes. In fact, apart from the availability of CFD codes and high-performance computing, computer-based shape optimization has been enabled by the evolution of approaches to numerically and programmatically manipulate shapes, namely CAD, and the development of numerical optimization methods. Ever since the origin of CAD, halfway of the 20th century, this technology continuously improved industrial product development and production processes. What started with the use of spline-based modeling techniques for designing car bodies, turbine/compressor blades, aircraft fuselages and wings is now fundamental in a multitude of disciplines such as mechanical, aerospace and civil engineering. Emerging technologies such as 3D printing techniques further demonstrate the importance of CAD tools in next-generation product development and manufacturing.

1.2 CAD-Based Aerodynamic Shape Optimization

Figure 1.1 illustrates simplified CAD-based design optimization loops. The process starts by parameterizing a reference shape with a set of DoFs. The reference shape's performance is evaluated by running a CFD simulation. Based on the results, an optimization algorithm determines desirable search directions, to improve the current design in conformity to the shape parameterization technique. The necessary CFD mesh is generated or deformed according to the shape in each evaluation. This process is iterated until convergence of the given objective function or until the user-defined maximum number of optimization cycles be reached, and finally, the resulting improved shape is obtained. Some optimization algorithms require the gradient of the performance criteria with respect to (w.r.t.) the DoFs, to determine the search directions; this can be computed by combining the geometric sensitivities of the surface mesh nodes w.r.t. the DoFs and those of the performance criteria (objective functions) w.r.t. the surface mesh nodes. The former are computed by differentiating the shape parameterization software; the latter require the differentiation of the CFD flow solver, for instance through the adjoint technique. In an aerodynamic shape

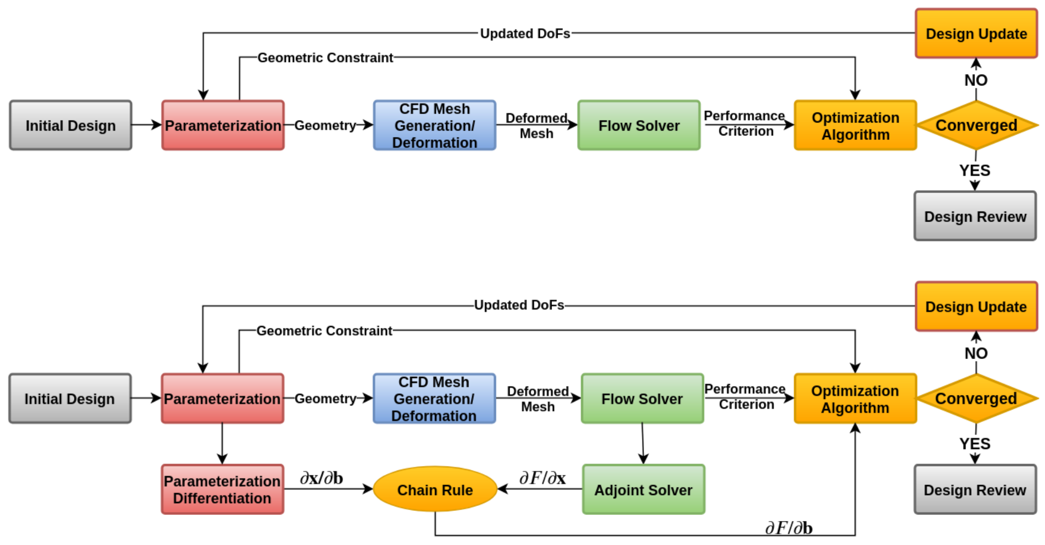


Figure 1.1: Schematic overview of two CAD- and CFD-based shape optimization loops using a mesh deformation strategy, a single performance criterion and considering geometric constraints. Colors identify the main constituents of the optimization loop: the parameterization (red), the mesh generation or deformation (blue), the CFD solver (green) and the optimization algorithm (yellow). Top: Gradient-free optimization workflow. Bottom: Gradient-based optimization workflow. $\delta\boldsymbol{x}/\delta\boldsymbol{b}$ are the sensitivities of the surface mesh nodes w.r.t. the DoFs. $\delta F/\delta\boldsymbol{x}$ are the sensitivities of the performance criterion (objective function) w.r.t. the surface mesh nodes. $\delta F/\delta\boldsymbol{b}$ are the sensitivities of the same quantity w.r.t. the DoFs.

optimization framework, adjoint solvers are developed to provide gradients at a cost that does not scale with the number of DoFs. In Figure 1.1, the essential components of a shape optimization workflow are identified, namely:

- the shape parameterization method and its differentiation, in case of gradient-based optimizations;
- the CFD mesh generation or deformation technique;
- the numerical optimization algorithm;
- the CFD flow solver and its adjoint, in the case of gradient-based optimization.

This section gives a brief introduction to each of the components needed by a typical optimization workflow, illustrating their roles and highlighting areas in which this thesis is contributing to.

Computational Fluid Dynamics. CFD is dealing with the computational analysis of systems involving fluid flow, heat transfer, and other associated phenomena [2]. CFD is applicable in a wide range of research and engineering problems, including aerodynamics and aerospace analysis, turbomachinery, weather simulation, environmental engineering, marine and biological applications, and engine and combustion analysis [3]. The past 20 years witnessed a significant increase in its use due to advantages such as its turnaround time and scalability [2]. Industry and academia widely use CFD to simulate flows and perform aerodynamic or hydrodynamic shape optimization.

The continuous adjoint method is helpful to compute the gradients of the performance criteria, cast in the form of objective functions, w.r.t. surface displacement, at a computational cost substantially independent of the number of DoFs. To this end, the adjoint system of equations corresponding to the governing fluid flow equations (a.k.a. state or primal equations) is formulated and solved.

This thesis does not contribute to the development of the CFD solver and its adjoint, which were both used as black-boxes. The Parallel-Unstructured-Multi-Row-Adjoint (PUMA) flow solver suite, developed by the PCOpt/NTUA, was used. This is briefly presented in Appendix E.

Numerical Optimization. Optimization problems are widely in use in many thematic areas such as computer science, engineering and economics; the development of solution methods for this kind of problems has been of interest in mathematics for centuries [4]. An optimization problem is formulated by first identifying one or more objectives, namely quantitative measures of the performance of the system under study, such as the total pressure losses of a compressor row or the lift or drag of a wing. The objective functions depend on specific characteristics of the system controlled by values, called design variables or DoFs. The goal of numerical optimization is to find the values of such variables that minimize (or maximize) the objective. Often the optimization problem is constrained: for instance, by limiting the values that the DoFs can assume or by some constraint functions that must take on values in specific ranges. Constraints play an essential role in finding meaningful optimal shapes. In CFD-based aerodynamic shape optimization, objective and constraint functions are typically (not always) related to the aerodynamic performance.

Many optimization algorithms exist: choosing the right algorithm is fundamental as it may determine how quickly and if a problem can be solved. Gradient-based methods use the gradient of the objective function and constraints w.r.t. the DoFs to find the minimum (or maximum) on the basin of attraction[1] of the user-supplied starting point. On the contrary, gradient-free methods are designed to find global or local optima without using derivative information and by methodically searching the solution space.

[1]The basin of attraction is the set of initial values that would lead the steepest descent algorithm to the same minimum.

Optimization problems are defined by their size, namely the number and characteristics of objective functions, constraints and DoFs. If the objective function and the constraints are linear functions of the DoFs, the problem is a Linear Programming (LP) problem. Quadratic Programming (QP) regards the minimization of a quadratic objective function with linear constrains. For both the LP and QP problems, reliable solution procedures are available [5, 6]. Non-Linear Programming (NP) problems are more challenging to solve: the objective function and constraints are non-linear functions of the DoFs. Solving NP problems generally requires iterative procedures in which search directions change in each iteration.

In this thesis, NP is used to optimize aerodynamic shapes, such as in Chapter 7, inside routines of the parameterization methods, such as in Chapter 2 and 3, and to handle geometric constraints, such as in Section 7.3.3. Among the used algorithms, there are the Quasi-Netwon and SQP methods, being gradient-based optimization techniques, and the EA methods, which are all used as black-box tools. In Appendix C, the Quasi-Netwon and SQP methods are reviewed. Regarding EA methods, the software suite EASY developed by the PCOpt/NTUA, which is illustrated in Appendix D, is used.

Shape Parameterization. Shapes and meshes involved in CFD-based shape optimization require parameterization; as the task that determines the DoFs, parameterization affects the computational cost and the success of the optimization [7].

Geometry description must often fulfill disparate requirements. For the optimization algorithm, it is crucial to have a parametric description that improves the ability to efficiently explore the design space, limiting the possible redundancies, interdependence and number of the DoFs, and avoiding geometric constraint violation. Moreover, if a gradient-based optimization algorithm is used, with the adjoint method computing the gradient of the objective function w.r.t. the surface displacement, the parameterization method must also provide geometric sensitivities, namely the gradient of surface displacement w.r.t. the DoFs. Through the chain rule, these yield the gradient of the objective function, w.r.t. the DoFs. For the CFD solver, however, the geometry description must have rigorous requirements of continuity and smoothness in order not to affect the quality of the numerical results. For the optimization workflow, the CFD mesh must be available for each shape variation reliably. Finally, from the industrial workflow point of view, the integration of the improved shapes in the design process is paramount.

One of the main tasks of this thesis is to investigate CAD-based parameterization methods. There are many parameterization approaches, and each of them has its pros and cons. In Chapter 2 and 3, the parameterization task is analyzed in-depth, a literature review of various parameterization methods is provided, and multiple alternatives are compared. CAD-based approaches are promising under many aspects, but criticalities must be solved.

Chapter 2 illustrates an original CAD-based morphing method developed and integrated into shape optimization. The tool is named B-Rep-Morpher. In detail, the process parameterizes the shape of B-Reps. The user defines a small number of “handles” to be placed around or on the B-Rep shapes to be optimized. Displacement vectors associated with these handles are used as DoFs during the optimization run. The technique transfers the displacements of the handles to the surface, effectively morphing it. This approach enables morphing surfaces directly in B-Rep format, retaining the ability to export the shape in a standard CAD format. The performance of the proposed method is assessed in aerodynamic shape optimization problems involving an airfoil, a duct, an aircraft model and a compressor stationary blade row.

In Chapter 3, a turbomachinery blade row shape parameterization tool developed by the PCOpt/NTUA [8] is extended. The extension includes the ability to build the geometry with NURBS and export it in standard CAD format. The already available procedure is based on design parameters that possess a clear physical or geometric meaning and corresponds to the established description of turbomachinery blade rows using the mean-camber-surface of the blade’s row and profile curves associated with it. The blade and casing geometry is built up with NURBS curves and surfaces, which ensures high smoothness of the resulting shape. NURBS curves are also used as input data, allowing for flexible control of the shape with a tunable number of DoFs. The tool is tested in the generation of blades of axial flow turbomachines. NURBS theory is briefly mentioned in Chapter 2 and deepened in Appendix A. To import existing blades geometries into the optimization workflow, a re-parameterization tool that performs the conversion from a B-Rep to the equivalent parameterized blade is developed.

Chapter 4 illustrates the necessary tools to integrate the aforementioned parameterizations into optimization loops. In detail, it presents a strategy for updating the CFD surface mesh to the already changed B-Rep-Morpher or GMTurbo models and the differentiation of parameterization tools to support gradient-based optimizations.

CFD Mesh Generation and Displacement. Mesh generation is one of the focal points in aerodynamic shape optimization. Generating a suitable mesh for complex shapes is time-consuming. It may require manual labor, especially when dealing with meshes for viscous flow simulations because it is necessary to simulate the boundary layer correctly.

Therefore, in shape optimization, the automatic generation or displacement of meshes according to shape variations is paramount. When using parameterization methods that deform the computational mesh directly, the problem of mesh displacement is alleviated: the parameterization method must still guarantee good quality deformed meshes. In this thesis, CAD-based parameterization is used: as previously discussed, Chapter 4 presents a method to adapt an existing surface mesh to CAD-based shape variations. Once the surface mesh is adapted to a new shape, the volume mesh also needs to be deformed: Chapter 5 proposes a method based on RBF to displace large meshes. Mesh displacement based on RBF interpolation is known for its ability to preserve the quality of the mesh, even for large displacements, without being dependent on mesh connectivity. However, in case of large meshes, such as those used in real-world CFD applications, RBF interpolation, in its standard formulation, becomes excessively expensive. Chapter 5 proposes a cost reduction technique for mesh displacement based on RBF by splitting the process into two steps and accelerating it through the preconditioning of the linear system (needed by the RBF method) based on geometric considerations and the FMM. The theory and performance of the FMM and linear solvers are illustrated in detail in Chapter 6 and Appendix F, respectively. Such a method and the programmed software are validated on three test cases related to the deformation of CFD meshes inside a duct, a turbine stator row and around a car model. The duct is optimized for total pressure losses using the SQP method and the turbine stator for capacity and total pressure losses using EAs. Third-party software tools, such as Pointwise [9] and AutoGrid5 [10], are used to generate meshes for all the CFD applications in this thesis.

Finally, Chapter 7 deals with the optimization of a compressor stator using both the parameterization method presented in Chapter 2 and 3, their differentiation presented in Chapter 4, the mesh displacement techniques proposed in Chapter 4 and 5 and gradient-based and -free optimization algorithms.

Chapter 2

The B-Rep-Morpher Parameterization Tool

Parameterization is a crucial component in shape optimization, as it determines the design space and, thus, it has an enormous influence on the optimization results. This chapter is dedicated to a shape parameterization method based on morphing that acts directly on NURBS-based B-Reps. The proposed technique requires the definition of a small number of "handles", which are strategically placed around or on the B-Rep shapes to be optimized. Displacement vectors associated with these handles are used as design variables during the optimization run. Using RBF as an interpolation method, these displacements are transferred from the handles to the NURBS control points of the B-Rep model; this approach offers the advantage that the updated surface remains in B-Rep format and is, thus, exportable to a STEP file. The proposed method comprises two successive deformation steps. Each deformation is controlled by an independent set of handles, increasing the flexibility of the morphing action. The performance of the proposed method is assessed in aerodynamic shape optimization problems involving an airfoil, a duct, an aircraft model and a compressor stator. An assessment of the proposed method based on Parametric Effectiveness (PE) is also included.

2.1 Literature Survey on Shape Parameterization

Surface parameterization techniques for use in shape optimization should:

- be able to handle complex, yet smooth, regular and realistic shapes and allow the imposition of geometric constraints;
- ensure a wide range of reachable shapes with the minimum number of design

variables because the complexity of shape optimization scales with this number [11];

- give rise to properly scaled design variables to facilitate the convergence of the optimization loop [12];
- be able to easily compute derivatives of surface nodal coordinates w.r.t. the design variables; in gradient-based optimization, this should be combined with the adjoint method that computes the gradients of the objective and constraint functions w.r.t. the surface nodal coordinates [13];
- offer the ability to deform the existing CFD surface mesh by adapting it to the new shape, which allows the displacement of the volume mesh to fit the new boundary, so as to avoid re-meshing;
- retain compatibility with the CAD software for further processing in industrial workflows.

Shape parameterization has been an area of continuous and extensive research; therefore, a vast amount of relevant literature can be found. A detailed survey of shape parameterization techniques for design optimization is provided in [14]; these techniques can be classified into mesh-based and CAD-based ones.

Mesh-based parameterization. Mesh-based parameterization relies upon either direct nodal deformations or space morphing techniques. The former perturb surface mesh nodes and give rise to the most extensive design space but require smoothing to avoid unrealistic designs or invalid meshes [15]. On the other hand, space morphing techniques deform the space in which meshes are embedded. They speed-up the optimization process by avoiding re-meshing and enable the continuation of new CFD simulations from a previous one on a different domain. Space morphing/deformation functions $\boldsymbol{d} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ assign a displacement to each point in space. Displacements imposed on a set of points (”handles”) determine the deformation field. The most common space deformation techniques, such as lattice-based Free Form Deformation (FFD) and RBF, are compared in [16]. FFD is a well-established deformation technique. The basic idea of FFD is based on embedding the object to be deformed in a parallelepiped lattice and deforming it using a trivariate tensor-product Bézier or B-spline function. RBF is a method for interpolating scattered data, in this case the displacement imposed on a set of scattered points; these are interpolated to the surface to be deformed. In [16], their comparison is conducted on shapes such as a car body and a pipe and is based on the following criteria: computational performance, numerical robustness, adaptivity, precision and quality of the deformation. The computational performance of the parameterization method is often negligible in shape optimization loops; nevertheless, [16] concludes that RBF and FFD require the same amount of time when using a similar number of handles. The robustness of a deformation method is quantified by considering the capability of the technique to tolerate defects in the input data such as low-quality mesh elements; again, [16] found that RBF and FFD had the same performance, which is justified by the fact that both are space deformation methods. The quality of the deformation includes several aspects; the most high-level one is that the deformation should be free of unexpected oscillations or artifacts. In this case, [16] observed that RBF has better results than FFD; in fact, FFD transformations are not easily predictable and have continuity problems in case the control lattice covers only a portion of the shape to be deformed. The adaptivity of a deformation method is quantified in terms of the capability of the technique to approximate a particular shape with an as low as possible number of DoFs. Again, RBF showed to be superior to FFD. The precision of a deformation method defines the accuracy in fulfilling positional constraints. RBF allows for the exact fulfillment of these constraints while FFD only for qualitative fulfillment. Moreover, RBF handles are placed arbitrarily in space (point-based), in contrast to lattice-based FFD, which handles coincide with the vertices of polyhedra. Finally, [16] recognizes the superiority of RBF over FFD on the analyzed cases.

Examples of RBF-based parameterization are presented in [17] and [18]. In [17], airfoil shapes in the discrete form are controlled by RBF handles placed off the shape. In [18], the RBF model is employed to parameterize and optimize sails trim by applying rigid displacements, such as rotation and translation, to sets of surface nodes. For instance, sail shapes are modified by assigning a rigid rotation about one sail axis while keeping the points on the other sail fixed; other transformations are also prescribed. The overall deformation is achieved by applying rigid transformations in cascade to the points of the surface mesh; then, their displacements are propagated to the interior mesh by RBF interpolation. RBF handles with a null displacement are added to a cylindrical surface around the sails so that the deformation induced by the RBF interpolation is limited in space.

Overall, mesh-based parameterizations succeed in building effective design spaces, even for complicated shapes; however, any link with the CAD model is lost during the optimization. Upon completion of the optimization loop, it is desirable to export the optimized shapes in a CAD format, though this unavoidably introduces approximation error and may compromise the obtained performance gain. During the optimization loop, handling shapes in other than CAD formats generally makes it harder to impose geometric constraints. For instance, keeping surfaces of revolution, such as the casing in turbomachinery applications, intact while morphing the blade surfaces requires handling the intersection of the blade with the casing or keeping these intersections fixed; both operations are readily available in CAD systems.

CAD-based parameterization. CAD-based methods can be classified as constructive, NURBS-based and space morphing. In constructive CAD tools, shapes are defined starting from standard geometric features, such as curves, extrusions and holes. This can be done using generic CAD packages or application-specific parameterization software. For instance, in turbomachinery design, it is common to have dedicated CAD kernels, which use spanwise and chordwise distributions of various geometric quantities (angles and thicknesses), usually in the form of polynomial or NURBS curves. The parameterization technique proposed in Chapter 3 belongs to the latter; in the same chapter, a more detailed literature overview of these methods is given. Constructive CAD tools generate geometries from scratch and, if necessary, can vary the topology of the model. They are defined by a sequence of instructions intrinsically accommodating geometric constraints, but they may over-constrain the designed shapes. Moreover, constructive CAD methods are not easily differentiable [19]. Pure NURBS-based methods may overcome this problem [20]. Such methods use NURBS control points' coordinates, contained in B-Rep models, to directly deform the shape, allowing for rich design spaces. It is challenging to generalize such methods to cases with many trimmed patches by keeping adjacent NURBS surfaces watertight and eventually tangent when control points are displaced. In [21], a technique for respecting geometric continuities over NURBS patch interfaces, which relies on the discrete filtering of the NURBS control point displacements, is presented; this provides a rich design space and shape derivatives. However, NURBS-based methods can be impractical in some industrial cases, the CAD models of which may contain thousands of NURBS control points. The third class of methods, namely space morphing techniques applied to B-rep models, displaces the NURBS control points indirectly; the parameterization technique proposed in Chapter 2 belongs to the latter. These introduce just a few design variables and provide smooth shape variations, even for CAD models with thousands of NURBS control points; however, these methods are not commonly found in the literature. An approach based on FFD, proposed in [22] for structural analysis, applies the same transformation to both the CAD model and the computational mesh. However, the resulting CAD model only approximates the corresponding mesh surface, and, in [22], the method is demonstrated only for small shape modifications. Commercial CAD packages, such as Rhino [23], allow using morphing boxes to alter a single NURBS surface. The software developed in [18] can approximately bring the shape of an RBF-morphed mesh back to CAD. In [24], a method for editing B-Rep solid models via direct push-pull modeling of patches is proposed. In this method, the user must specify the target location and orientation of the push-pulled faces. Then, the solid undergoes a boundary regeneration process by combining the new surface locations with the topology information. In this process, the solid model regeneration may fail due to geometry-topology inconsistency; [24] introduced a paradigm to compute valid solid models in such cases, by changing the topology. Despite its robustness, it cannot easily be used for shape optimization.

The re-parameterization of geometries available in standard B-Rep formats, such as STEP files [25, 26], in which shapes are represented by a collection of connected patches, is a key component of the method proposed in this chapter. The method is based on shape morphing techniques applied to these NURBS-based B-Rep models. Shape morphing relies on the RBF interpolation model and decouples the construction of the reference geometry from the parameterization to be used during the optimization; moreover, it can handle complex geometries with a user-defined number of parameters, which result in well-scaled design variables for use in shape optimization.

2.2 Shape-Morphing Strategy and Theoretical Background

Before the details of the proposed shape-morphing technique are presented in Section 2.3, Section 2.2.1 concisely summarizes the main phases of the parameterization procedure and its integration into shape optimization. They are based on two mathematical entities, NURBS and RBF, for which the concepts and notations used in this chapter can be found in Sections 2.2.2 and 2.2.3.

2.2.1 Shape-Morphing Strategy

Figure 2.1 illustrates the workflow defining the shape parameterization strategy based on morphing. Firstly, the reference shape is imported in the form of a B-Rep model, practically as a STEP file. Because STEP files might contain non-NURBS entities, these are converted to NURBS, making them compatible with what follows. Handles, the purpose of which is to deform the shapes, are then positioned on or around these NURBS patches. To increase morphing flexibility, a series of successive deformation actions, each corresponding to a set of handles, is necessary; according to them, shapes are incrementally deformed (multi-step deformations). The displacements of the handles are used as design variables during the optimization. Section 2.3 presents the details of the procedure for deforming sets of NURBS

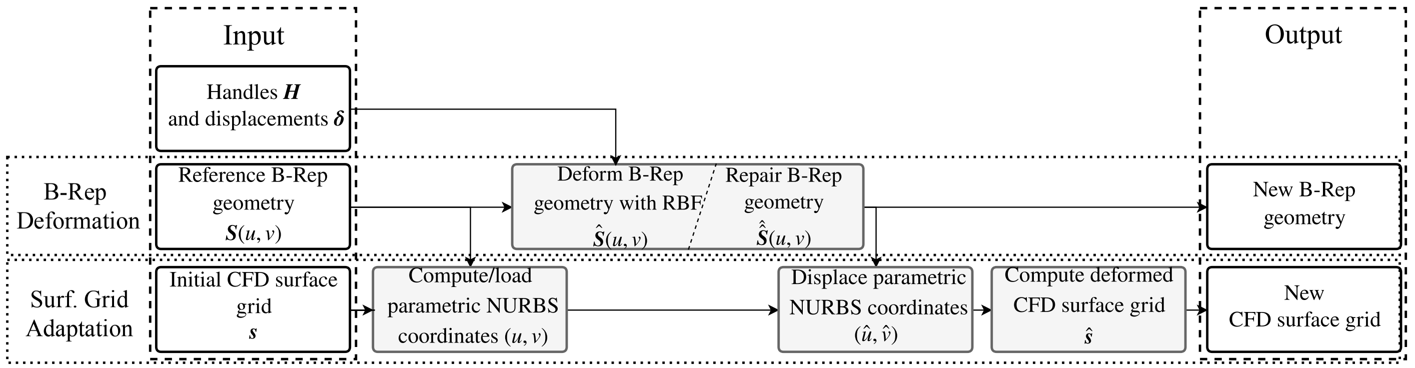

Input
Handles $\boldsymbol{H}$ and displacements $\boldsymbol{\delta}$
B-Rep Deformation
Reference B-Rep geometry $\boldsymbol{S}(u,v)$
Deform B-Rep geometry with RBF $\hat{\boldsymbol{S}}(u,v)$
Repair B-Rep geometry $\hat{\hat{\boldsymbol{S}}}(u,v)$
Surf. Grid Adaptation
Initial CFD surface grid $\boldsymbol{s}$
Compute/load parametric NURBS coordinates (u,v)
Displace parametric NURBS coordinates $(\hat{u},\hat{v})$
Compute deformed CFD surface grid $\hat{\boldsymbol{s}}$
Output
New B-Rep geometry
New CFD surface grid

Figure 2.1: Main phases of the parameterization procedure based on shape morphing and CFD surface mesh deformation. Within an optimization loop, the deformation of the volume CFD mesh follows. The multi-step B-Rep deformation procedure is described in Section 2.3. The surface mesh adaptation procedure is described in Section 4.1.

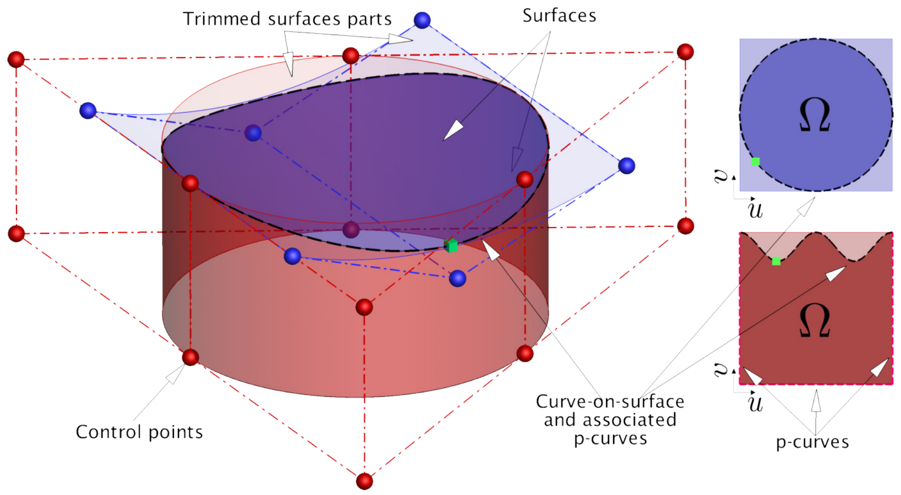


Figure 2.2: Left: Two intersecting/trimmed NURBS surfaces and bi-directional control nets. The shared curve (dashed) and trimmed parts of the surfaces, invisible in the CAD model, are illustrated. Right: Trimmed parametric domains of the two surfaces along with the dashed p-curves corresponding to the shared curve. The green point is one of the many test points controlling geometric continuities; see Section 2.3.2. It is represented on the curve-on-surface and the two p-curves.

surfaces, following the displacement of handles, while satisfying geometric continuities. This procedure results in modified B-Rep models to be exported as STEP files. Associated with the same task is the deformation of the CFD surface mesh of the reference shape by adapting it to the deformed B-Rep model: the algorithm is illustrated in detail in Chapter 4 and summarized as follows. To this end, the NURBS parametric coordinates of each surface mesh node are needed. If not available from the mesh generation software, these should be reconstructed by projecting mesh nodes onto the NURBS parametric space of the patches pertaining to the reference B-rep model. Once the shape has been morphed, the trimming-curves displacements of the NURBS surfaces in the parametric space drive the RBF-based displacement of the nodal parametric coordinates; the so-computed new nodal parametric coordinates determine the new positions of the surface mesh nodes. The last phase is the displacement of the 3D CFD mesh, for which the RBF-based method described in Chapter 5 is used.

2.2.2 NURBS and Relevant Issues

NURBS are analytical functions describing 2D or 3D curves up to complex 3D surfaces. Because of their flexibility, NURBS are frequently employed in CAD systems. In standard vendor-neutral file formats for exchanging information among CAD systems, such as the STEP file format, B-Rep primarily (though not exclusively, as noted above) relies upon

NURBS patches, for the purpose of portability. Using NURBS patches to represent shapes enables the exchange of information in standard CAD format.

A NURBS curve is a parametric curve mathematically defined as

$$\boldsymbol{C}(u) = \sum_{i=0}^{I} R_i(u)\boldsymbol{P}_i \quad , \tag{2.1}$$

where $\boldsymbol{P}_i$ are the control points forming a control polygon/net, $R_i(u)$ are the rational basis functions, and u is the parameter. If $\boldsymbol{P}_i \in \mathbb{R}^2$, $\boldsymbol{C}(u)$ is a curve on a plane else, if $\boldsymbol{P}_i \in \mathbb{R}^3$, $\boldsymbol{C}(u)$ is a curve in the 3D space. Similarly, a NURBS surface is a two-parameter (u and v) function, defined as

$$\boldsymbol{S}(u, v) = \sum_{i=0}^{I} \sum_{j=0}^{J} R_{i,j}(u, v)\boldsymbol{P}_{i,j} \quad . \tag{2.2}$$

In this case, the control points $\boldsymbol{P}_{i,j} \in \mathbb{R}^3$ form a bi-directional control net [27]. More information about NURBS theory is provided in Appendix A.

NURBS surfaces have the limitation of being four-sided patches; thus, to create complex shapes, many trimmed patches must be combined. Trimming is the operation that “cuts” surfaces using curves lying on them, increasing the flexibility in shape representation. For instance, a surface with a hole can be represented by a single trimmed surface. The trimmed-away portions of the surface are neither discarded nor rendered when showing the B-Rep model. Curves used for trimming surfaces are called curves-on-surface ($\boldsymbol{C}_{cs}$) or p-curves ($\boldsymbol{C}_{pc}$) if represented in the parametric space of surfaces sharing the 3D curve-on-surface. P-curves and curves-on-surface are also used to describe the natural edges of surfaces. The portion of the surface that is part of the B-Rep model is the parametric domain Ω, defined by a set of p-curves forming one or more loops in the parametric space of a surface, as illustrated in Figure 2.2, where the shared $\boldsymbol{C}_{cs}$ corresponds to two $\boldsymbol{C}_{pc}$ in the parametric domains of the two intersecting surfaces. Trimming is a crucial operation for building complex shapes. However, it may introduce gaps between intersecting NURBS patches; continuity across these interfaces (to a specified tolerance) is ensured by appropriate control points positioning rather than by making control points on the edges of the surfaces coincide. In a B-Rep model, $G0$ continuity does not require that adjacent NURBS surfaces share the same number or distribution of control points. Figure 2.2 illustrates a simple example in which the control points of a curve-on-surface do not coincide with those of the adjacent surface. Such configurations create geometric continuity issues when B-Rep models are changing during an optimization loop; a remedy to this is presented in Section 2.3.2.

Table 2.1: RBF activation functions.

Name	$\phi(r)$	Used for	See Section
Inverse multiquadric [28]	$\left(\left(\frac{r}{\sigma}\right)^2 + 1\right)^{-1/2}$	Shape morphing and 3D mesh displacement	2.3.1, 5.3.1
Wendland C2 [29]	$\begin{cases} \left(1 - \frac{r}{\sigma}\right)^4 \left(1 + 4\frac{r}{\sigma}\right) & \text{if } r < \sigma \\ 0 & \text{if } r \geq \sigma \end{cases}$	Shape morphing	2.3.1
Wendland C0 [29]	$\begin{cases} \left(1 - \frac{r}{\sigma}\right)^2 & \text{if } r < \sigma \\ 0 & \text{if } r \geq \sigma \end{cases}$	Continuity fixing and 3D mesh displacement	2.3.2, 5.3.2
Thin plate spline [30]	$r \log r^r$	Displacement of parametric coordinates	4.1

2.2.3 RBF-Based Interpolation and Relevant Issues

RBF interpolation is extremely versatile since it can interpolate values given at scattered points by returning the exact values at those points. In this study, quantities to be interpolated are the known 2D or 3D displacements defined at distinct source nodes, which are

- the handles of the (3D) parameterization in Section 2.3.1,
- NURBS control points (3D) in Section 2.3.2,
- NURBS parametric coordinates (2D) in Section 4.1,
- mesh nodes (3D) in Section 4.1 and Chapter 5.

Radial basis functions are real-valued functions $\phi : \mathbb{R} \rightarrow \mathbb{R}$ depending only on the distances of a point $\boldsymbol{x} \in \mathbb{R}^Q$ from the so-called RBF interpolation sources $\boldsymbol{x}_k \in \mathbb{R}^Q, k \in [1, K]$. The RBF deformation function $\boldsymbol{d} : \mathbb{R}^Q \rightarrow \mathbb{R}^Q$ takes the form

$$\boldsymbol{d}(\boldsymbol{x}) = \sum_{k=1}^{K} \boldsymbol{c}_k \phi(||\boldsymbol{x} - \boldsymbol{x}_k||), \tag{2.3}$$

where $||.||$ is the Euclidean distance, and the coefficients $\boldsymbol{c}_k \in \mathbb{R}^Q$ are computed so as to correctly reproduce the imposed displacements $\boldsymbol{d}(\boldsymbol{x}_k) = \boldsymbol{\delta}_k \in \mathbb{R}^Q, \ \forall k \in [1, K]$ at the source nodes; this requires the numerical solution of a $K \times K$ linear system. Because the method proposed in this chapter requires the solution of linear systems of small size, computational cost is not an issue, even in multi-step deformations.

The behavior of the interpolation is profoundly influenced by the chosen RBF activation function ϕ [31], with either local or global support. Activation functions used in this thesis are reported in Table 2.1; some of them depend on σ which also affects the interpolation [31]. In locally supported RBF activation functions, σ determines the region of influence of the RBF around each source node; this implies that $\phi(r) \neq 0$ if and only if $r < \sigma$. The displacement of the handles dictated by the optimization is interpolated at the NURBS control points using the RBF method, with the Wendland C2 function (if the user wishes to apply local deformations) or the inverse multiquadric function (for global ones). These functions are bounded ($|\phi(r)| \leq 1,\ \forall r \geq 0$) and strictly monotonically decreasing ($\phi(r_1) > \phi(r_2),\ \forall r_1 < r_2, r_1 \geq 0$); that is, the displacements interpolated at the NURBS control points fade away far from the handles, providing smooth and intuitive deformations (Section 2.3.1). Nodal displacements in the 2D parametric space of NURBS surfaces are computed using the σ-free thin plate spline function, which has excellent interpolation capabilities in spaces where deformations are known along the boundaries and interpolated at the internal nodes (Section 4.1). Local deformations of surface mesh nodes (Section 4.1) and NURBS control points (Section 2.3.2) are carried out with the Wendland C0 function. In RBF interpolations with global support, equation 2.3 is extended by additional polynomial terms to guarantee the exact affine displacement by respecting translation, rotation and scaling. These terms are not included in this chapter; the reader may find more about them in Section 5.2, in which additional aspects of RBF interpolations are analyzed, or [32].

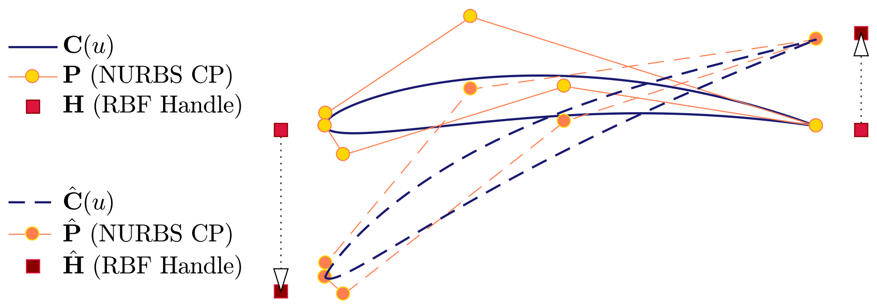
2.3 RBF-Based B-Rep Shape-Morphing Framework

In the proposed method, shapes described by or transformed into NURBS patches undergo RBF-based morphing, see Section 2.3.1. By doing so, geometric continuities at the interfaces of NURBS patches can be jeopardized. Section 2.3.2 proposes an approach for preserving positional ($G0$) and tangential ($G1$) continuity between surfaces. Hereafter, surfaces and curves of a B-Rep model are not indexed.

2.3.1 B-Rep Deformation Driven by Handles

During the optimization, shapes are modified by associating displacement vectors $\boldsymbol{\delta}_k$, $k \in [1, K]$ to a user-defined number of handles $\boldsymbol{H}_k$. These displacements are interpolated at the NURBS control points $\boldsymbol{P}_i$, $i \in [0, I]$ and $\boldsymbol{P}_{i,j}$, $(i, j) \in [I \times J]$ of curves $\boldsymbol{C}_{cs}(u)$ and surfaces $\boldsymbol{S}(u, v)$. Changed NURBS curves $\hat{\boldsymbol{C}}_{cs}(u)$ and surfaces $\hat{\boldsymbol{S}}(u, v)$ are created by means of the same rational basis functions $R_i(u)$ and $R_{i,j}(u, v)$ as the reference ones and displaced control points $\hat{\boldsymbol{P}}_i$ and $\hat{\boldsymbol{P}}_{i,j}$. The so-morphed NURBS curves and surfaces are computed as

$$\hat{\boldsymbol{C}}_{cs}(u) = \sum_{i=0}^{I} R_i(u) \underbrace{(\boldsymbol{P}_i + \boldsymbol{d}_M(\boldsymbol{P}_i))}_{\hat{\boldsymbol{P}}_i} \tag{2.4}$$



$\mathbf{C}(u)$
$\mathbf{P}$ (NURBS CP)
$\mathbf{H}$ (RBF Handle)
$\hat{\mathbf{C}}(u)$
$\hat{\mathbf{P}}$ (NURBS CP)
$\hat{\mathbf{H}}$ (RBF Handle)

Figure 2.3: Deformation of two NURBS curves describing a 2D airfoil shape. The NURBS curves alter by interpolating the displacements $\boldsymbol{\delta}$ of the RBF handles $\boldsymbol{H}$ to the control points $\boldsymbol{P}$.

and

$$\hat{\boldsymbol{S}}(u,v) = \sum_{i=0}^{I}\sum_{j=0}^{J} R_{i,j}(u,v)\underbrace{\left(\boldsymbol{P}_{i,j} + \boldsymbol{d}_M(\boldsymbol{P}_{i,j})\right)}_{\hat{\boldsymbol{P}}_{i,j}} , \tag{2.5}$$

where $\boldsymbol{d}_M(.)$ is an RBF deformation operator (equation 2.3) trained on the displacements $\boldsymbol{\delta}_n$ of $\boldsymbol{H}_k$, $k \in [1, K]$. A 2D example is illustrated in Figure 2.3: two handles are used to deform an airfoil shape described by two NURBS curves, separately for its pressure and suction sides, which together form the B-Rep model.

In shape morphing, a frequent requirement is to keep certain parts of the overall shape fixed. Two possible ways to fulfill this requirement are proposed. In the first, although the entire geometry is considered deformable, fixed handles coinciding with the NURBS control points, describing the portion of the shape that should remain intact, are inserted. This option must be used if a smooth transition from deformable to fixed parts is required. For example, this approach is used in Section 2.4.2 to keep the inlet and outlet sections of a duct fixed while deforming its central part, ensuring a smooth transition in between. The other option is to deform only a subset of the NURBS surfaces in the B-rep model. After the deformation action, the p-curves at the intersection of deformed and undeformed surfaces are updated by re-trimming. This allows the displacement of the intersection lines over the fixed surfaces while maintaining the validity of the CAD model. This approach is used in Section 2.4.3 to redesign an aircraft wing while keeping its fuselage fixed and, in Section 5.5.4 and Chapter 7, to keep the same axisymmetric hub and shroud surfaces of a stationary blade-row while deforming the blade itself. Updated p-curves in the deformed B-Rep model are denoted by $\hat{\boldsymbol{C}}_{pc}$.

The RBF interpolation offers flexibility in shape modifications, allowing handles to be

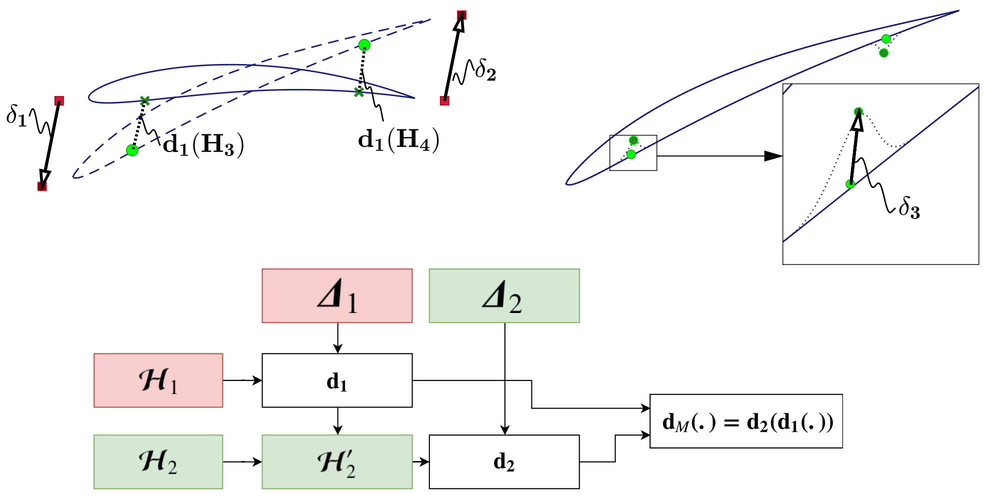


Figure 2.4: Two-step deformation of an airfoil shape. $\boldsymbol{\Delta}_1 = [\boldsymbol{\delta}_1, \boldsymbol{\delta}_2]^\top$ are the displacements of the set of handles $\boldsymbol{\mathcal{H}}_1 = [\boldsymbol{H}_1, \boldsymbol{H}_2]^\top$ which are active in the first step (red). $\boldsymbol{\Delta}_2 = [\boldsymbol{\delta}_3, \boldsymbol{\delta}_4]^\top$ are the displacements of the set of handles $\boldsymbol{\mathcal{H}}_2 = [\boldsymbol{H}_3, \boldsymbol{H}_4]^\top$ which are active in the second step (green). Top-left: The first step, in which active handles are marked in red and passive in green. New active handles are used to train the deformation function $\boldsymbol{d}_1 : \mathbb{R}^2 \to \mathbb{R}^2$, which is used to displace the passive handles from positions $\boldsymbol{\mathcal{H}}_2$ to $\boldsymbol{\mathcal{H}}_2'$ together with the shape. Top-right: The second step, in which green handles become active. Active handles are used to train the deformation function $\boldsymbol{d}_2 : \mathbb{R}^2 \to \mathbb{R}^2$ to further displace the shape deformed by $\boldsymbol{d}_1$. Bottom: Schematics for the construction of the two-step deformation function $\boldsymbol{d}_M : \mathbb{R}^2 \to \mathbb{R}^2$.

placed in arbitrary locations. For even higher flexibility, multi-step deformations[1] can be used: these are performed one after the other to deform surfaces incrementally. In each step, different sets of handles are used. Using more than one step makes it possible to perform more complex shape modifications, defining multiple sets of handles with increasing levels of control. Practically, the use of a few handles producing low-frequency deformations and many handles making high-frequency ones is recommended. In most cases, this gives rise to a two-step deformation approach. However, the proposed method and the programmed software support any number of steps. By solving the RBF problems (one for each step) sequentially, it is possible to use various activation functions and σ values. It is recommended that handles responsible for low-frequency deformations be placed at a distance from the shape; just a few of them are usually enough. Activation functions associated with these handles should be of global support. In contrast, handles responsible for high-frequency deformations should be placed on or next to the shape surface, in areas where higher deformation resolution is needed. The corresponding activation functions should be of local support. Depending on the application, users can choose whether handles

[1]"Multi-step deformations" denotes a sequence of distinct deformations and not the division of a single deformation into many steps.

move along the axes of the Cartesian or cylindrical coordinate system.

One of the metrics that can be used to measure the performance of the proposed method is the Parametric Effectiveness (PE) [33, 34]. PE is used to compute the degree to which the proposed method alters the search space compared to an optimization controlling NURBS control point positions directly. PE is the ratio of the maximum objective function improvement that can be achieved using the parameterization in hands to the one that could be achieved if surface nodal points were free to move independently; both must be computed with the same root-mean-squared boundary displacement. PE requires derivatives computed from an available adjoint run [35]. The definition of PE is provided in Appendix G.

Unlike single-step RBF-based deformation, in which all handles interact with each other, the use of various sets of handles allows successive modifications over the same part of the shape. Each step allows deforming the shape with a different geometric modeling paradigm. However, to maintain the modeling paradigm imposed by the user at each step, sets of handles must be displaced hierarchically. In each step, a distinction is made between active and passive sets of handles. The former is a single set used to train the RBF deformation function of the current step. The latter includes all sets that will become active in the subsequent steps; their position is passively displaced by the RBF deformation functions of all previous steps. More precisely, to perform an S-step deformation, handles must be organized in S sets $\mathcal{H}_s$, $s \in [1, S]$:

$$\begin{gathered}\mathcal{H}_1 = [\boldsymbol{H}_1, \cdots, \boldsymbol{H}_{n_1}]^\mathsf{T}, \ \ \mathcal{H}_2 = [\boldsymbol{H}_{n_1+1}, \cdots, \boldsymbol{H}_{n_2}]^\mathsf{T}, \ \ \cdots, \\ \mathcal{H}_S = [\boldsymbol{H}_{n_{S-1}+1}, \cdots, \boldsymbol{H}_N]^\mathsf{T}, 1 \le n_1 < \cdots < n_{S-1} < n_S \equiv N\,.\end{gathered} \tag{2.6}$$

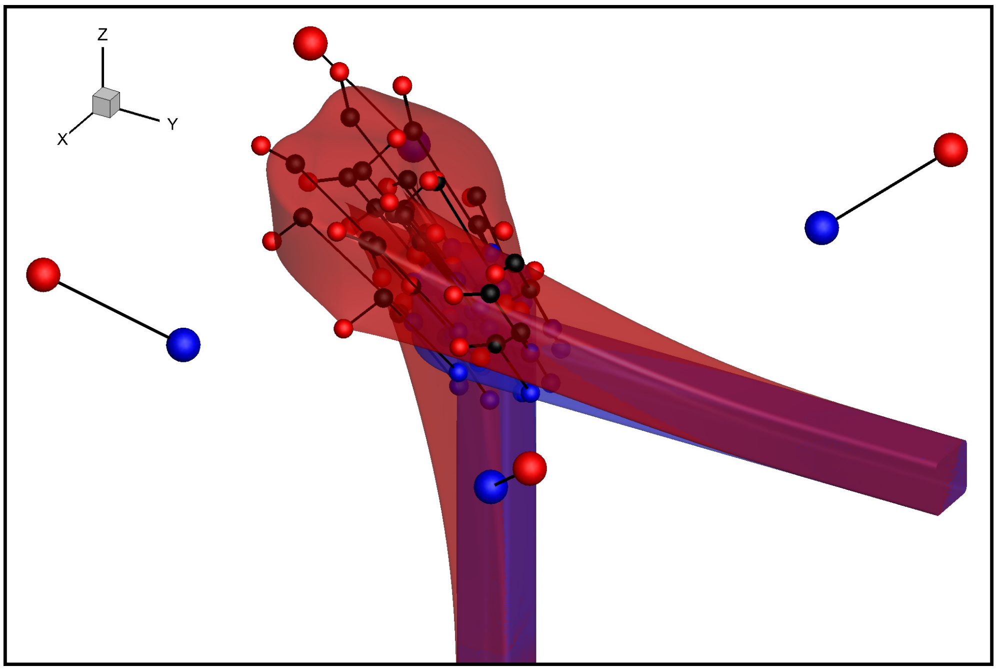
For an S-step deformation, the operator $\boldsymbol{d}_M$ in equations 2.4 and 2.5 includes the deformation functions of all steps, namely $\boldsymbol{d}_M(\boldsymbol{x}) = (\boldsymbol{d}_S \circ \boldsymbol{d}_{S-1} \circ ... \circ \boldsymbol{d}_1)(\boldsymbol{x})$ for S steps, where the s^{th} deformation function is trained on the set

$$\mathcal{H}'_s = \begin{bmatrix} \boldsymbol{H}_{n_{s-1}+1} + (\boldsymbol{d}_{s-1} \circ \cdots \circ \boldsymbol{d}_1 \circ \boldsymbol{d}_0)(\boldsymbol{H}_{n_{s-1}+1}) \\ \vdots \\ \boldsymbol{H}_{n_s-1} + (\boldsymbol{d}_{s-1} \circ \cdots \circ \boldsymbol{d}_1 \circ \boldsymbol{d}_0)(\boldsymbol{H}_{n_s-1}) \\ \boldsymbol{H}_{n_s} + (\boldsymbol{d}_{s-1} \circ \cdots \circ \boldsymbol{d}_1 \circ \boldsymbol{d}_0)(\boldsymbol{H}_{n_s}) \end{bmatrix}, \tag{2.7}$$

for the set of displacements $\boldsymbol{\Delta}_s = \{\boldsymbol{\delta}_n, \ n \in [n_{s-1} + 1, n_s]\}$; $\boldsymbol{d}_0(\boldsymbol{x})$ is the null deformation function ($\boldsymbol{x} = \boldsymbol{d}_0(\boldsymbol{x})$).

This process is better explained using the following pair of examples. The first one concerns the arbitrary deformation of an airfoil. In Figure 2.4, the first deformation step makes low-frequency shape modifications and, also, displaces the position of the handles of the second step. The second step makes high-frequency shape modifications. In the second step, handles placed close to the surface remain close to it even after the first-step deformation, maintaining the same modeling paradigm prescribed on the reference shape. Figure 2.4 also illustrates the composition of a two-step deformation function. The second example, shown in Figure 2.5, is concerned with the shape optimization of the double elbow duct further analyzed in Section 2.4.2. The duct shape is altered by displacing two sets of handles corresponding to two deformation steps. In the first step, the four active handles deform the duct shape and are used to displace the passive handles of the second step (from the "blue" to the "black" position). In the second step, the second set of handles becomes active and additionally deforms the shape through its displacement.

In the 2D airfoil case of Figure 2.3, after the deformation, $C0$ continuity at the leading and trailing edges is guaranteed because the first and last control points of the two NURBS curves coincide. In fact, if two control points $\boldsymbol{P}_\alpha$ and $\boldsymbol{P}_\beta$ coincide before the transformation ($\boldsymbol{P}_\alpha \equiv \boldsymbol{P}_\beta$), this will be the case for the deformed shape, too, because of $\boldsymbol{d}_M(\boldsymbol{P}_\alpha) \equiv \boldsymbol{d}_M(\boldsymbol{P}_\beta)$. However, in 3D CAD models, this is generally not the case, and positional and tangential ($G0$ and $G1$) continuity between surfaces must be recovered after the RBF-based deformation by repositioning some NURBS surface control points. For example, any shape-morphing action applied to a CAD model, such as the DPW6 wing-body configuration, Figure 2.6, would break the $G0$ continuity of some surfaces, rendering the CAD model invalid. The DPW6 is an aerodynamic model representative of a modern transonic commercial aircraft with the CAD model available [36]. The wing geometry is defined by 9 NURBS patches, containing 34,765 control points overall, whereas the half of the fuselage is formed by 23 patches, containing 134,128 control points overall. Figure 2.6 illustrates the shape-morphing action based on a two-step deformation. In the first step, one handle, which is allowed to move in the xz-plane, controls the overall shape of the wing. In the second step, handles placed close to the leading and trailing edges control the wing locally through their displacements in the z-direction; displacements in the x-direction are not allowed in order to keep the axial chord of the wing constant. The fuselage shape is excluded from the morphing actions and its shape remains intact. The deformation action smoothly changes the shape of the wing but creates gaps, such as those at the narrow trailing edge surface or over the small surface at the tip of the wing, which renders the deformed CAD model temporarily invalid, Figure 2.7. In Section 2.3.2, a procedure for repairing discontinuities of these kinds is proposed.



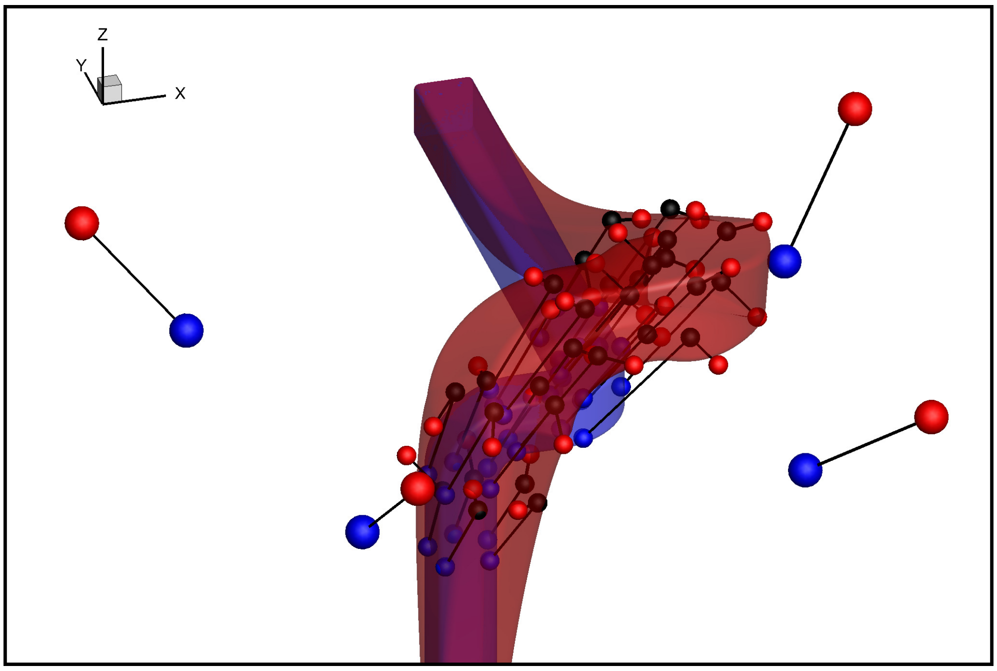


Figure 2.5: Double elbow duct. Reference (blue) and optimized (red) geometry for minimum total pressure losses. The shape is altered by displacing the handles (spheres) using a two-step approach. The first step is based on 4 handles (large spheres), whereas the second on 30 handles (smaller spheres). Handles are shown in their initial (blue) and final (red) positions. Moreover, the intermediate (black) positions of the handles of the second step are shown. These are computed after the deformation action of the first step. Inlet and outlet sections are kept intact by fixed handles (not shown), coinciding with the NURBS control points.

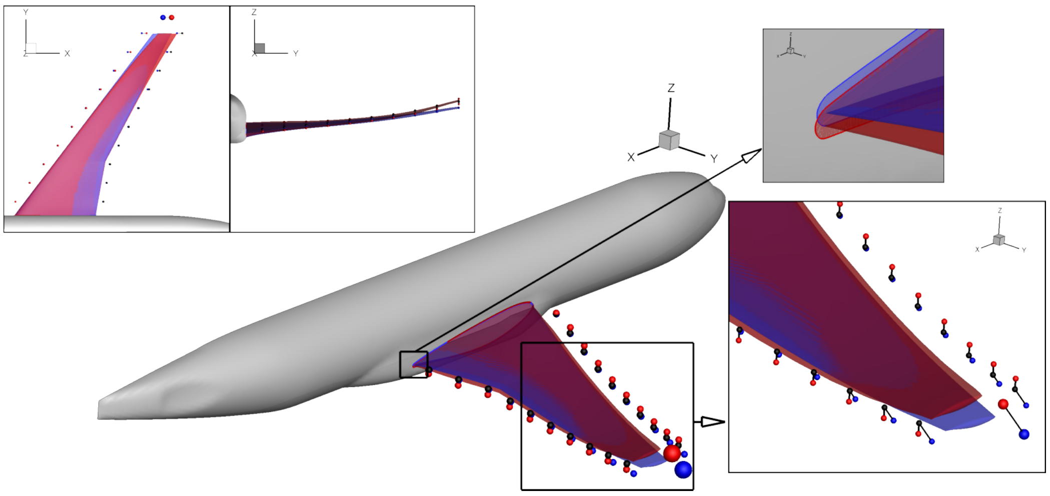


Figure 2.6: DPW6 aircraft. Shape deformation. The fuselage shape remains intact while the wing is deformed. The initial shape of the wing and the initial handle positions are shown in blue, the deformed wing and the final handle positions in red. The deformation action is carried out in two steps; the black spheres show the positions of the handles at the beginning of the second step.

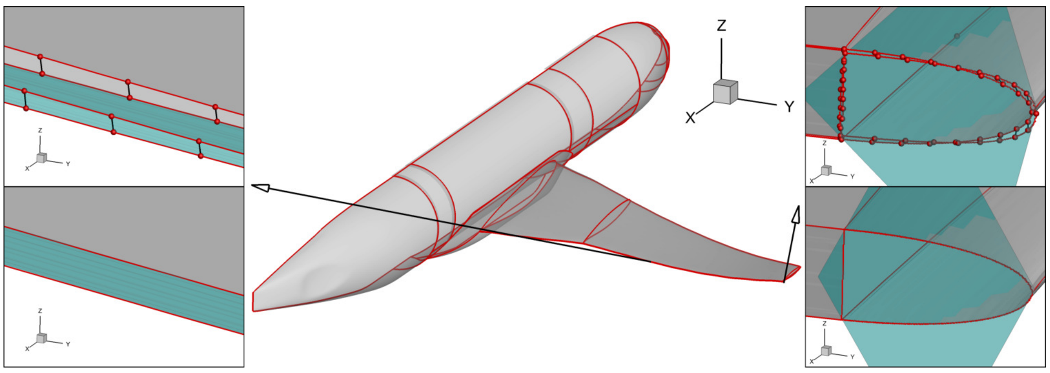


Figure 2.7: DPW6 aircraft. $G0$ continuity fixing. The RBF-based displacements of the NURBS control points create gaps at joints, violating $G0$ continuity. The surfaces are sewed by minimizing the squared distances between test points, here illustrated as red spheres placed along the edges of the CAD model. Top: Close-up view of a narrow surface at the wing trailing edge, top-left, and a small surface at the wing-tip, top-right, prior to the fix. Bottom: Same close-up views after the fix.

2.3.2 Geometric Continuity Correction

As mentioned above, during the RBF deformation, NURBS control points are displaced in a way that generally violates continuity between neighboring NURBS patches in the B-rep model, which needs to be corrected. $G0$ continuity is ensured by "sewing" the p-curves $\hat{\boldsymbol{C}}_{pc}(u)$ of the deformed surfaces to the corresponding deformed curves-on-surface $\hat{\boldsymbol{C}}_{cs}(u)$ by modifying the surfaces. This approach requires an evaluation of the $G0$ continuity at test points distributed on the curves-on-surface. Figure 2.2 shows a single test point on a curve-on-surface and the corresponding p-curves of adjacent surfaces; Figure 2.7 illustrates some test points in the Cartesian space for the DPW6 B-rep model. These test points $u_{cs,t}$, $t \in [1, N_T]$ are distributed in the parametric space of the curves-on-surface $\boldsymbol{C}_{cs}$ of the reference B-Rep model, and the corresponding parameters $u_{pc,t}$ of each p-curve $\boldsymbol{C}_{pc}$ in the corresponding surface $\boldsymbol{S}$ are computed by solving

$$\min_{u_{pc,t}} \left\| \boldsymbol{S}(\underbrace{u_t, v_t}_{\boldsymbol{C}_{pc}\left(u_{pc,t}\right)}) - \boldsymbol{C}_{cs}\left(u_{cs,t}\right) \right\| . \tag{2.8}$$

Such a minimization problem is solved with Newton's Method (Appendix C). Multiple ways to compute (by equations 2.1 and 2.2) the same 3D point on the surface of the model, up to a tolerance, are established; in fact, for all test points, $\boldsymbol{C}_{cs}\left(u_{cs,t}\right) \approx \boldsymbol{S}\left(\boldsymbol{C}_{pc}\left(u_{pc,t}\right)\right)$. After the B-Rep model is deformed, $\hat{\boldsymbol{C}}_{cs}\left(u_{cs,t}\right)$ and $\hat{\boldsymbol{S}}\left(\hat{\boldsymbol{C}}_{pc}\left(u_{pc,t}\right)\right)$ no longer coincide. In order to fix it, a minimization problem is set up to reduce the distance between the points computed through the deformed curves-on-surface $\hat{\boldsymbol{C}}_{cs}\left(u_{cs,t}\right)$ and p-curves $\hat{\boldsymbol{S}}\left(\hat{\boldsymbol{C}}_{pc}\left(u_{pc,t}\right)\right)$. This minimization problem uses a subset of the control points $\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}} := [\forall\, \hat{\boldsymbol{P}}_{i,j} : (i,j) \in \boldsymbol{\Gamma}]$ in each surface $\hat{\boldsymbol{S}}$ as DoFs. Namely, the minimization problem is defined for each surface as

$$\begin{aligned} \min_{\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}}} C_{G0}(\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}}) = \min_{\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}}} \sum_{t=1}^{N_T} \Bigg\| \, \hat{\boldsymbol{S}}\left(\hat{\boldsymbol{C}}_{pc}\left(u_{pc,t}\right)\right) - \hat{\boldsymbol{C}}_{cs}\left(u_{cs,t}\right) + \\ + \sum_{(i,j)\in\boldsymbol{\Gamma}} R_{i,j}\left(\hat{\boldsymbol{C}}_{pc}\left(u_{pc,t}\right)\right) \Delta\hat{\boldsymbol{P}}_{i,j} \Bigg\|^2 , \end{aligned} \tag{2.9}$$

where $\hat{\boldsymbol{S}}\left(\hat{\boldsymbol{C}}_{pc}\left(u_{pc,t}\right)\right)$ is the point computed on the deformed surface, $\hat{\boldsymbol{C}}_{cs}\left(u_{cs,t}\right)$ is the point computed on the deformed curve and $\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}} := [\forall\, \Delta\hat{\boldsymbol{P}}_{i,j} : (i,j) \in \boldsymbol{\Gamma}]$ are the corrections to the surface control points identified by $\boldsymbol{\Gamma}$. $\boldsymbol{\Gamma}$ is the set of indices defined as

$$\boldsymbol{\Gamma} = \bigcup_{t=1}^{N_T} \Gamma_t \,,\; \Gamma_t = \forall (i,j) \in [I \times J] : \frac{R_{i,j}(u_t, v_t)}{\sum_{l=0}^{I} \sum_{m=0}^{J} R_{l,m}(u_t, v_t)} \geq \bar{R}_t \,, \tag{2.10}$$

where $\bar{R}_t$ is a threshold defined by averaging all non-zero $R_{i,j}(u_t, v_t)$. It is necessary to modify only the control points identified by $\boldsymbol{\Gamma}$, instead of the full set $(i, j) \in [I \times J]$, in order to well pose the minimization problem of equation 2.9. Due to the local support property of NURBS, many control points might play no role in minimizing the cost function $C_{G0}(\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}})$. To avoid singularities in the Hessian matrix derived from C_{G0}, these control points must not be included in the minimization problem. Excluding control points that have little influence also improves robustness and convergence rate. Test points are automatically arranged on the curves-on-surface based on distance, curvature and knot-distributions criteria. Since NURBS are polynomial expressions, testing the G-continuities of two NURBS entities at a finite number of points is sufficient to guarantee continuity along the entire curve-on-surface. The proposed method based on the minimization of $C_{G0}(\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}})$ overcomes over-sampling, with the only extra burden being the increased computational cost for computing C_{G0}; however, the number of DoFs $(\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}})$ of the cost function is independent of the number of test points. When computing the new surface according to the corrections $\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}}$, since these corrections are applied only to the control points identified by $\boldsymbol{\Gamma}$ in each NURBS surface, the resulting shape might be irregular; that is, some rows of control points might move while others remain unaffected. An additional procedure is, therefore, applied to regularize the deformation induced by the corrections. The displacements $\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}} := [\forall\ \boldsymbol{\Delta}\hat{\boldsymbol{P}}_{i,j} : (i, j) \in \boldsymbol{\Gamma}]$ computed by minimizing $C_{G0}(\boldsymbol{\Delta}\hat{\boldsymbol{P}}_{\boldsymbol{\Gamma}})$ are propagated to the rest of the control points by the RBF interpolation. The locally supported RBF activation function used in this phase is the Wendland C0 (Table 2.1) with σ chosen to be four times the maximum displacement norm, $\sigma = 4 \max_{(i,j)\in\boldsymbol{\Gamma}} \left\|\Delta\hat{\boldsymbol{P}}_{i,j}\right\|$, so as to smoothly propagate the deformation while retaining a local effect. The deformed $G0$-continuous surfaces are given by

$$\hat{\hat{\boldsymbol{S}}}(u, v) = \sum_{i=0}^{I} \sum_{j=0}^{J} R_{i,j}(u, v) \left(\hat{\boldsymbol{P}}_i + \boldsymbol{d}_C(\hat{\boldsymbol{P}}_{i,j})\right) , \tag{2.11}$$

where $\boldsymbol{d}_C$ is the interpolation operator (C stands for correction). As mentioned above, the minimization problem of equation 2.9 is solved using Newton's method. The latter requires the computation of the Hessian matrix. The gradient of C_{G0} w.r.t. the displacement of a control point $\Delta\hat{\boldsymbol{P}}_{l,k}$ is:

$$\frac{\partial C_{G0}}{\partial \Delta\hat{\boldsymbol{P}}_{l,k}} = 2 \sum_{t}^{N_T} R_{l,k}(u_t, v_t) \left| \hat{\boldsymbol{S}}\,(u_t, v_t) - \hat{\boldsymbol{C}}_{cs}\,(u_{cs,t}) + \sum_{(i,j)\in\boldsymbol{\Gamma}} R_{i,j}(u_t, v_t) \Delta\hat{\boldsymbol{P}}_{i,j} \right| . \tag{2.12}$$

The second derivatives of the cost function C_{G0} w.r.t. two corrections $\Delta\hat{\boldsymbol{P}}_{i,j}$ and $\Delta\hat{\boldsymbol{P}}_{l,k}$ are

$$\frac{\partial^2 C_{G0}}{\partial\Delta\hat{\boldsymbol{P}}_{i,j}\partial\Delta\hat{\boldsymbol{P}}_{l,k}} = \begin{bmatrix} 2\sum_t^{N_T} R_{i,j}(u_t,v_t)R_{l,k}(u_t,v_t) & 0 & 0 \\ 0 & 2\sum_t^{N_T} R_{i,j}(u_t,v_t)R_{l,k}(u_t,v_t) & 0 \\ 0 & 0 & 2\sum_t^{N_T} R_{i,j}(u_t,v_t)R_{l,k}(u_t,v_t) \end{bmatrix}. \tag{2.13}$$

Since the Hessian matrix is constant (i.e., it does not depend on the NURBS control point positions), and many of its entries are zero due to the local support property of the NURBS, the Hessian matrix for each NURBS surface is inverted once with a sparse decomposition [37]. Similarly, the computationally expensive rational basis functions that appear in the objective function and its first derivative are computed just once and re-used, since they do not change during the minimization of C_{G0}. In fact, by pre-computing these quantities, it is possible to reduce the time needed to perform a Newton iteration by as much as two orders of magnitude. Moreover, these minimization problems, being independent of each other, can be solved in parallel. In some B-Rep models, the distribution of control points does not allow the optimization to reduce the C_{G0} objective function sufficiently to make the CAD model valid. In fact, a curve that contains too few control points cannot be sewed to another curve effectively due to the limitation in shape flexibility. In this case, additional knots are inserted into the reference B-rep model.

$G1$ continuity is corrected by solving a constrained minimization problem: on condition that $G0$ continuity is satisfied, $G1$ continuity requires that adjacent NURBS surfaces share the same tangent plane along the shared curve-on-surface. For two adjacent surfaces $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ that are $G0$ continuous at the points $(u_t,v_t)_{S1}$ on $\boldsymbol{S}_1$ and $(u_t,v_t)_{S2}$ on $\boldsymbol{S}_2$ (i.e. $\boldsymbol{S}_1\left((u_t,v_t)_{S1}\right) \cong \boldsymbol{S}_2\left((u_t,v_t)_{S2}\right)$), the $G1$ continuity condition at such test points is expressed as

$$\det\left(\left.\frac{\partial \boldsymbol{S}_1(u,v)}{\partial u}\right|_{(u_t,v_t)_{S1}}, \left.\frac{\partial \boldsymbol{S}_1(u,v)}{\partial v}\right|_{(u_t,v_t)_{S1}}, \left.\frac{\partial \boldsymbol{S}_2(u,v)}{\partial u}\right|_{(u_t,v_t)_{S2}} \right) = 0\,, \tag{2.14}$$

which is a measure of the coplanarity of the tangential vectors computed at the test points on the two surfaces.

From equation 2.14, a differentiable cost function can easily be derived as the summation of the determinants of all test points shared by two or more surfaces and minimized by displacing the surface NURBS control points similarly to the $G0$ correction case. Whereas

$G0$ continuity correction requires solving an independent unconstrained problem for each surface, $G1$ continuity calls for the solution of a constrained optimization problem involving all surfaces; the cost function for the latter involves the displacement of the control points of all deformable surfaces of the B-Rep model. However, the RBF deformation function is implicitly smooth [38], and in many cases, $G1$ continuity does not need to be fixed. The Sequential Quadratic Programming (SQP) method is used to handle the gradient-based constrained optimization with many design variables. The corrective deformation, which ensures $G1$ continuity, is interpolated with the RBF method at all NURBS control points.

2.4 Method Demonstration in CFD Shape Optimizations

This section demonstrates the application of the proposed method and the programmed software in three shape optimization case studies. First, the method is used in a 2D case, to (re-)parameterize an airfoil by also computing its effect on the search space through the computation of the PE. Then, the method contributes to the gradient-based optimization of a double elbow duct and a multi-objective EA-based optimization of an aircraft wing. All CFD simulations are performed using the PUMA solver, employing the Spalart-Allmaras turbulence model (Appendix E). Here, the emphasis (within the optimization loop) is on the capabilities of the B-Rep-Morpher. However, to perform the optimization task, additional tools that are presented in some of the next chapters are also used. In detail, the differentiation of the parameterization tool is presented in Chapter 4, as well as the method to adapt the surface mesh to the changed shapes. Then the volume meshes are displaced with the two-step RBF-based mesh displacement software developed in this thesis and described in Chapter 5. EA-based optimizations are performed using the in-house software EASY (Appendix D), whereas gradient-based optimizations rely upon the SQP (Appendix C) and the adjoint methods (Appendix E). Although the proposed method and software support an arbitrary number of deformation steps, all applications presented herein make use of a two-step deformation approach.

2.4.1 RAE-2822 Airfoil Optimization

Several studies on the RAE 2822 airfoil can be found [39]. The free-stream flow conditions correspond to test case 6 of [40] with Mach number equal to 0.725, flow angle 2.92° and Reynolds number 6.50×10^6. The quantity to be minimized is the drag coefficient C_D. Inequality constraints are imposed on the lift coefficient $C_L \geq {C_L}^0$, the pitching moment coefficient $C_m \leq {C_m}^0$ and the airfoil area $A \geq A^0$. ${C_L}^0$, ${C_m}^0$ and A^0 are the corresponding

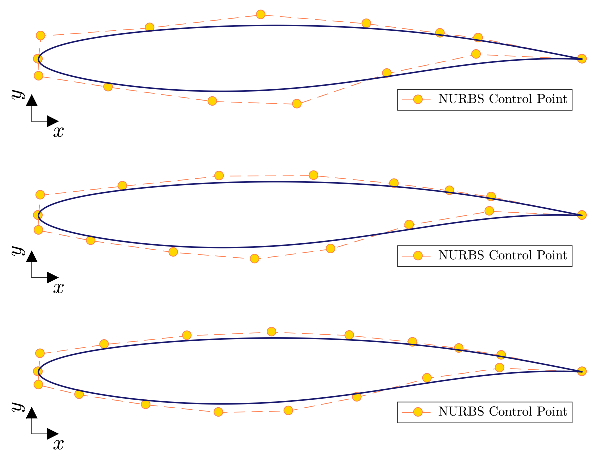


Figure 2.8: RAE 2822 airfoil. The NURBS-based parameterizations introducing the NURBS control points ordinates as DoFs are illustrated. During the optimization, the control points corresponding to the leading and trailing edges are kept fixed. From top to bottom, the airfoil is parameterized with 12, 14 and 16 DoFs.

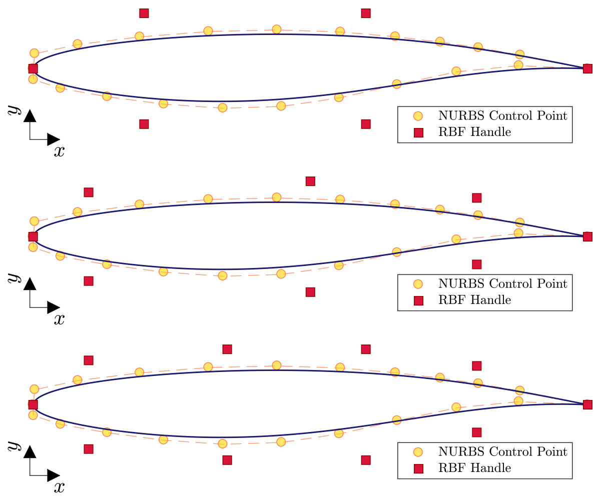


Figure 2.9: RAE 2822 airfoil. The RBF-based parameterizations introducing the RBF handles displacements in the y-direction as DoFs are illustrated. The RBF handles displacements control the position of the NURBS control points. During the optimization, the handles corresponding to the leading and trailing edges are kept fixed. From top to bottom, the airfoil is parameterized with 4, 6 and 8 DoFs.

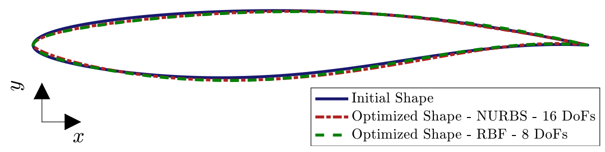


Figure 2.10: RAE 2822 airfoil. Original shape compared to the ones with improved drag obtained from different shape optimizations employing the NURBS control point positions and RBF handles displacements as DoFs. Differences in the shapes are barely visible.

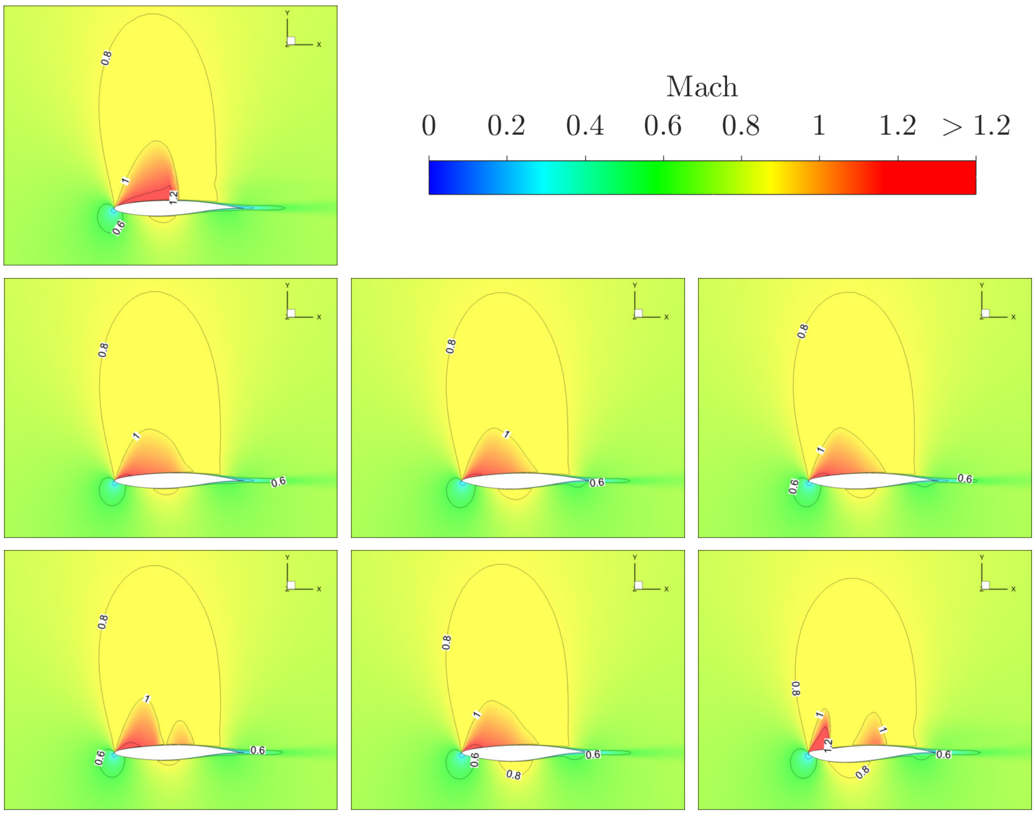


Figure 2.11: RAE 2822 airfoil. Mach number fields around the initial and optimized airfoils. Top-Left: the initial shape. Middle: shapes optimized with the RBF-based parameterization; from left to right, the airfoil is parameterized with 4, 6 and 8 DoFs respectively. Bottom: shapes optimized with the NURBS-based parameterization; from left to right, the airfoil is parameterized with 12, 14 and 16 DoFs respectively.

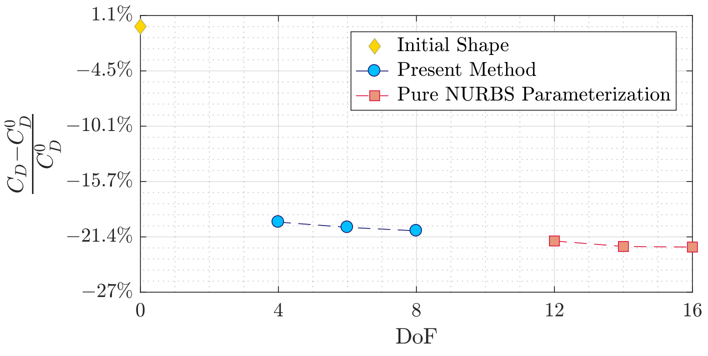


Figure 2.12: RAE 2822 airfoil. C_D improvement obtained by optimizing the shape of the airfoil with different numbers of DoFs and two different parameterization methods. Constraints are imposed on C_L, C_m and A. C_D^0 is the drag coefficient of the initial shape.

values of the initial shape. Leading and trailing edges are kept fixed, which results in constant axial chord length. The gradient-based optimization is conducted using an SQP-based algorithm by also bounding the DoFs. The continuous adjoint and the NURBS derivatives are used to compute the gradients.

Lift, drag and pitching moment coefficients are, respectively, defined as

$$C_L = \frac{2F_{\mathrm{L}}}{\rho u^2 S}\ ,\ C_D = \frac{2F_{\mathrm{D}}}{\rho u^2 S}\ ,\ C_m = \frac{M}{qS\,c}\ , \tag{2.15}$$

where F_{L} and F_{D} are the lift and drag forces, M the pitching moment, u the speed of the airfoil relative to the fluid, S the reference surface area, q the dynamic pressure and c the airfoil chord-length.

The purpose here is to assess the quality of shapes computed by the proposed parameterization method compared to the outcome of an optimization with the NURBS control points' coordinates as DoFs. To this end, the parametric effectivenesses (Appendix G) in either parameterization are also computed. In detail, the two parameterization methods to be compared, with different numbers of DoFs each, are:

- Three airfoil parameterizations based on NURBS curves with varying numbers of control points each. NURBS curves for the pressure and suction sides with 8, 9 or 10 control points are used. During the optimization, only 6, 7 or 8 control points per curve, respectively, are allowed to move in the y-direction yielding 12,14 or 16 DoFs in total. These are illustrated in Figure 2.8.
- Three parameterizations based on RBF handles which control the NURBS curves. The pressure and suction sides are approximated using NURBS curves with 10 control points each; their positions are controlled by 6, 8 or 10 handles placed around the airfoil, Figure 2.9. During the optimization, 4, 6 or 8 handles, respectively, move in the y-direction; thus, the optimization is conducted with 4, 6, or 8 DoFs. These are illustrated in Figure 2.9. The inverse multiquadric activation function with global support is used.

The PE of the NURBS-based parameterization with 16 DoFs is equal to 0.65 for the baseline shape. The use of the proposed method with 8 DoFs yields a PE value of 0.55. Even if the parameterization based on handles reduces the number of DoFs by 50%, the PE is, however, decreased by just 16%. Figure 2.12 illustrates the C_D that the two parameterization methods with different numbers of DoFs achieve. As expected, by increasing the DoFs, slightly lower C_D values result. However, the reduction in the number of DoFs by the proposed method implies just a small decrease in the capabilities of the

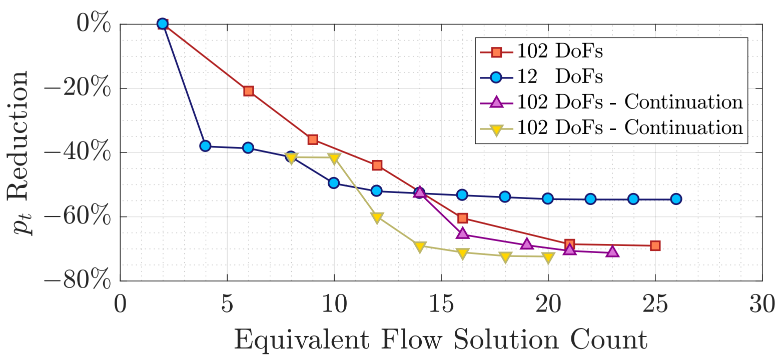


Figure 2.13: Double elbow duct. Convergence history in terms of equivalent flow solution counts (the gradient computation cost is about that of one flow solution) of four SQP-based optimizations using various numbers of DoFs. One is conducted by using 12 DoFs (first deformation step) and the other by using 102 DoFs (first and second steps). Two more convergence histories are obtained by suspending the optimization with 12 DoFs and re-starting it with 102 DoFs, by retaining the values of the first 12 DoFs; these are referred to as "Continuation".

optimization. The proposed parameterization finds designs that reduce C_D up to ~20%. Comparatively, the pure NURBS parameterization finds designs that reduce C_D up to ~22% but with significantly more DoFs. To conclude, the parameterizations based on NURBS curves controlled by RBF handles can reach an almost similar C_D reduction and shape with a reduced number of DoFs, compared to that based purely on NURBS, for the case under consideration. Figure 2.10 illustrates some of the improved shapes compared to the initial one: differences are small. Figure 2.11 illustrates the Mach number field for the initial and optimized shapes. A shock wave appears over the suction side of the initial airfoil. On the optimized airfoils, different shock wave patterns appear due to the different shapes; thus, different values of C_L and C_m are computed. However, it can be seen that, for all the optimized shapes, the shock intensity is significantly reduced and, consequently, the drag induced by the shock becomes smaller.

2.4.2 Double Elbow Duct Optimization

The double elbow duct B-Rep model is composed of a single untrimmed NURBS surface with 2,128 control points. The reference shape and parameterization are illustrated in Figure 2.5. The shape-morphing action is based on a two-step deformation. In the first step, 4 handles placed off the surface control the entire shape, whereas the second step uses 30 handles on the surface, with local support. Each handle is allowed to move in all three Cartesian directions, thus contributing three DoFs to the optimization. Therefore, the total number of DoFs is $3(30 + 4) = 102$. Fixed handles are placed at the inlet and outlet, in

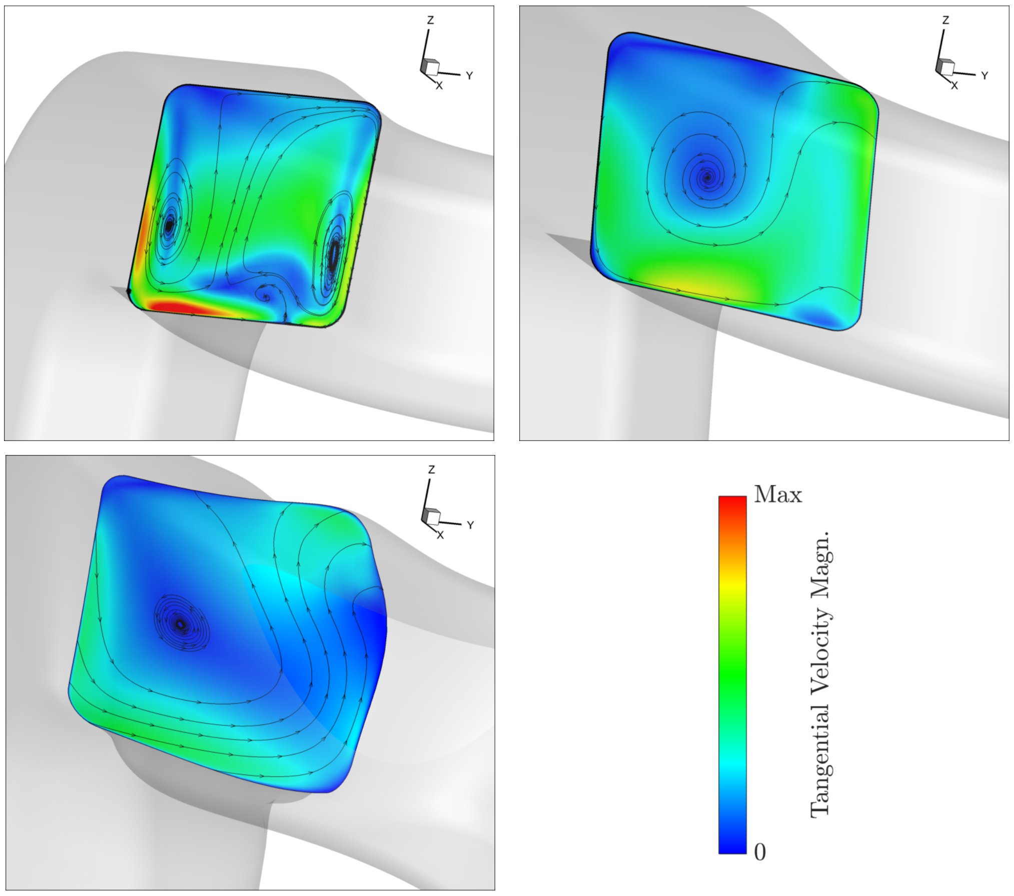


Figure 2.14: Double elbow duct. View of the secondary flow structure at a cross-section between the two elbows for the reference shape (left) and the ones optimized by employing 12 DoFs (middle) and 102 DoFs (right). Contours represent the normalized velocity magnitude over the cross-section plane.

positions corresponding to the NURBS control points, to constrain the displacement of these sections of the duct. The gradient-based shape optimization is conducted to reduce the mass-averaged total pressure losses Δp_t of the duct defined as

$$\Delta p_t = \frac{-\int_{S_I} p_t \rho \boldsymbol{V}\cdot\boldsymbol{n}\, dS - \int_{S_O} p_t \rho \boldsymbol{V}\cdot\boldsymbol{n}\, dS}{\int_{S_I} (p_t - p)\rho \boldsymbol{V}\cdot\boldsymbol{n}\, dS}, \tag{2.16}$$

where p is the pressure, p_t the total pressure, ρ the density, $\boldsymbol{V}$ the velocity, $\boldsymbol{n}$ the outward unit vector normal to the surface and S_O and S_I the duct outlet and inlet, respectively. The fluid is incompressible, and the Reynolds number based on the hydraulic diameter of the inlet is equal to 10^6. Figure 4.8 depicts geometric sensitivities for the displacement of a handle along the z-direction. Figure 2.13 compares the optimization convergence histories by using different numbers of DoFs for the parameterization. The convergence of the optimization conducted using a “one-shot” run with all 102 DoFs simultaneously is compared to that obtained by starting with a low-dimensional design space allowing low-frequency shape changes (first step) and, then, adding the DoFs which are responsible for high-frequency shape modifications (second step). It appears that a small number of DoFs offers a great gain in the optimization in the first cycles. In contrast, using all handles simultaneously from the beginning leads to more extended SQP line searches and higher computational cost. Specifically, the SQP-based algorithm is able to reduce the total pressure losses w.r.t. the starting shape by almost 40% using 12 DoFs, at half the cost of the first SQP cycle employing 102 DoFs, which enables a reduction of “just” 20%. However, after a few cycles, the latter can make better design improvements by exploiting the greater number of DoFs. The best convergence rate is obtained by using a sequential optimization strategy in which only a few DoFs are initially used for a few SQP cycles and the optimization continues with a larger number of DoFs. This strategy also results in a slightly better design.

Figure 2.5 illustrates the improved geometries computed by the proposed parameterization method and compares them with the reference geometry. Figure 2.14 presents the secondary flow between the two elbows for the reference shape and the optimal ones, resulting from optimizations conducted using 12 and 102 DoFs. The first deformation step changed the duct shape mainly by increasing its cross-sectional area, thus slowing down the flow, which reduces the wall shear stress and diffusion losses. The increased length between the two elbows and changes in their curvature contribute to the suppression of secondary flows and the reduction of centripetal forces required to turn the flow, hence reducing diffusion losses and areas with flow separation. The second deformation step introduces more local control and performs local changes; this results to less intense secondary flow patterns and lower total pressure losses.

2.4.3 DPW2 Aircraft Model Optimization

In this section, aerodynamic shape optimization is conducted for the DPW2 wing-body configuration [41]. The DPW2 common research model is representative of a commercial aircraft. The wing geometry is defined by 13 NURBS patches, containing 3,638 control points overall; half of the fuselage is formed by 11 patches, containing 4,577 control points overall. Figure 2.16 illustrates the wing parameterization adopted, which is similar to the one illustrated in Figure 2.6 for the DPW6 test case. That is, in the first step, one handle is allowed to move in the XZ-plane and controls the overall shape of the wing. In the

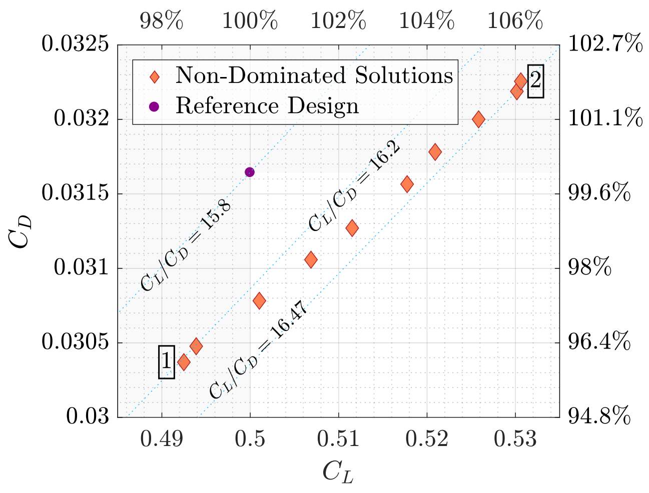


Figure 2.15: DPW2 aircraft. Results of the EA-based optimization for maximum lift and minimum drag. The wing shapes corresponding to the two Pareto points marked with integers (1 and 2) are compared to the reference design illustrated in Figure 2.16.

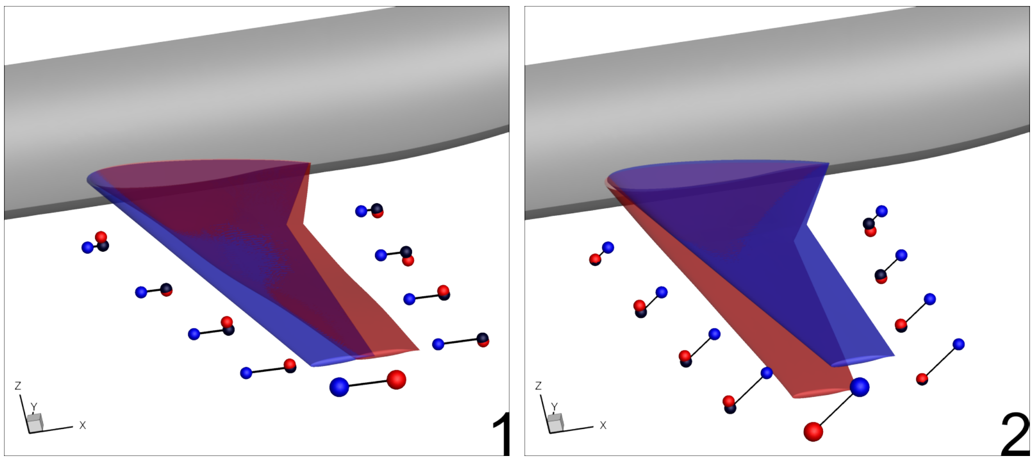


Figure 2.16: DPW2 aircraft. Two selected designs (red) from the front on non-dominated solutions in Figure 2.15 are shown in comparison to the reference design (blue), with the corresponding handle displacements. The fuselage shape remains intact (gray). The shape-morphing action is based on a two-step deformation: the single handle of the first step appears as the largest sphere.

second step, handles placed close to the leading and trailing edges control the shape locally by moving in the Z direction, whereas the fuselage shape is kept intact by excluding the corresponding surfaces from the morphing action. The intersection between fuselage and wing is updated by re-trimming. The two-objective shape optimization aims at minimum drag and maximum lift at an infinite Mach number of 0.75, Reynolds number of 3×10^6 and $0°$ infinite flow angle. The optimization is carried out with EASY; in detail, with a (5, 10) Metamodel-Assisted EA (MAEA) using RBF metamodels. Figure 2.15 illustrates the Pareto front for the two objective functions, found at the cost of 250 CFD evaluations; Figure 2.17 illustrates the pressure distributions on the surface of some of these designs and the refrence one. On the Pareto front, a solution that increases the C_L/C_D ratio by 3.0%, improving both lift and drag, can be identified. Another design improves the C_L/C_D ratio by 4.3%; however, in this case, C_L increases at the expense of C_D. [42] was able to reduce C_D by 2.6% by keeping the C_L equal to that of the initial geometry by varying the angle of attack. A gradient-based method and a CAD-free lattice-based FFD parameterization for the wing and wing-body junction were used with 96 DoFs overall. A similar improvement is achieved with the proposed parameterization, yielding $\approx$ 2.75% reduction in C_D while keeping C_L equal to that of the reference geometry, though with just 12 DoFs.

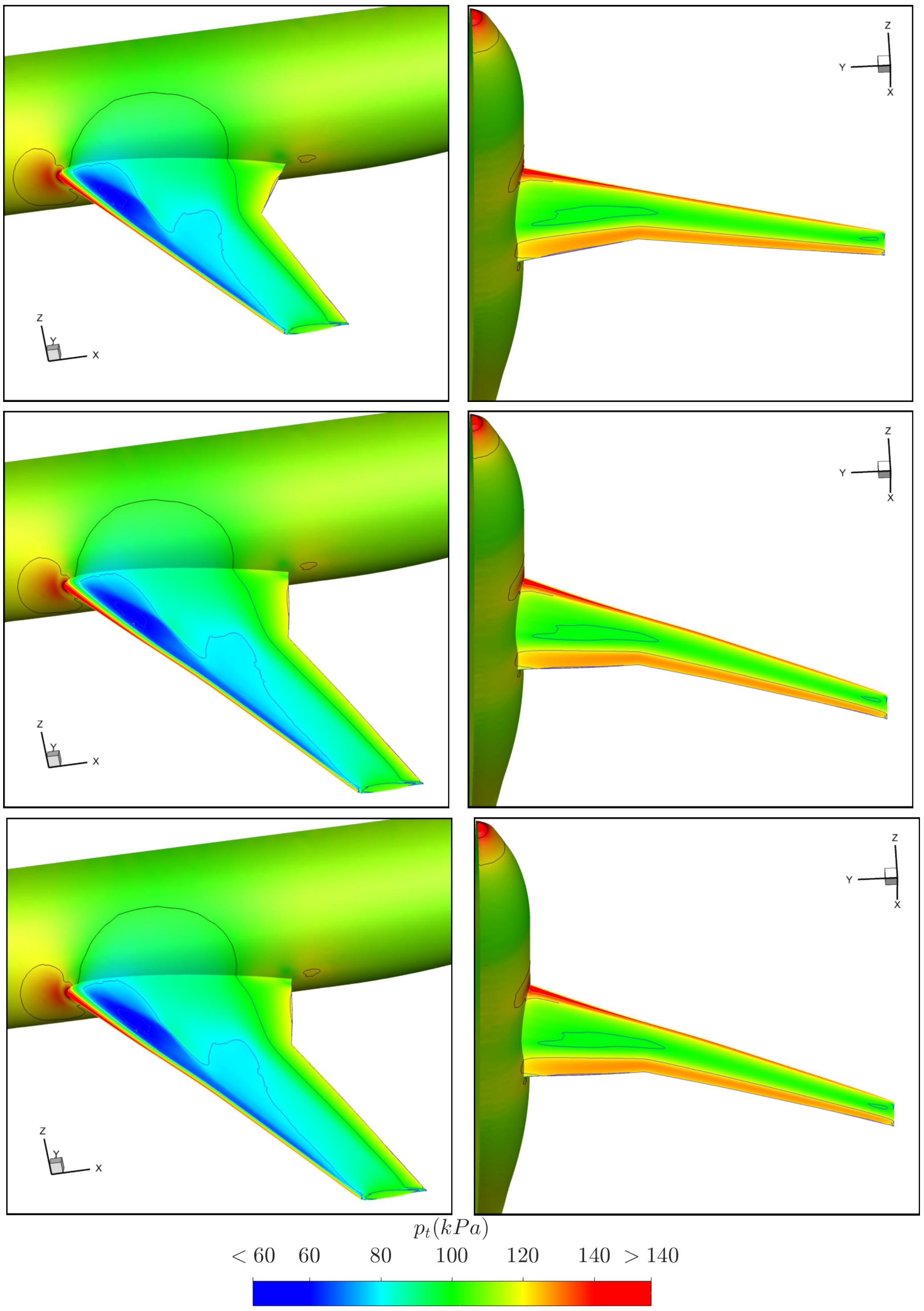


Figure 2.17: DPW2 aircraft. Total pressure distribution on the surface of three selected designs from Figure 2.15. Top: reference design. Middle: design marked with 1 in the above-mentioned figure. Bottom: design marked with 2.

Chapter 3

The GMTurbo Parameterization Tool

This chapter is concerned with the parameterization of blades with a dedicated blade parameterization method that exploits fundamental notions of turbomachinery. This method is programmed at and used by the PCOpt/NTUA (its standard version has been developed in the context of the PhD thesis of K. Tsiakas [8] and extended in this thesis) and will be referred to as the GMTurbo parameterization software. The procedure is based on design parameters that possess a clear physical or geometric meaning and corresponds to the established construction of turbomachinery rows using the mean-camber-surface of the blade and profile curves associated with it. GMTurbo is here extended by building up the blade and casing geometry with NURBS curves and surfaces, which assures high smoothness of the resulting shape. NURBS curves are also used as input data, allowing for a flexible control of shapes with a tunable number of DoFs. NURBS patches are exported to other CAD or meshing software via standard exchange files. The tool is tested in the generation of blades of axial flow turbomachines. To import existing blade geometries into the optimization workflow, a re-parameterization tool that performs the conversion from a B-Rep representation to a GMTurbo-parameterized blade is herein developed.

3.1 Literature Survey on Turbomachinery Parameterization

Compressor and turbine blades are designed and customized for each specific application in products. Hence, methods to design and optimize the shapes of blades, taking into account various performance parameters, are essential. Commonly used CAD packages provide tools for the parameterization of complex geometries, such as surface and solid modeling. However, these tools are not conveniently applicable to turbomachinery design and CFD-based shape optimization. Designers favor working with tools which involve design parameters with aerodynamic and physical meanings. For example, the tangent direction of the mean-camber-line or -surface at the leading edge of a blade, which is related to the velocity triangles of the fluid flow, is one of the favored parameters. At the same time, the shapes created by the design method must be compatible with existing CAD technologies in order for them to be included in manufacturing workflows and be compatible with external mesh generation tools and analysis software. Therefore, the parameterization tool must be equipped with the ability to export the shape in standard CAD formats, such as B-Reps in STEP or IGES format.

Turbomachinery row parameterization methods have been developed since the advent of CAD technology. For instance, [43] relies on conformal mapping to produce airfoils in 2D that correspond to airfoils lying on surfaces of revolution in 3D. [44] recommended a model to construct axial turbine blade sections, using 11 parameters which define the position and radii of the leading and trailing edge circles and blade angles, the pressure and suction sides tangency points on these circles and throat position on the suction side. The suction side downstream of the throat is represented with an arc, whereas third order polynomials are used for the upstream part of the suction side and the pressure side. At junction points, C^1 continuity is guaranteed. The method was extended in [45], using Bezier curves instead of polynomials and circular arcs, raising the number of parameters to 17. In [46], the blade section geometry is formed by two thickness distributions attached to two parabolic camber-lines. The pressure and suction sides are defined by fourth-order splines, ensuring C^3 continuity at junction points. The method was modified in [47] so that the central parts of the suction and pressure sides be defined by mapping a desirable curvature distribution determined using a Bezier curve, leading to smooth Mach numbers distributions.

Several researchers used Bezier or B-Spline curves, eventually adding a circle at the leading and trailing edges, to parameterize and, then, optimize blade shapes. In [48], blade surfaces are constructed by lofting airfoils given as B-spline curves. Airfoils must lie on planar, cylindrical or conic surfaces. [49] used Bezier curves to model an airfoil. To ensure curves' continuity at the leading and trailing edges, some Bezier control points were aligned. The airfoil was optimized using control points' coordinates and the stagger angle as DoFs. [50] employed fourth-order splines to parameterize blade sections, for the design of transonic turbine rows. [51] combined the B-Spline representation of airfoils with the typical construction using the camber-line. The B-Spline control points are placed combining the camber-line with the thickness distributions. However, coordinates of control points are generally not suitable for optimizing 3D blades and even less to design shapes. For this reason, the parameterization should use parameters with physical meaning. [52] used a shape parameterization method based on Bezier curves to model 3D axial turbine and compressor rows. The camber-line of each 2D blade section is described by a Bezier curve with three control points, with their location determined by the leading and trailing edge blade angles, stagger angle and axial chord. Pressure and suction sides of the blade sections are defined using two Bezier curves, their role being to superimpose thickness on either side of the camber-line. The first control point of suction and pressure side curves corresponds with the leading edge; the second one is placed perpendicularly to the camber-line. The 3D blade shape is built superimposing 2D blade sections lying on cylindrical surfaces. The 2D sections are stacked using a line passing by the leading or trailing edge or the center of gravity of the section. A Bezier curve specifies the stacking line. [52] used a similar approach. The camber-line is defined by a curve with four control points with their location determined by six design parameters. These are the angular coordinates of leading and trailing edges in the cylindrical coordinate system, the angles that the camber-line forms with the tangents to the circles of the surface of revolution and two parameters that regulate the curve camber. 2D sections are built in a conformal space by adding thickness to the camber-line and transferred to the 3D space to construct the overall blade shape.

More recently, the progress made in CAD technology has offered the possibility to develop more advanced parameterization tools. Companies or academic teams have developed many software packages for turbomachinery design and parameterization for shape optimization. TURBODesign Suite [53] is a turbomachinery row modeler, based on a 3D inverse design method. The software handles various kinds of machines such as fans, pumps, compressors and turbines in axial or radial configurations. Rows are generated by providing data such as the speed of rotation, flow-related quantities at the inlet section, the meridional profile, blade thickness and the number of blades in the row. PropCad [54] is a software suite for the geometric modeling of marine propellers. The designs are generated by resorting to a database of common propeller shapes. The designed geometries may be exported in standard CAD format for further analysis with external tools. FINE/Design3D [55] is a software suite composed of various tools to design and improve the multidisciplinary performance of turbomachinery cascades. The suite has parametric modeling capabilities and provides multiple optimization algorithms. It interfaces with the mesh generator and CFD solver from the same software house. The BladeModeler [56] software is a tool for the design of rotating machinery components. It is used to design axial, mixed-flow and radial blade components. It allows the inclusion of geometric features, such as blade fillets, cut-offs and trims as well as the possibility to import complex tip geometries. BladeCAD [57] is a shape design tool for turbomachinery rows. The tool requires the definition of the blade cross-sections, which are then skinned into 3D geometry. Sections are built by providing blade angles, chord length, stagger angle, and thicknesses distributions. The T4T [58] software was developed for blades design purposes. The tool satisfies the basic requirements of turbomachinery blade design; it is compatible with CAD, mesh generators and analysis software through standard CAD output formats.

The use of a CAD package with general applicability, such as SolidWorks [59], to directly construct parametric models for turbomachinery blades is an alternative approach, with the substantial advantage that the designer makes use of all the infrastructure of the CAD package [25]. However, the designer might be asked to build dedicated parametric models for different types of blades. Additionally, the designer should rely on the geometric modeling tools of the specific CAD package, and it may be difficult to introduce other geometric modeling and geometric manipulation methodologies.

The motivation for developing the GMTurbo software [8], presented in this section, was to create a tool fulfilling essential requirements of turbomachinery row shape design, compatible with in-house analysis, mesh generation and morphing tools, differentiable (to be included in adjoint-based shape optimization) and with available and extendable source code. The method creates the row geometry by first parameterizing the shape of the blade's mean-camber-surface and, then, adding thickness. The majority of data given as input to the parameterization software are NURBS curves defining the meridional shape of the row as well as other geometric distributions in the spanwise direction. The NURBS curves make the parameterization method quite flexible, offering also a compact description for a wide range of blade geometries. To parameterize axisymmetric geometries, conformal mapping of a surface of revolution on the (m-meridional; θ-peripheral) plane is used. The final shape is generated by NURBS surface generation techniques such as skinning, revolving and trimming. NURBS also makes easier joining curves with C^1 continuity along the leading and trailing edges and exporting the shape in standard CAD format easier enough.

The background GMTurbo code has been developed within K. Tsiakas' PhD thesis [8] (in the same NTUA research group). In this thesis, it is extended to build the shape representation in NURBS format, Section 3.2.4 and the re-parameterization tool is developed to import existing blades into the GMTurbo format, Section 3.3. Section 3.2 illustrates the whole parameterization method. Two main ingredients used in the method described are NURBS (Appendix A) and conformal mapping (Appendix B).

3.2 GMTurbo Parameterization Process

The GMTurbo parameterization process of turbomachinery blades consists of four basic steps, as follows:

1. Parameterization of the row generatrices and the meridional shape of the leading and trailing edges, Section 3.2.1.
2. Parameterization of the mean-camber-surface of the blade, Section 3.2.2.
3. Superposition of thickness distribution on the mean-camber-surface to form the blade shape, Section 3.2.3.
4. Build and export the geometry in standard CAD format, Section 3.2.4.

In the following sections, the term "streamwise" refers to a distribution of data from inlet to outlet, whereas spanwise refers to a distribution from hub to shroud.

3.2.1 Meridional Shape

The first step of the row construction is the definition of the meridional shape of a single blade. For rows that revolves around the z-axis, the meridional shape is a projected 2D rz-space obtained from the cylindrical coordinate space $r\theta z$ by omitting the coordinate θ. The meridional shape is defined by four NURBS curves in the rz-space. These define *(a)* the hub generatrix, *(b)* the shroud generatrix, *(c)* the trace of the leading edge and *(d)* the trace of the trailing edge. Inlet and outlet boundaries are added to the meridional projection by connecting hub and shroud with a segment. Overall, these six curves determine an area called meridional contour for which a parameterization $h : [0, 1]^2 \to \mathbb{R}^2$ is defined as follows:

$$\boldsymbol{h}(u, v) = (r(u, v), z(u, v))^{\mathsf{T}} , \tag{3.1}$$

with $\boldsymbol{h}(u, 0)$ and $\boldsymbol{h}(u, 1)$ being the hub and shroud, respectively, and $\boldsymbol{h}(0, v)$ and $\boldsymbol{h}(1, v)$ the inlet and outlet, respectively. Moreover, the leading and trailing edges traces are defined by $\boldsymbol{h}(u_{LE}, v)$ and $\boldsymbol{h}(u_{TE}, v)$.

The user specifies also N_{pos} spanwise positions $v_i : i \in [1, \dots, N_{\text{pos}}]$, with $v_1 = 0$ and $v_{N_{\text{pos}}} = 1$, to generate additional intermediate streamwise generatrices. Each of these generatrices corresponds to a surface of revolution; the blade is built upon these intermediate surfaces of revolution between hub and shroud. As a result, the isospan generatrices defining these surfaces need to be computed through linear interpolation between the hub and shroud generatrices. These are defined by blending hub and shroud curves. Then, the mapping of

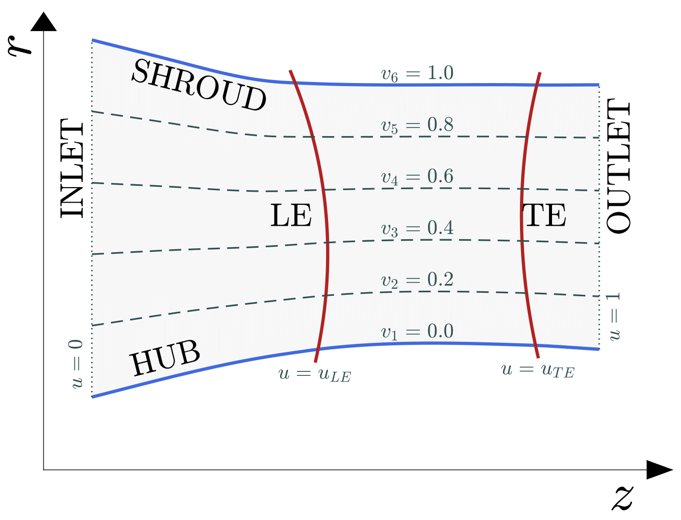


Figure 3.1: GMTurbo meridional shape parameterization. The leading and trailing edges traces and the hub and shroud generatrices are illustrated along with inlet and outlet boundaries (dotted curves) and the generatrices of intermediate (isospan) surfaces of revolution (dashed curves), on which blade sections are defined. Solid curves are provided as input to GMTurbo in the form of NURBS curves. Dashed and dotted curves are generated by the software. The image of $\boldsymbol{h}(u, v)$ (equation 3.1) is highlighted in light gray.

Equation B.1 is defined for each spanwise position v_i as:

$$\Phi|_{v_i} \; : \; (r(u, v_i)\cos\theta, r(u, v_i)\sin\theta, z(u, v_i)) \mapsto (m(u, v_i), \theta)$$
$$\text{with} \quad m(u, v_i) = \int_0^u \frac{\sqrt{\left(\frac{\mathrm{d}r(t,v_i)}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}z(t,v_i)}{\mathrm{d}t}\right)^2}}{r(t, v_i)} \mathrm{d}t\,. \tag{3.2}$$

An example of the meridional shape and the isospan curves is illustrated in Figure 3.1.

In a shape optimization, the control points of the leading and trailing edges are moved in the z-direction to modify the shape smoothly. If the meridional space occupied by the row has to remain fixed, these quantities are usually not modified. Using NURBS, the number of control points, i.e. the number of DoFs, is user-defined. The control points of the generatrices can be displaced to optimize the casing shape.

3.2.2 Mean-Camber-Surface

The second step after the definition of the meridional shape of a single blade is the construction of its medial surface. This construction is done discretely by building a camber-line on each isospan section $v \in [0, 1]$, as defined in the previous section. Each camber-line is built on the $m\theta$-plane corresponding to this isospan surface of revolution and is chosen to be represented by a cubic NURBS curve. Each of the blade isospan position requires the

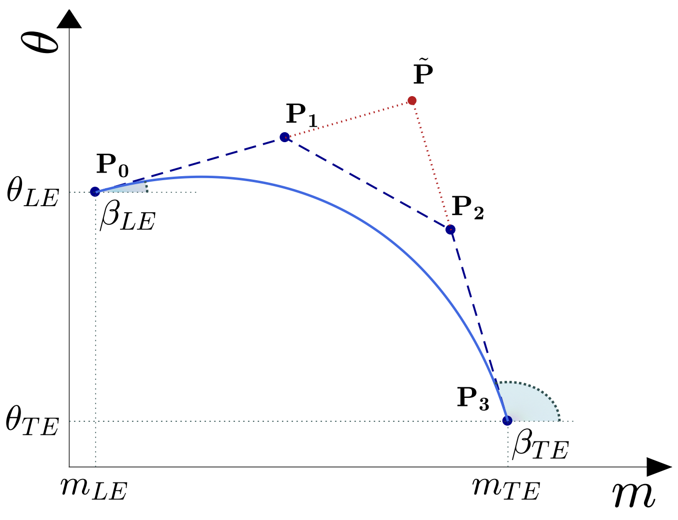


Figure 3.2: GMTurbo camber-line parameterization. Illustration of a camber-line on the $m\theta$ plane with some parameters. The curve is represented by a NURBS curve with four control points. For higher values of ξ_{LE}, the control point $\mathbf{P_1}$ moves towards $\tilde{\boldsymbol{P}}$; analogously for ξ_{TE} and the control point $\mathbf{P_2}$.

following three steps to define the four control points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ of the NURBS curve characterizing the camber-line.

- Define the leading and trailing edges peripheral positions θ_{LE} and θ_{TE}, respectively. Control points $\mathbf{P}_0$ and $\mathbf{P}_1$ are, then, defined as:

$$\begin{aligned} \mathbf{P}_0 &= (m_{LE}, \theta_{LE})^\mathsf{T} \\ \mathbf{P}_3 &= (m_{TE}, \theta_{TE})^\mathsf{T} \ . \end{aligned} \tag{3.3}$$

  m_{LE} and m_{TE} are inferred from the meridional shape.

- Define the metal angle β_{LE} and β_{TE} at the leading and trailing edges, respectively. These angles allow defining the point $\tilde{\boldsymbol{P}}$ as the intersection of the line passing by $\mathbf{P}_0$ and having angular coefficient β_{LE} with the line passing by $\mathbf{P}_3$ and having angular coefficient β_{TE}. Since the mapping defined by Equation 3.2 is conformal, all angles on the $m\theta$-plane are preserved during the inverse mapping to the 3D space.
- Define how the metal angles fade along the camber-line by defining the parameters $\xi_{LE} \in [0, 1]$ and $\xi_{TE} \in [0, 1]$. The control points $\mathbf{P}_1$ and $\mathbf{P}_2$ are defined by:

$$\begin{aligned} \mathbf{P}_1 &= \xi_{LE}\mathbf{P}_0 + (1 - \xi_{LE})\tilde{\boldsymbol{P}} \\ \mathbf{P}_2 &= \xi_{TE}\mathbf{P}_3 + (1 - \xi_{TE})\tilde{\boldsymbol{P}} \ . \end{aligned} \tag{3.4}$$

Figure 3.2 illustrates the quantities defined above (θ_{LE}, θ_{TE}, β_{LE}, β_{TE}, ξ_{LE}, ξ_{TE}) used to construct the camber-line at each isospan section. All these quantities are defined as spanwise distributions via NURBS to provide a smooth and flexible design scheme.

In a shape optimization, the control points of these distributions are moved to modify the shape smoothly.

3.2.3 Superposition of Thickness

After building the camber-line for each spanwise position, the blade thickness must be added to create the blade profiles. The user defines non-dimensional semi-thickness profiles for the pressure and suction side at several spanwise positions in the function of the arc-length of the camber-line and two streamwise distributions for the pressure and suction side, of factors to dimensionalize the former profiles. By performing spanwise interpolations, the non-dimensional semi-thickness factor for each point on the pressure and suction side is computed. Then, a streamwise distribution for the semi-thickness of each blade side is used to compute the dimensional value of the semi-thickness of any point of the blade. The semi-thickness is added on the $m\theta$-plane normal to the camber-line, to ensure that the blade profile sides lie on the isospan surfaces of revolution. The length preserving factor $1/r$ of equation B.7 is also taken into account. Namely:

$$\begin{aligned} \mathbf{x}_{PS}\left(t, v_i\right) &= \mathbf{x}_{MCL}\left(u(t), v_i\right) + \tau_{PS}\left(t\right)\; s_{PS}\left(v_i\right)\; \frac{\mathbf{n}\left(u(t), v_i\right)}{r\left(t, v_i\right)} \\ \mathbf{x}_{SS}\left(t, v_i\right) &= \mathbf{x}_{MCL}\left(u(t), v_i\right) + \tau_{SS}\left(t\right)\; s_{SS}\left(v_i\right)\; \frac{\mathbf{n}\left(u(t), v_i\right)}{r\left(t, v_i\right)} \end{aligned} \tag{3.5}$$

where $\mathbf{x}_{PS}(u, v_i)$ and $\mathbf{x}_{SS}(u, v_i)$ are the pressure and suction side curves at the isospan position v_i, respectively, $\mathbf{x}_{MCL}(s, v_i)$ the camber-line on the $m\theta$-plane corresponding to v_i; t is the parameters that make $u(t)$ an arc-length normalized parameter of the camber-line curve. $\tau_{PS}(t)$ and $\tau_{SS}(t)$ denote the non-dimensional semi-thickness, $s_{PS}(v_i)$ and $s_{SS}(v_i)$ the thickness scaling factor at the particular isospan position v_i, $\mathbf{n}$ the unit vector normal to the camber-line and $r(t, v_i)$ the radius. The first and second control points of $\tau_{PS}(t)$ and $\tau_{SS}(t)$ must have the same abscissa to ensure a C^1 continuous junction at the leading and trailing edges between the pressure and suction sides. The streamwise non-dimensional distributions of the semi-thickness at various isospan positions and the two spanwise distributions of the thickness for the pressure and suction side are defined as NURBS curves.

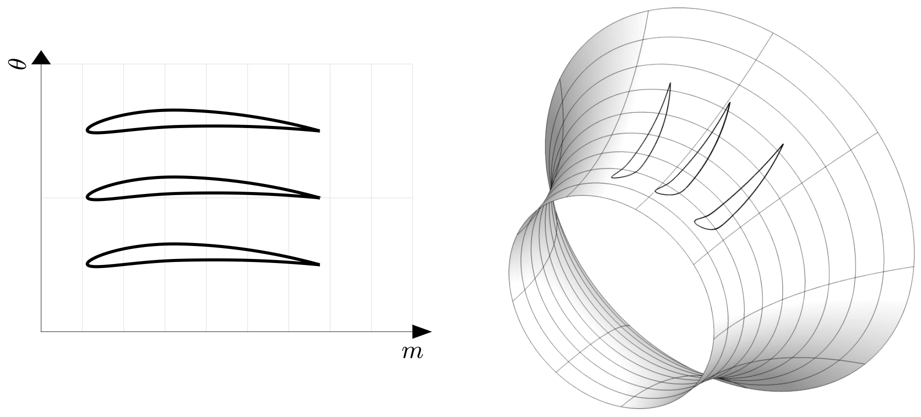


Figure 3.3: Left: Airfoils in the $m\theta$-plane. Right: 3D representation of the airfoils on the surface of revolution. Figure B.1 illustrates the 3D reference system.

3.2.4 Building and Exporting the Shape to Standard CAD Exchange Files

The 3D blade shape construction is completed by performing the inverse mapping of the airfoils defined in the $m\theta$-planes to the corresponding axisymmetric surfaces defined by the isospan meridional curves. The inverse mapping consists of producing the blade sections in Cartesian coordinates by inverting the mapping Φ defined in equation 3.2. Namely:

$$\begin{aligned} \Phi^{-1}\big|_{v_i} &: \ (m(u,v_i),\theta) \mapsto (r(u,v_i)\cos\theta, r(u,v_i)\sin\theta, z(u,v_i)) \\ \text{with} \quad u &: \ m(u,v_i) = \int_0^u \frac{\sqrt{\left(\frac{\mathrm{d}r(t,v_i)}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}z(t,v_i)}{\mathrm{d}t}\right)^2}}{r(t,v_i)}\mathrm{d}t\,. \end{aligned} \tag{3.6}$$

The inverse mapping requires to invert the integral of $m(u,v_i)$, to find the parameter u from the value of m, which is done numerically employing a hash table [60] and linear interpolation. The parameter u is used to find the values of r- and z-coordinates from the meridional projection; this is combined with the peripheral θ-coordinate to yield the 3D blade sections lying on the surface of revolution defined at each isospan blade section. Figure 3.3 illustrates the mapping of some airfoils into a surface of revolution.

For various applications, such as numerical simulations, it is fundamental to have a high-quality B-Rep model of the blade available in an exchangeable format. GMTurbo is extended to export the geometry to vendor-neutral CAD formats, IGES or STEP, for visualization and mesh generation purposes. IGES and STEP formats support many different entities, among which free-form surfaces in the form of NURBS surfaces that can be used to describe the skin of 3D models. The NURBS surfaces defining the whole row or the fluid domain of a single blade are constructed as follows.

In the 3D Cartesian space, airfoils are spanwise interpolated through NURBS curves, and a skinning algorithm [27, §10.3] is used to construct the NURBS surfaces of the blade. After the definition of the blade shape, the construction of the whole row is performed by rotating the blade by the user-defined pitch angle around the axis. Hub and shroud surfaces are generated as surfaces of revolution [27, §8.5] based on the generatrices defined by the user on the meridional projection. The definition of the fluid domain requires to build also the inlet, outlet and periodic surfaces; these are created as follows. Similar to the blades themselves, periodic surfaces are generated through skinning, and the inlet and outlet boundaries are produced as Coons patches through bilinear blending [27, §8.2]. Finally, hub and shroud surfaces must be trimmed: in order to avoid numerical instabilities, the blade sides are extended; the extension algorithm is described in detail in [61].

3.3 GMTurbo B-Rep Re-Parameterization Tool

Shape optimization usually starts from a reference shape to be modified in order to achieve better performance. Sometimes, the initial geometry is available in B-Rep format, being the standard for exchanging CAD files. However, in order to employ GMTurbo in the optimization process, the shape must first be translated in the GMTurbo format. In this section, a method to import a B-Rep format into a GMTurbo compatible form is presented. The procedure requires to compute the meridional shape, Section 3.3.1, and camber-lines and thickness profiles, Section 3.3.2, of the existing B-Rep model.

3.3.1 Meridional Shape

The first step of the re-parameterization technique is the computation of the meridional contour. Having the NURBS surfaces of the hub and shroud, the generatrices of each surface are obtained by taking an isoparametric line from the NURBS surfaces. These curves, one for the hub and one for the shroud, are the generatrices in the Cartesian xyz-coordinates that must be projected onto the rz-plane using the expressions

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ z &= z \end{aligned} \tag{3.7}$$

for the NURBS control points' coordinates, for a row revolving around the z-axis. If the B-Rep model revolves around another cardinal axis, equations 3.7 should be adjusted accordingly.

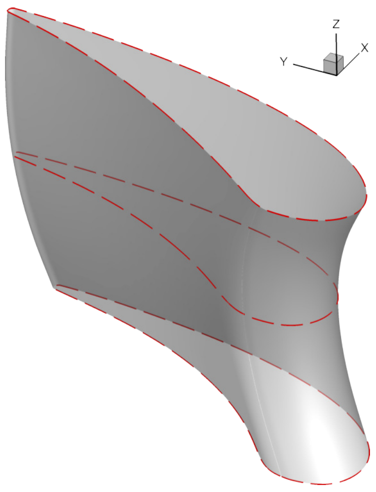


Figure 3.4: GMTurbo B-Rep re-parameterization tool. The blade is intersected with three revolved surfaces at isospan 0, 0.5 and 1. The intersections are illustrated as dashed red curves.

Next, a user-defined number of N_{pos} spanwise generatrices is created. Having the NURBS curves of the hub and shroud generatrices on the meridional plane, a linear interpolation of the control points produces intermediate generatrices, similarly to the ones illustrated in Figure 3.1. After defining the N_{pos} generatrices, in order to attain spanwise distributions of data, the operations take place for each spanwise generatrix, as described in the following sections. Leading and trailing edge meridional traces are also defined in the next section.

3.3.2 Mean-Camber-Lines and Thickness Profiles

Based on the N_{pos} generatrices defined in the previous section, N_{pos} NURBS surfaces of revolution are generated. Then, the intersections of these NURBS surfaces with the pressure and suction side surfaces, in NURBS form, are computed; the resulting NURBS curves are illustrated in Figure 3.4. These curves are the airfoils lying on the surfaces of revolution; the airfoils in the $m\theta$-space are computed using the mapping $\Phi : (x, y, z) \mapsto (m, \theta)$ defined in equation 3.2. Then, the analysis continues with the N_{pos} airfoils defined on the $m\theta$-planes. For each airfoil at span position $v_i, i \in [1, N_{\text{pos}}]$, the set of parameters that must be identified are the values of $\theta_{LE}, \theta_{TE}, \beta_{LE}, \beta_{TE}, \xi_{LE}, \xi_{TE}$, the airfoil thickness profile curves $\tau_{PS}(t)$ and $\tau_{SS}(t)$ and the thickness factors values s_{PS} and s_{SS} for the pressure and suction sides. These values require the definition of an approximated airfoil camber-line compatible with the GMTurbo parameterization: a cubic NURBS curve with four control points.

An algorithm, which computes the approximated camber-line using the exact definition of camber-line in a least-square sense, is developed: a line joining the leading and trailing edges of an airfoil being equidistant from the pressure and suction side curves. The structure of the algorithm is the following:

1. compute the Voronoi diagram of the airfoil boundary nodes (densely sampled);
2. prune Voronoi nodes lying outside the airfoil shape to obtain its skeleton;
3. reduce the density of nodes of the skeleton by *k*-means clustering;
4. least-square-fit the skeleton with a cubic NURBS with 4 control points by an EA;
5. refine the camber-line by SQP iterations.

The description of each of these steps follows.

The first and second steps refer to a skeletonization algorithm. Skeletonization is the process of generating the medial axis or “skeleton” of a shape. It was first discerned by [62] that for polygonal shapes, vertices of the Voronoi tessellation are an approximation of the skeleton of the shape. The Voronoi tessellation is a partitioning of a plane into regions based on proximity to a set of points; the Voronoi vertices delimit these regions. More details on the Voronoi tessellation are available in [63]. Skeleton and camber-line concepts are fundamentally related, mainly because both are based on the computation of the minimum distance of a point to a fixed set of points. In detail, the Voronoi tessellation generates a graph that divides the space into regions containing one of the seed points (the sampled points of the airfoil shape); any point within each region is closest to the seed point in that region. By construction, the Voronoi vertices are equidistant to at least three of the seed points. These three equidistant points from each center-point can be used to create circles inscribed in the airfoil, which is the typical construction procedure of the camber-line. [64] showed that the skeleton approximation converges to the actual skeleton as the sampling density becomes infinite; in practice, a “dense” sampling, in the order of thousands of points, is enough. The second step of the complete algorithm is necessary because the Voronoi tessellation generates vertices outside the airfoil shape, which leads to circles that would not be inscribed in the airfoil and whose centers, then, do not belong to the camber-line. These extra vertices are removed by solving the point-in-polygon problem by the ray-casting algorithm [65].

The skeletonization of an airfoil is illustrated in Figure 3.5. The algorithm succeeded in finding the camber-line for a large extent of the airfoil, but it fails close to the leading and trailing edges, where it might even create forks due to the radius of inscribed circles being lower than the semi-thickness. For this reason, refinement is needed in the following steps.

The third step, which reduces the density of nodes in the skeleton, is optional, though useful for the following reason. A NURBS curve must be fitted on the skeleton point

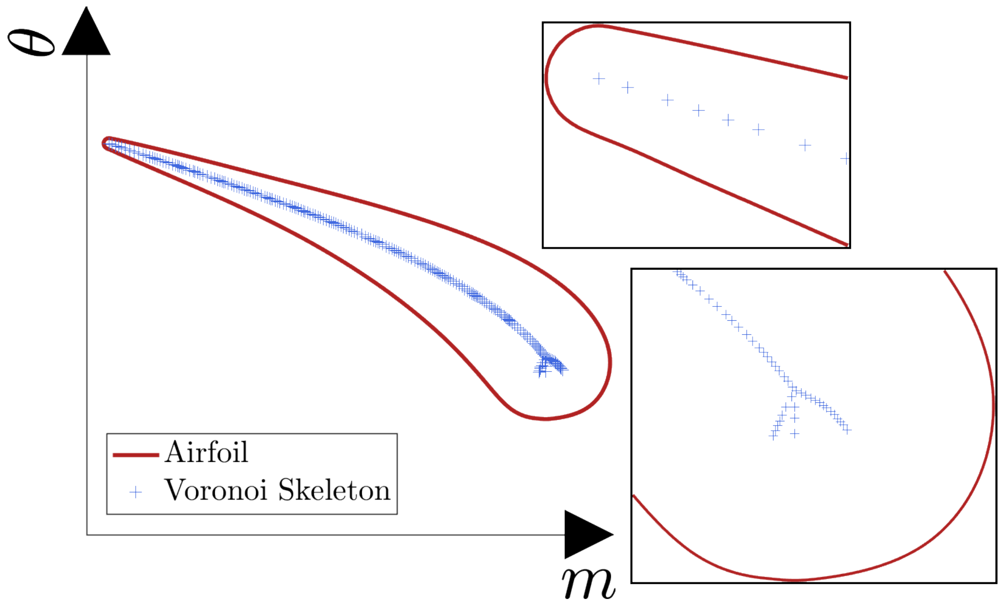


Figure 3.5: GMTurbo B-Rep re-parameterization tool. Approximated mean-camber-line computed by the Voronoi skeletonization algorithm. The point cloud is reduced by *k*-means clustering.

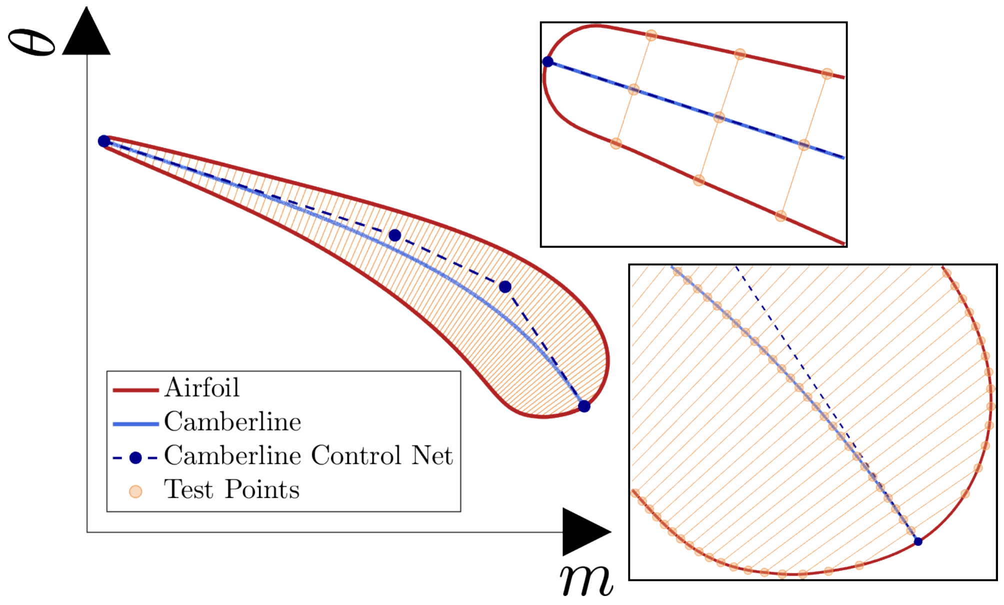


Figure 3.6: GMTurbo B-Rep re-parameterization tool. Result of the least-square fitting and refinement of the camber-line curve. Test points and segments used by the SQP-based refinement are illustrated.

cloud using a least square algorithm. The dense sampling of the airfoil shape needed by the skeletonization algorithm leads to skeletons formed by thousands of points, which are impractical to fit. For this reason, the amount of points is firstly reduced, from thousands to hundreds, using the k-means clustering algorithm [66]. k-means clustering partitions n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a representative point of the cluster. Such an algorithm provides a subset of the original point cloud that is a set of "well-distributed" skeleton points.

In the fourth step, the point cloud approximating the camber-line is fitted with a 4-points cubic NURBS. Curve fitting is based on a numerical optimization process. The objective function is the sum of the squares of the distances between the skeleton points and the curve. The DoFs are the $m\theta$-coordinates of the control points $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ and two NURBS parametric coordinates p_0 and p_3. Being $\boldsymbol{x}_{\text{BLADE}}(p)$ the airfoil periodic[1]. NURBS curve, $\boldsymbol{P}_0 = \boldsymbol{x}_{\text{BLADE}}(p_0)$ and $\boldsymbol{P}_3 = \boldsymbol{x}_{\text{BLADE}}(p_3)$, so that the resulting curve is implicitly constrained to have two intersections with the airfoil curve defining the leading and trailing edges. The periodicity of the airfoil curve is required to make the objective function continuous. To avoid local minima and due to the difficulty in providing a suitable set of initial DoFs, which would lead to the global minimum of the least-squares objective function, an EA-based optimization is used. Upper and lower bounds for $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are determined from the enlarged (doubled) bounding-box of the point cloud. The bounds of p_0 and p_3 are the minimum and maximum parametric coordinates of the airfoil contour. The resulting camber-line curve is an approximation of the real one, and especially next to the leading and trailing edges, the error might be higher due to the weakness of the Voronoi skeletonization mentioned before.

These approximations lead to the last step, which is a refinement of the camber-line estimate. A new objective function is set up, which is optimized by an SQP-based algorithm. The objective function is built at several test points sampled on the camber-line. A set of NURBS parametric coordinates determines the test points. A perpendicular line to the camber-line is constructed for each test point and intersected with the airfoil profiles so that two segments are identified: one on the side of the pressure side and another one on the side of the suction side. Figure 3.6 clarifies the segments. The objective is the sum of the absolute value of the difference of length between the segments on the pressure and suction side. Such an objective function reflects the definition above of camber-line: given

[1]A periodic curve is a closed curve with coinciding start and endpoints. The start and endpoints have unclamped spans so that derivatives continuities are guaranteed. Moreover, the parametric coordinate is also periodic, which means that parametric coordinates outside the range [0, 1] have an equivalent in such range so that the curve can be swept continuously. More information is provided in Appendix A.

the diameter and a center-point, it is possible to draw circles inscribed into the airfoil.

Having computed the NURBS cubic camber-line, hence $\boldsymbol{P}_0$, $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, $\boldsymbol{P}_3$ and $\tilde{\boldsymbol{P}}$ the angles $\theta_{LE}, \theta_{TE}, \beta_{LE}, \beta_{TE}$ and coefficients ξ_{LE} and ξ_{TE} are computed from their definitions:

$$\begin{aligned}
\theta_{LE} &= \boldsymbol{P}_{0,\theta} & \theta_{TE} &= \boldsymbol{P}_{3,\theta} \\
\beta_{LE} &= \tan^{-1}\left(\frac{\boldsymbol{P}_{1,\theta} - \boldsymbol{P}_{0,\theta}}{\boldsymbol{P}_{1,m} - \boldsymbol{P}_{0,m}}\right) & \beta_{TE} &= \tan^{-1}\left(\frac{\boldsymbol{P}_{2,\theta} - \boldsymbol{P}_{3,\theta}}{\boldsymbol{P}_{2,m} - \boldsymbol{P}_{3,m}}\right) \\
\xi_{LE} &= \frac{\left\|\boldsymbol{P}_1 - \tilde{\boldsymbol{P}}\right\|}{\left\|\boldsymbol{P}_0 - \tilde{\boldsymbol{P}}\right\|} & \xi_{TE} &= \frac{\left\|\boldsymbol{P}_2 - \tilde{\boldsymbol{P}}\right\|}{\left\|\boldsymbol{P}_3 - \tilde{\boldsymbol{P}}\right\|}
\end{aligned} \tag{3.8}$$

The last step to the complete re-parameterization of the blade B-Rep is the computation of the thickness profiles, which are combinations of the non-dimensional streamwise thickness profiles for each spanwise position and the spanwise thickness factor distributions. Having the airfoil and camber-line NURBS representations, the normal distances of the camber-line to both sides are computed, providing the dimensional thickness profile curves. Then, normalizing each profile so that its maximum value is one leads to the non-dimensional thickness profiles needed by GMTurbo. Finally, the normalization factor of each thickness profile is used to build the spanwise thickness factor curves for the pressure and suction sides.

3.4 GMTurbo and Re-parameterization Tool Demonstration

Indicative blade rows are presented in order to demonstrate the capabilities of GMTurbo and the GMTurbo re-parameterization tool. Figure 3.7 and 3.8 illustrates a turbine and a compressor stator, respectively, designed using the GMTurbo software. In detail, in Figure 3.7, a stator named MEL, which has been presented in [67] is demonstrated. This is imported in GMTurbo using the re-parameterization tool. In Figure 3.8, a compressor stator named TUB is presented. The shape illustrated herein is the result of an optimization conducted in Chapter 7. The starting shape that was imported in GMTurbo is illustrated in Figure 7.1. The MEL blade is designed using the input curves in Figure 3.9. The optimized TUB blade is created using the input curves in Figure 3.10, whereas the starting geometry is created with the ones in Figure 7.9.

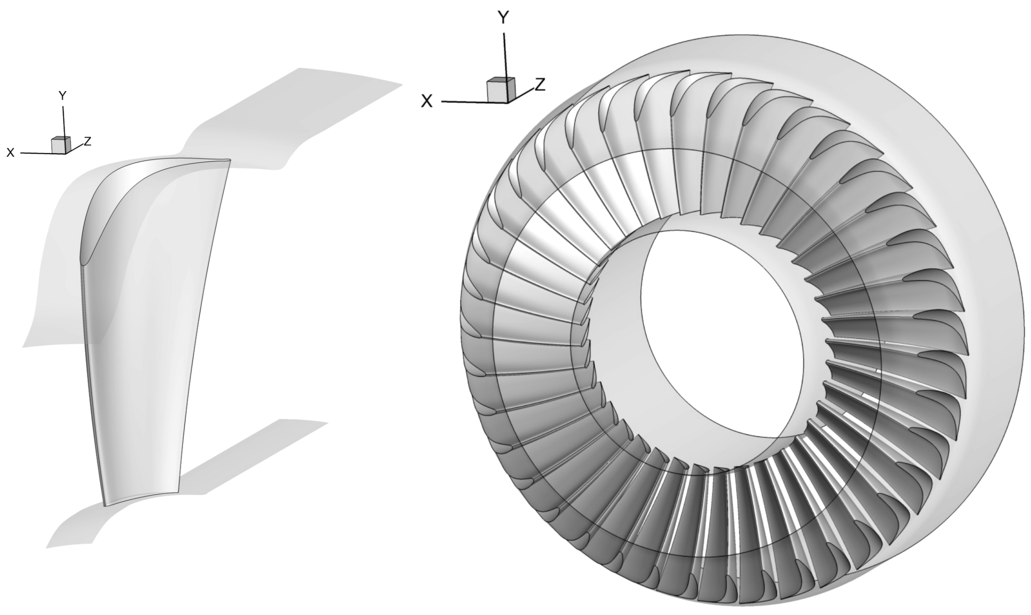


Figure 3.7: MEL Stator. NURBS surfaces generated using GMTURBO. Left: a single blade. Right: the full row.

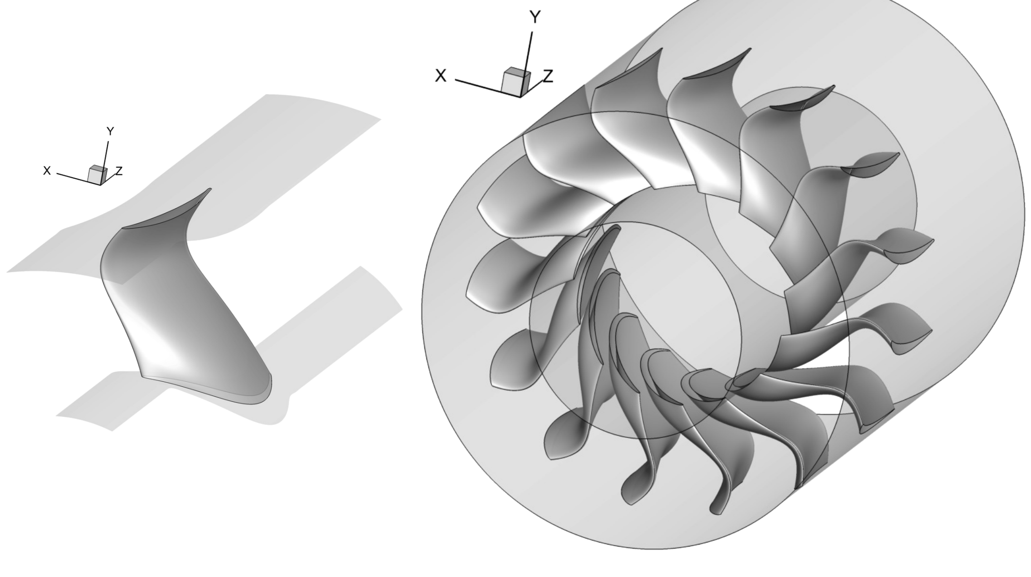


Figure 3.8: TUB optimized blade (See Chapter 7). NURBS surfaces generated using GMTURBO. Left: a single blade. Right: the full row.

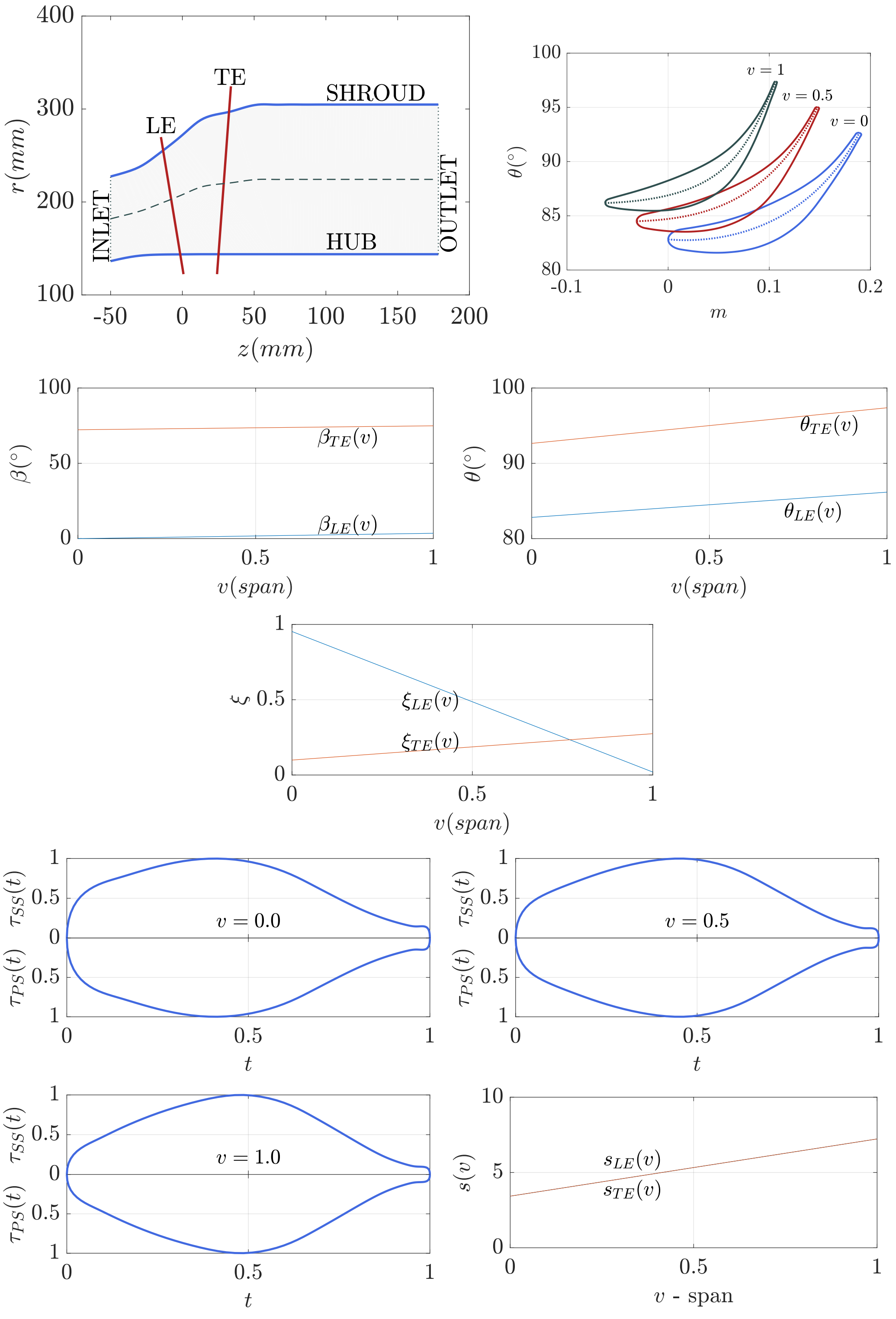


Figure 3.9: MEL Stator. Curves used as input to GMTurbo.

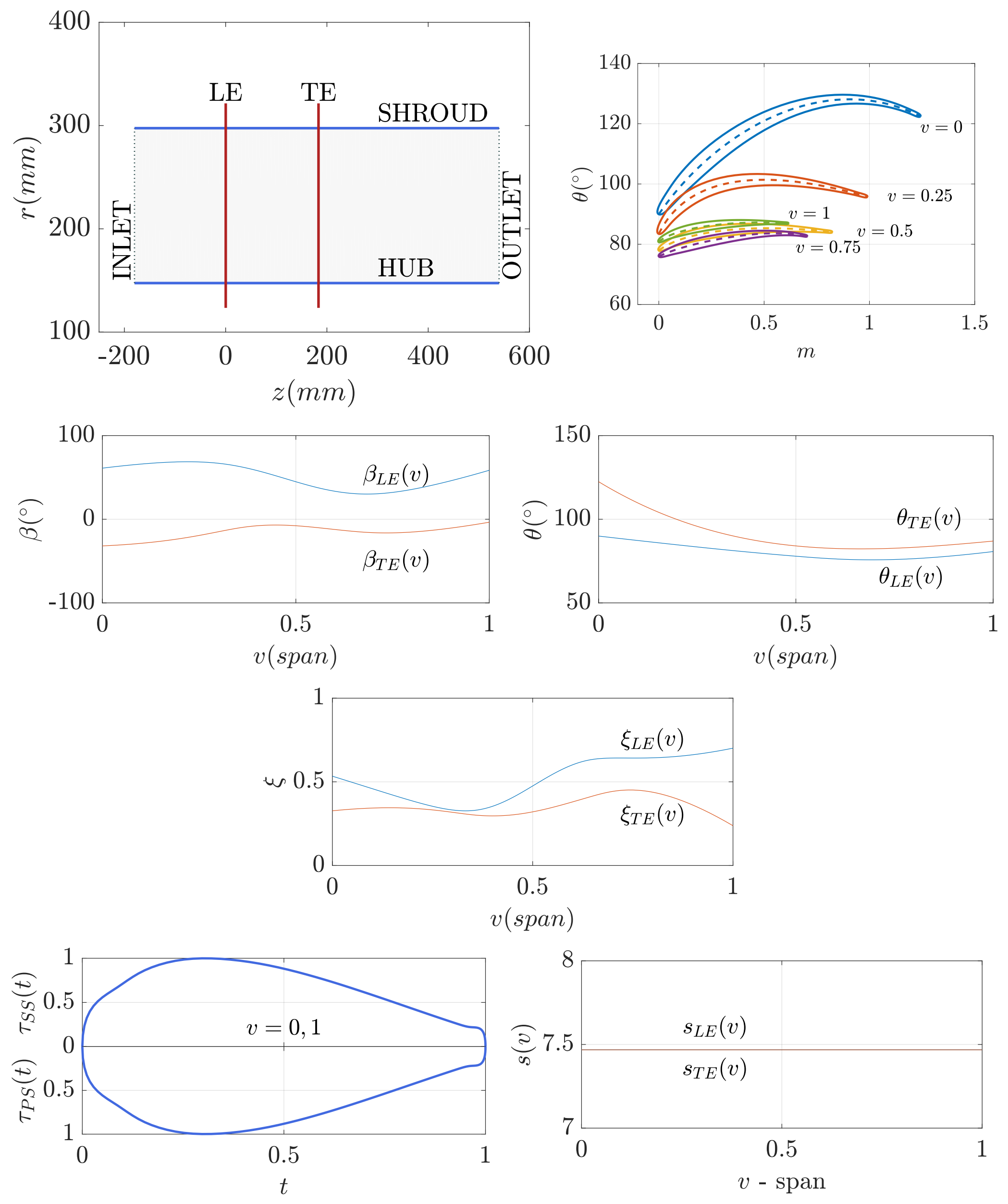


Figure 3.10: TUB optimized blade (See Chapter 7). Curves used as input to GMTurbo.

Chapter 4

CFD Surface Mesh Displacement and Parameterization Differentiation

In this chapter, two methods to integrate GMTurbo and the B-Rep-Morpher into gradient-based and -free shape optimization loops are illustrated. In detail, Section 4.1 presents a tool to deform CFD surface meshes according to deformed B-Rep models, and Section 4.2 explains a differentiation tool to compute geometric sensitivities; these are essential when using adjoint to compute gradients (in gradient-based optimization loops).

The strategy for updating the CFD surface mesh to the already changed B-Rep models enables the inclusion of the proposed parameterization methods into optimization loops, avoiding any into-the-loop dependence on mesh generation packages. The surface mesh is updated by computing new nodal coordinates based on the updated NURBS parametric coordinates; these are computed according to changes in the parametric domain of trimmed surfaces by an RBF-based interpolation. The displacement of the surface nodes is, then, processed to displace the CFD volume mesh.

4.1 CFD Surface Mesh Displacement

Deforming an existing 3D CFD mesh according to a morphed CAD model is fundamental to reduce the computational cost of aerodynamic shape optimizations. The proposed CFD surface mesh displacement method exploits the fact that, in the shape-morphing strategy presented in Chapter 2 and in the turbomachinery blades parameterization tool presented

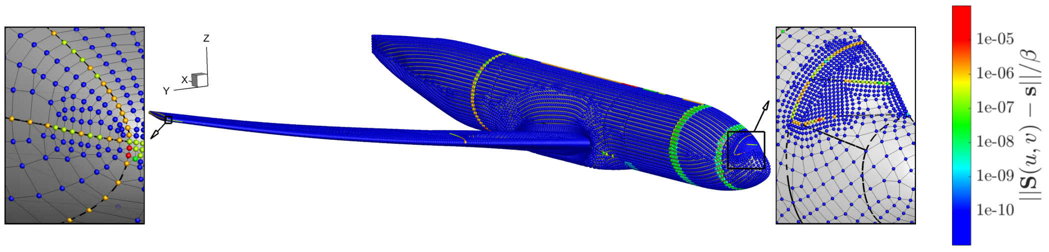


Figure 4.1: DPW6 aircraft. Results of the projection of a CFD surface mesh onto the CAD model. Residual values $||\mathbf{S}(u, v) - \mathbf{s}||$ are normalized w.r.t. the length of the diagonal of the surface mesh bounding box β. The edges of the CAD model are also shown as black dashed curves.

in Chapter 3, the CAD model topology does not alter during the deformations; thus, the NURBS-based representation of shapes enables the mapping of the CFD mesh surface nodes onto the (u, v) NURBS surfaces parametric spaces. Such a mapping works by determining which NURBS surface a mesh node $\boldsymbol{s} \in \mathbb{R}^3$ belongs to and identifying the parameters (u, v), so that $\boldsymbol{s} \approx \boldsymbol{S}(u, v)$ by taking into account the domain Ω of each NURBS surface (part of the surface that is not trimmed, Figure 2.2). The projection of each surface mesh node is checked separately on each surface. That is, for each surface, the (u, v) parameters of each node are computed by solving the problem

$$\min_{(u,v)\in\Omega} ||\boldsymbol{S}(u, v) - \boldsymbol{s}|| \ . \tag{4.1}$$

Surface mesh nodes at joints belong to more than one surface, and their parametric values are computed for all of them. Because the problem of equation 4.1 is overdetermined (three equations with two unknowns), it is solved by non-linear least-squares. Care must be taken in case the NURBS patch is not $C1$ continuous and in order to provide a good starting point for the least-squares iterations. More details on the point projection algorithm can be found in [27, §6.1]. Because the projection task is expensive, even if fully parallelized, due to a large number of CFD surface mesh nodes to be projected, the (u, v) values are computed once and stored. Figure 4.1 illustrates the residuals of the projection of a CFD surface mesh on the B-rep model of the DPW6 case. Mesh nodes lying close to or along the boundary of the parametric domains Ω are subject to higher residual values from the minimization of the problem in equation 4.1 due to CAD model tolerances involved in the definition of the domain Ω itself.

When deforming NURBS surfaces, their parametric domains Ω might change due to changes in the p-curves $\boldsymbol{C}_{pc}$, as described in Section 2.3.1 for the B-Rep-Morpher and analogously for the GMTurbo. Before computing the surface mesh, the mesh projected

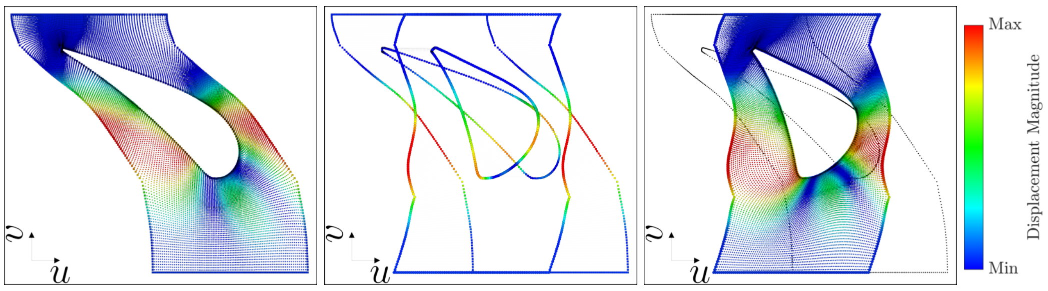


Figure 4.2: RRD stator (Chapter 5). Displacement of the hub surface mesh in the NURBS surface parametric space. Left: Initial positions of the nodes in the $u - v$ plane. Middle: displacements plotted along the p-curves of the reference and morphed shape. Right: The nodes displaced by the RBF interpolation. Colors represent the magnitude of the nodal displacements.

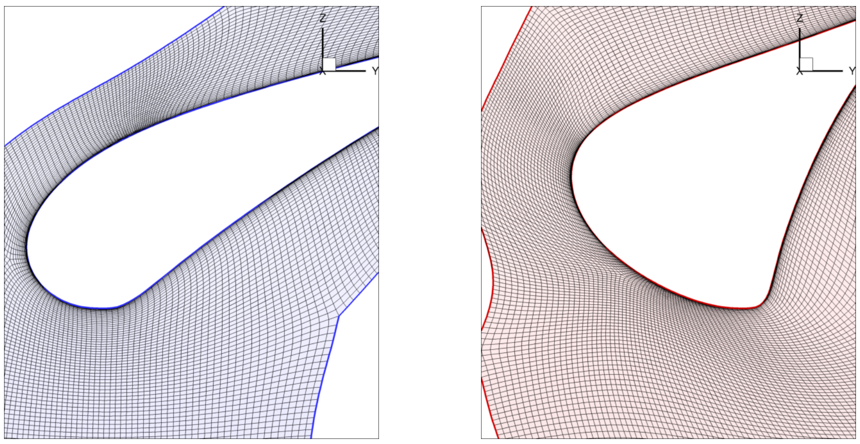


Figure 4.3: RRD stator. Reference and a morphed Cartesian mesh. Left: The reference surface mesh projected onto the (u, v) space of the hub surface. Right: The morphed mesh computed with the morphed $(\hat{u}, \hat{v})$ parametric nodes.

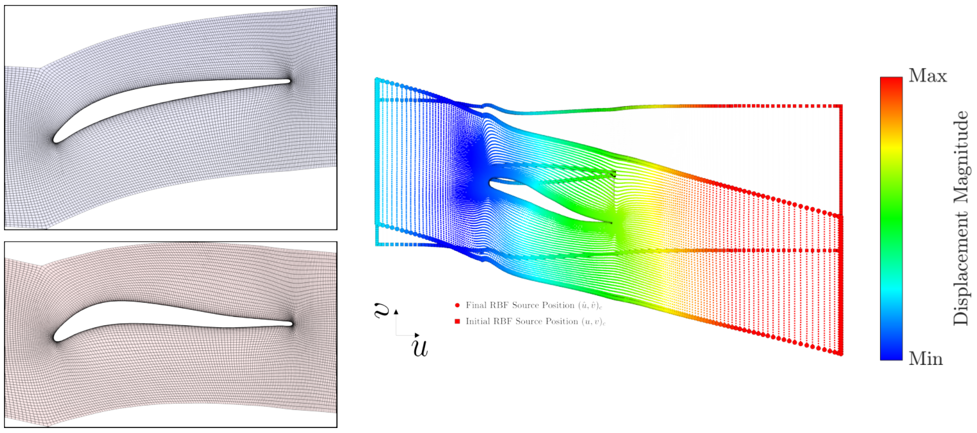


Figure 4.4: TUB stator. Adaptation of the surface mesh of the shroud for a design variation. Right: the adaptation of the mesh in the (u, v) NURBS parametric space. Left: the CFD mesh in the Cartesian space; top-left, the reference surface mesh; bottom-left the displaced surface mesh.

onto the parametric domain Ω needs to be adapted to the new parametric domain $\hat{\Omega}$ defined by the new p-curves $\hat{\boldsymbol{C}}_{pc}$. The parametric domain is adapted using RBF-based interpolation. As noted, there is a one-to-one correspondence between surfaces, curves and p-curves of the reference and deformed CAD models. For each parametric domain, nodes are distributed equidistantly along the p-curves (distances are measured in the parametric space of the surface) obtained from the reference and deformed CAD models: the displacement of each node $(u, v) \in \Omega$, computed from the p-curves, is used to find the RBF deformation function $\boldsymbol{d}_{\mathrm{PAR}} : \mathbb{R}^2 \to \mathbb{R}^2$. The thin-plate spline activation function (Table 2.1) is used. The new parametric values contained in $\hat{\Omega}$ are then computed as $(\hat{u}, \hat{v}) = (u, v) + \boldsymbol{d}_{\mathrm{PAR}}(u, v)$ and used to find the new CFD surface mesh nodes $\hat{\boldsymbol{s}} = \hat{\hat{\boldsymbol{S}}}(\hat{u}, \hat{v})$, where $\hat{\hat{\boldsymbol{S}}}$ is the new NURBS surface. For nodes along the joints, the average position is taken, and the surrounding nodes are adjusted by local RBF interpolation based on the Wendland C0; this ensures the validity of the CFD surface mesh in case computational nodes are close to each other according to the user-defined geometric tolerance associated with the B-Rep model.

Figures 4.2 and 4.3 illustrate the procedure for displacing the surface mesh in the NURBS parametric space for the hub surface of a turbine stator parameterized with the B-Rep-Morpher tool. The reference mesh in Figure 4.3 is projected onto and displaced into the NURBS parametric space of the hub surface, Figure 4.2. The displaced surface mesh in the Cartesian space is illustrated in Figure 4.3. Figure 4.4 shows the corresponding procedure for a compressor stator parametrized with the GMTurbo tool.

4.2 Differentiation of the Parameterization Procedure

CFD simulations are computationally expensive. In this regard, the use of a gradient-based optimization approach, which requires few iterations to find the optimum, is desirable. However, gradient-based methods require the gradient of the objective function w.r.t. the design variables. The simplest approach to compute the derivatives for each design variable is by means of Finite Differencess (FDs), where the effect of a parameter change is computed by analyzing both the baseline and perturbed geometries. For each DoF, the parameterization method creates a perturbed geometry for which the value change in performance enables to compute the derivative. This strategy is subject to the so-called “curse of dimensionality”, and quickly becomes impractical for models with large design spaces.

Gradient-based methods supported by the continuous or discrete adjoint for computing the objective function gradient have become very attractive in CFD-based optimization with many design variables, mainly because adjoint methods compute gradients at a cost independent of their number. However, adjoint methods primarily compute sensitivities of an objective function $\mathcal{F}$ w.r.t. the nodal coordinates on the mesh surface, namely $\partial\mathcal{F}/\partial\mathbf{x}$ [13]. To obtain the sensitivities of $\mathcal{F}$ w.r.t. the design variables $\boldsymbol{b}$, the parameterization method must be differentiated (i.e., $\partial\mathbf{x}/\partial\mathbf{b}$ must be computed; geometric sensitivities) and linked to the sensitivities computed by the adjoint method through the chain rule

$$\frac{\partial\mathcal{F}}{\partial\boldsymbol{b}} = \frac{\partial\mathcal{F}}{\partial\boldsymbol{x}}\frac{\partial\boldsymbol{x}}{\partial\boldsymbol{b}} \quad . \tag{4.2}$$

Three main methods are acknowledged for computing the aforementioned geometric sensitivities:

- Direct (analytic) differentiation. Closed-form expressions can be produced by differentiating all operations within the parameterization tool. Sensitivities computed in this way are generally accurate to working precision but can be unfeasible for sophisticated modeling features, such as NURBS surface-surface intersections.
- Algorithmic differentiation [68]. It is a technique to evaluate the derivative of a function specified by a computer program. The source code must be modified. It is based on the fact that any computer program executes a sequence of elementary arithmetic operations such as additions and multiplications and elementary functions such as exp, log and sin. By applying the chain rule in cascade to these operations, derivatives can be computed automatically.
- Numerical differentiation, namely FDs [69]. Differences between the base and perturbed geometries (first-order accurate) or just perturbed geometries (second-order accurate) are computed. While conceptually simple, this method requires pairing corresponding points on the base and perturbed geometries, before computing the corresponding deviation. Moreover, the selection of an appropriate perturbation size, in order to balance truncation and round-off errors, is fundamental.

[70] adopted an approach based on FDs and the parametric position of mesh nodes on the surface of the reference CAD model. Following a design variable perturbation, the new surface nodes positions are computed based on the parametric values computed on the original model, without taking into account the displacement of trimmed intersections. [71] used the FDs method to compute geometric sensitivities by comparing the parametric description of the faces in the original CAD model with the parametric description after the perturbation of one of the design parameters. [72] proposed an approach to compute the geometric sensitivities based on the computation of the intersection between an embedded

boundary Cartesian mesh and a mesh of the component geometry. [19] and [73] computed geometric sensitivities using methods based on geometric faceting representations, such as STL format exported directly from the CAD package or more accurate meshes. Baseline and perturbed faceted shapes presentations are compared by projecting the former onto the latter. [74] applied algorithmic differentiation to the open-source CAD kernel Open CASCADE Technology.

The differentiation method herein used makes use of FDs. Sensitivities with the FDs method are obtained by computing the relative change of the shape boundary position due to single design variable perturbations. Consequently, the cost for computing all the sensitivities $\delta\mathbf{x}/\delta\mathbf{b}$ scales linearly with the number of design variables. However, the time required to generate a single design by running the parameterization software is usually low, and the method is readily parallelizable. Moreover, with FDs, the parameterization is treated as a black box, which implies a straightforward implementation. Thanks to the strategy implemented to displace the CFD surface mesh according to shape variations (Section 4.1), geometric sensitivities are easily computed at the surface mesh nodes.

This section is structured as follows. Section 4.2.1 describes the algorithmic steps of the differentiation of GMTurbo (illustrated in Chapter 3) and the B-Rep-Morpher software (illustrated in Chapter 2). Applications shown in Section 4.2.2 include the TUB compressor stator (illustrated in Chapter 7) and a double elbow duct.

4.2.1 Algorithmic Steps of the Differentiation of the Parameterization Procedure

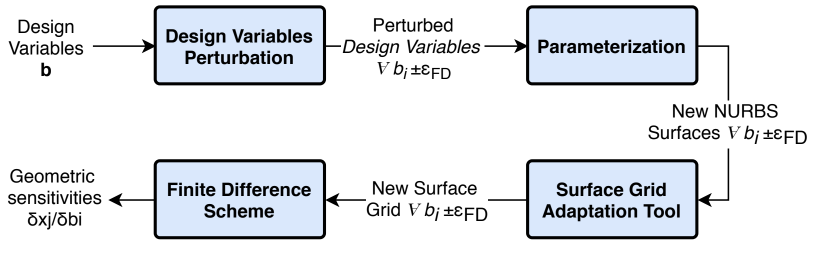

Design Variables b
Design Variables Perturbation
Perturbed Design Variables ∀ b_i ±ε_FD
Parameterization
New NURBS Surfaces ∀ b_i ±ε_FD
Surface Grid Adaptation Tool
New Surface Grid ∀ b_i ±ε_FD
Finite Difference Scheme
Geometric sensitivities δxj/δbi

Figure 4.5: Sketch of the differentiation procedure for the GMTurbo and B-Rep-Morpher parameterization tool. Input to the process is the set of design variables. Each design variable is perturbed and used to generate the new NURBS surfaces with the GMTurbo or B-Rep-Morpher parameterization. The new NURBS surfaces are used to feed the surface mesh displacement tool (see Section 4.1). Finally, the so-generated meshes are used to compute the geometric sensitivities using the FDs method.

The differentiation process is sketched in Figure 4.5. This approach requires to perturb each design variable and compute the surface mesh for each of the unperturbed and

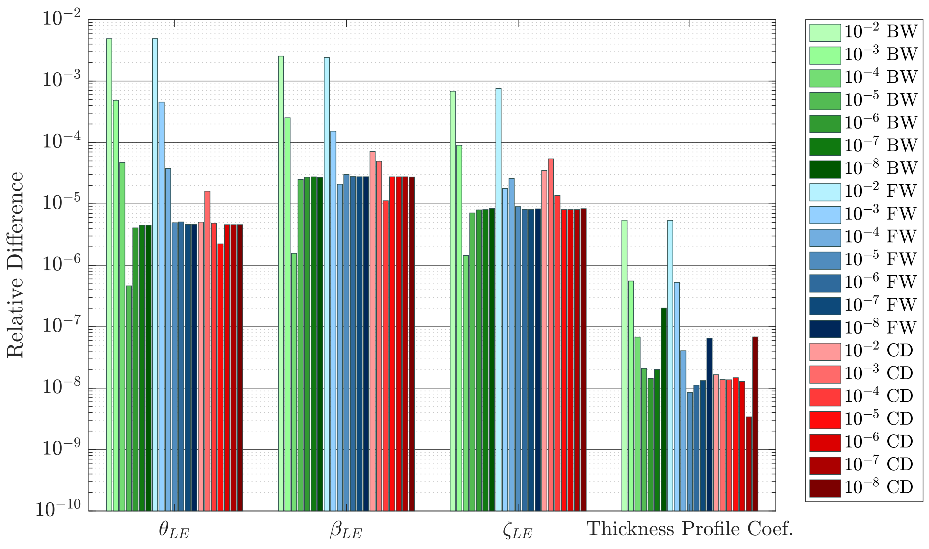


Figure 4.6: TUB stator. FDs step-size independence study for 4 design variables of the GMTurbo tool. BW stands for backward, FW for forward and CD for the central difference scheme. Comparison is made after the combination with the flow sensitivities as prescribed by the chain rule in Equation 4.2. The relative difference is defined by equation 4.6.

perturbed models. Then, the FDs scheme is applied for nodes lying on the surface. Given a point on the original geometry, it must be possible to determine its new position on the perturbed ones. To this end, the algorithm described in Section 4.1 fits the purpose and provides geometric sensitivities directly on the surface mesh nodes, which is requisite to apply the chain rule of equation 4.2. Since the surface mesh displacement method applies to both the GMTurbo and B-Rep-Morpher parameterization tools, the differentiation procedure is the same for both.

Denoting by $\mathbf{S}_{i+}(u,v)$ the surface generated by adding ε_{FD} to the design variable b_i, $\mathbf{S}_{i-}(u,v)$ the surface generated by subtracting ε_{FD} from the design variable b_i, the geometric sensitivities for a surface node j is computed using one of the following schemes:

- forward difference

$$\frac{\partial \mathbf{x_j}}{\partial \mathbf{b_i}} = \frac{\mathbf{S}_{i+}(u_j,v_j) - \mathbf{S}(u_j,v_j)}{\varepsilon_{FD}} + O(\varepsilon_{FD}) \tag{4.3}$$

- backward difference

$$\frac{\partial \mathbf{x_j}}{\partial \mathbf{b_i}} = \frac{\mathbf{S}(u_j,v_j) - \mathbf{S}_{i-}(u_j,v_j)}{\varepsilon_{FD}} + O(\varepsilon_{FD}) \tag{4.4}$$

- central difference

$$\frac{\partial \mathbf{x_j}}{\partial \mathbf{b_i}} = \frac{\mathbf{S}_{i+}(u_j, v_j) - \mathbf{S}_{i-}(u_j, v_j)}{2\,\varepsilon_{FD}} + O(\varepsilon_{FD}{}^2) \tag{4.5}$$

4.2.2 Demonstration of the Differentiation of the Parameterization Procedure

The software has been integrated into CAD-based, gradient-based shape optimizations. Examples can be found in Chapter 7 and Section 2.4. Figure 4.7 illustrates the geometric sensitivities for 4 design variables for the TUB compressor stator blade shown in Chapter 7 and parameterized with the GMTurbo tool. Figure 4.8 depicts geometric sensitivities for the displacement of a handle along the z-direction of the parameterization of the double elbow duct using the B-Rep-Morpher tool.

An FDs step-size independence study is carried out for the TUB compressor stator blade: 7 different step sizes ranging from 10^{-2} to 10^{-8} and three different schemes, forward, backward and central difference, are used. The relative difference, by changing the scheme and step-size for 4 design variables, is shown in Figure 4.6. The design variables are the same 4 presented in Figure 4.7. The comparison is made after the combination of the geometric sensitivities with the flow ones as prescribed by the chain rule in Equation 4.2. The relative difference is defined as

$$\frac{\left|\frac{\delta \mathcal{F}}{\delta b_i} - \left(\frac{\delta \mathcal{F}}{\delta b_i}\right)_{ref}\right|}{\left|\left(\frac{\delta \mathcal{F}}{\delta b_i}\right)_{ref}\right|} \tag{4.6}$$

where the reference value $\left(\frac{\delta \mathcal{F}}{\delta b_i}\right)_{ref}$ is computed as the average of the gradients $\frac{\delta \mathcal{F}}{\delta b_i}$, by varying the step-size and scheme for each design variable b_i. As can be seen, reducing the step-size, the relative difference converges to an acceptable value, which makes the method robust and precise for a wide range of step-sizes.

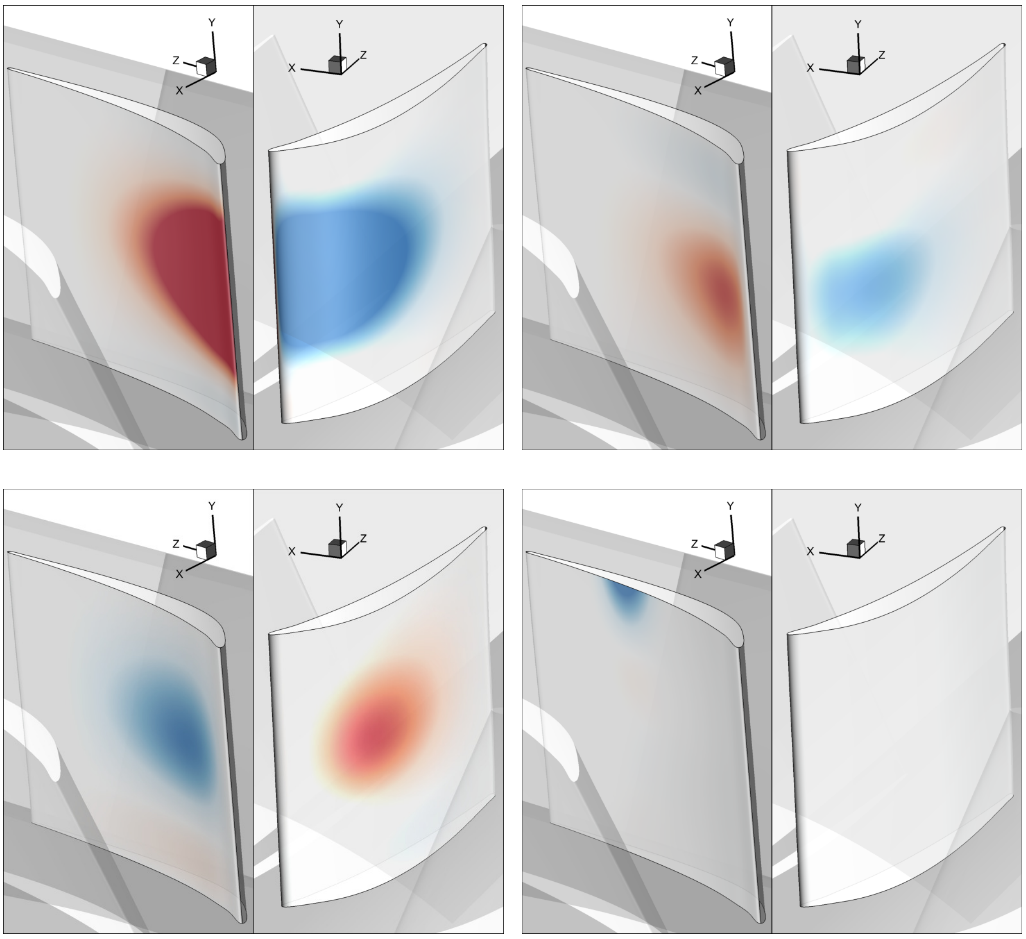


Figure 4.7: TUB stator. Geometric sensitivity w.r.t. 4 design variables. Colors quantify normal sensitivities. Blue indicates that the surface is pulled outwards the fluid domain by a positive change in the design variable, whereas red that the surface is pushed in the opposite direction. Top-Left: θ_{LE}. Top-Right: β_{LE}. Bottom-Left: ζ_{LE}. Bottom-Right: Thickness profile coefficient of the pressure side.

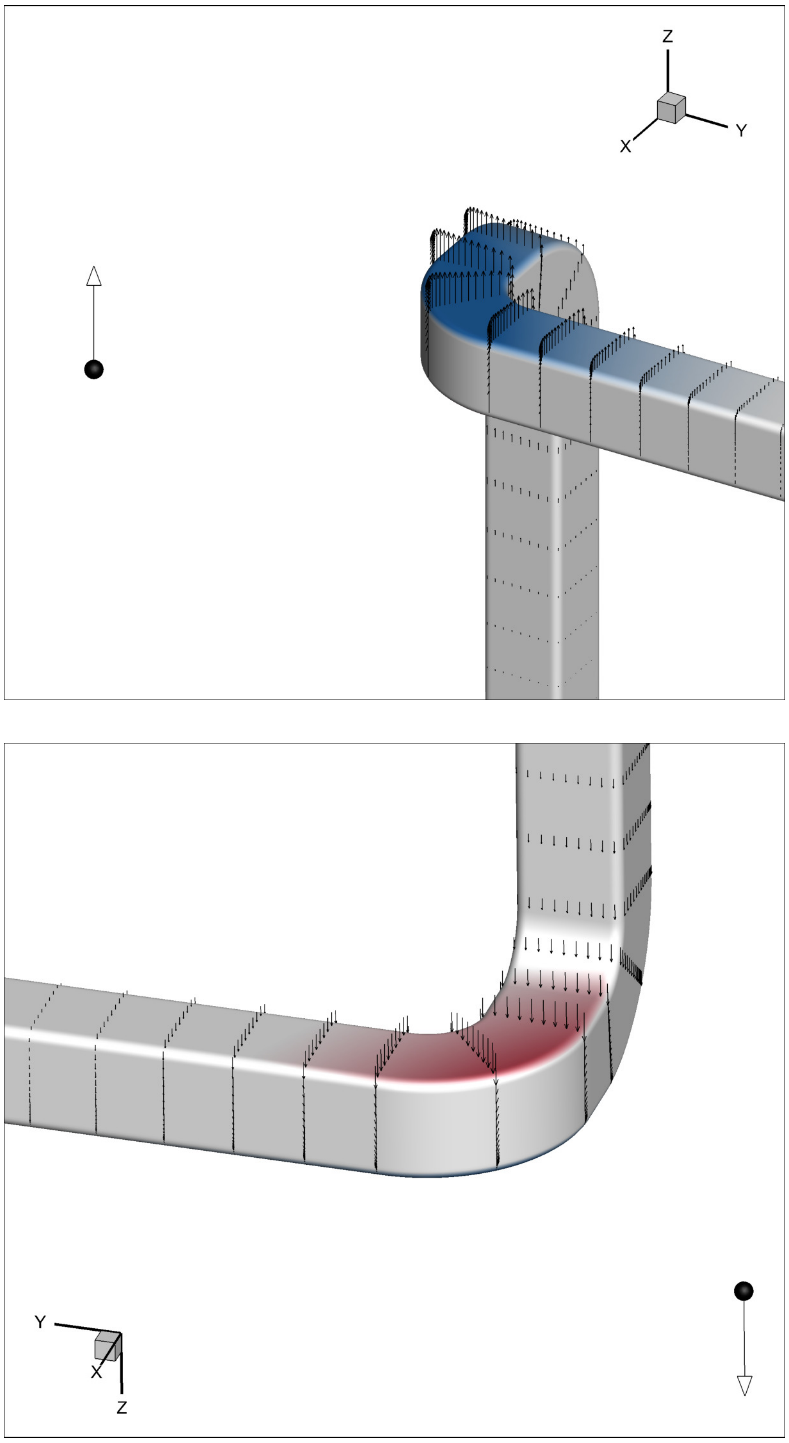


Figure 4.8: Double elbow duct. Geometric sensitivity w.r.t. a design variable, namely the displacement in the z-direction of the handle illustrated as a sphere. Colors quantify normal sensitivities. Blue indicates that the surface is pulled outwards the fluid domain by a displacement of the handle in the positive z-direction, whereas red that the surface is pushed in the opposite direction. Vectors illustrate 3D geometric sensitivities.

Chapter 5

A Two-Step Radial Basis Function-Based CFD Mesh Displacement Tool

Mesh displacement based on RBF interpolation [32] is known for its ability to preserve the validity and quality of the mesh, even for large displacements, without being affected by mesh connectivity. However, in case of large meshes, such as those used in real-world CFD applications, RBF interpolation, in its standard formulation, becomes excessively expensive. This chapter proposes a cost reduction technique for mesh displacement based on RBF by splitting the process into two steps. In the first step, named “predictor”, a data reduction algorithm that adaptively agglomerates mesh boundary nodes by reducing the RBF interpolation problem size is used. Upon completion of the first step, due to the agglomeration and the fact that the RBF interpolation is applied to the boundary nodes too, the so-displaced boundaries do not perfectly match the given displacements; thus, the position of the boundary nodes must be corrected during the second step, named “corrector”. The latter performs a local deformation based on RBF kernels with local support, to make the boundary conform to the known displacements of its nodes. The proposed method is accelerated by means of the Sparse Approximate Inverse (SPAI) preconditioner based on geometric considerations and the Fast Multipole Method (FMM). The method and the programmed software are validated on three test cases related to the deformation of CFD meshes inside a duct and a turbine stator row as well as around a car model.

5.1 Introduction and Literature Overview on Mesh Displacement

In CFD, the need for adapting an existing mesh to displaced boundaries arises in many applications, including aerodynamic shape optimizations, aeroelastic simulations and flow simulations in the presence of moving bodies. Mesh displacement is an alternative to re-meshing since the latter might hinder the continuation of new simulations from available numerical solutions on the unstructured mesh, for instance, of the previous domain.

Various mesh displacement methods have been proposed in the past to meet specific requirements springing from diverse types of simulations or even disciplines. These can be classified in several ways. Some of the encountered classifications are methods based on interpolations, control meshes and physical analogies [75], algebraic vs. partial differential equation methods [76], connectivity-based vs. point-based methods [77], methods for structured or unstructured meshes [78] and techniques that can more or less efficiently be parallelized [79]. Recent surveys, such as [75] and [80], enumerate the pros and cons of mesh displacement methods based on various applications and quality criteria. In detail, [80] compared six methods based on physical analogies on eight test cases, while [75] overviewed many conventional methods from a theoretical point of view.

Mesh displacement methods that have been around for a long time are spring analogies that model the mesh as a network of linear [81], torsional [82], semi-torsional [83] and ball-vertex springs [84] and solve the static equilibrium equations to find the updated nodal locations. Generally, in the linear spring analogy approach, each edge of the mesh is replaced by a tension spring with the spring stiffness taken to be inversely proportional to the edge length. The equilibrium lengths of the springs are set equal to the initial lengths of the mesh elements edges, and the known displacements at the mesh boundaries are used as boundary conditions of the model. Other methods based on the same analogy have been proposed to improve the quality and robustness of the technique: for instance, the ball-vertex method introduces supplementary linear springs that resist the motion of a vertex toward the opposite faces. These methods are efficient for meshes for structural analysis and inviscid flow simulations; [75] reported invalid elements and high computational cost to handle large displacements of CFD meshes with stretched elements for viscous flow simulations. Besides, these methods require the connectivity of the CFD mesh to be available, and extension to generic polyhedral hybrid meshes is, thus, difficult.

Continuum elastic approaches have been proposed by several authors, such as [85–87]. They displace the mesh by solving the linear elasticity equations on the mesh itself; thus, connectivity should be known. The elasticity equations contain material properties, which are related to the mesh characteristics. For instance, the modulus of elasticity can be set to be the inverse of a mesh element volume or related to its aspect ratio[1], which leads to rigid displacement of mesh elements close to the mesh boundary. Usually, the elasticity equations are discretized with a Finite Element Method (FEM) method. Laplacian methods [88] solve the Laplace PDEs to diffuse the surface mesh node displacements into the domain. The method is efficient for single-frequency deformations, although large multiple frequency deformations may lead to invalid meshes. Better quality of the displaced mesh can be achieved by solving the biharmonic smoothing equation [89], at an increased computational cost. The algebraic damping method [90] is based on the displacement of each internal mesh node in terms of the displacement of the closest node on the moving boundaries. The resulting deformation appears to be very rigid close to the boundaries but, for large mesh deformations, algebraic smoothing might be necessary to improve mesh quality. The Delaunay graph method [91] is based on the generation of a control mesh based on the Delaunay triangulation of the boundary nodes and the mapping of the internal nodes on the Delaunay graph. The triangulation is adapted to geometry changes, and volume mesh nodes are relocated through barycentric interpolation. The Inverse Distance Weighting (IDW) method [92] computes the location of internal nodes through the direct interpolation of the boundary nodal displacements using weights depending on their distance from the boundary; usually, the squared distance is used which has been found to preserve better mesh orthogonality close the boundaries. Transfinite interpolation [93] is based on the interpolation of the deformations along mesh lines, which is computationally efficient though limited to structured meshes [94].

Mesh displacement based on RBF interpolation has proved to be robust for large deformations [18]. [75] compared the most common techniques, including linear and torsional springs, linear elasticity and several interpolation-based methods such as the RBF and IDW. The paper concluded by recognizing mesh displacement based on RBF interpolation as one of the most promising approaches regarding robustness and mesh quality; its high computational cost and bad scalability can be mitigated by greedy data reduction algorithms. [95] benchmarked a mesh displacement approach based on RBF interpolation against six techniques previously tested in [80] and concluded that the former, though more expensive, yields deformed meshes of better quality. [92] compared mesh displacement based on RBF and IDW and demonstrated a reduction in the computational

[1]For the face of a polyhedral mesh element the aspect ratio is the maximum edge length squared to face area. If the area is zero the aspect ratio is considered zero.

cost of the latter, compared to the former, by a factor of 20, for a hexahedral mesh for inviscid flow simulations with ~$75,000$ cells. However, the RBF method produced slightly better mesh quality than IDW.

The inefficiency of the RBF-based mesh displacement, in its standard form, is due to the need to solve a linear system of equations with rank equal to the number of the displaced and fixed boundary nodes, besides the computation of the displacements of the internal nodes. In recent years, many researchers focused on the cost reduction of RBF interpolation. Despite the progress made in this field during the last years, the problem is still open to new strategies and improvements.

The use of RBF kernels with local support was the first breakthrough to reduce the cost of RBF interpolations [31]. They lead to sparse matrices that can be solved more efficiently. However, a trade-off between the smooth propagation of the deformation (mesh quality) and the sparsity of the matrix (computational cost) exists. For large deformations, local support must be enlarged, and the problem becomes similar to one with RBF kernels with global support, vanishing the benefits of local support [31]. This problem was alleviated in [96] by dividing the deformations into smaller steps controlled by locally supported RBF interpolations with small radii, leading to a series of very sparse linear systems to be solved. Similar methods, to be referred to as multilevel RBF techniques, involve successive levels of nested RBF interpolations through which, on each level, the solution from the previous coarser level is interpolated [97, 98]. The mesh displacement method proposed in [99] was based on domain decomposition and local RBF interpolations resulting in a series of small problems.

Greedy algorithms [78] typically start from a coarse approximation to the mesh deformation and iteratively refine it until the desired accuracy is reached. Greedy methods use a subset of the surface mesh nodes to describe the new shape, leaving the rest of the nodes for error checking; so, they are more efficient than standard RBF interpolation, although they cannot precisely reproduce all surface deformations. The iterative procedure required to guarantee the error drop to a prescribed tolerance is time-consuming for tight tolerances. [100] compared various data reduction methods both by considering the actually imposed displacements and computing the subset of surface mesh nodes a priori. [77] proposed an agglomeration strategy of the boundary nodes as in multigrid methods. [101] suggested an incremental least-squares solver, which, similarly to greedy algorithms, uses a subset of the surface mesh nodes to approximate the deformation. [102] proposed the use of a multiscale RBF interpolation with multiple support radii to capture deformations at different scales. The interpolation matrix is built starting from a coarse subset of source nodes. The algorithm proceeds by iteratively adding the remaining source nodes using a support radius such that the newly added nodes do not affect the previous, ending up with an easier to solve linear system. [78] suggested a correction step such as a Delaunay graph mapping, after the approximation step; however, locally supported RBF interpolation appears to be a better choice from the quality and robustness point of view [103].

This chapter introduces a two-step cost reduction technique for mesh displacement, Section 5.3. The two steps successively deform the mesh based on RBF interpolation, the principles of which are outlined in Section 5.2. The interpolation is further accelerated thanks to the SPAI preconditioner, Section 5.4.1, the FMM, Section 5.4.2, and an integer lattice-based method, Section 5.4.3.

The proposed strategy and the programmed software are used on three benchmark test cases, Section 5.5. In Section 5.5.1, the mesh quality resulting from the proposed method is compared with standard RBF. In Section 5.5.2, the scalability of the software is analyzed in order to make it efficient in cases with huge mesh sizes. In Section 5.5.3, the performance of the software is investigated by varying the main input parameters. Finally, in Section 5.5.4, the effectiveness of the proposed mesh displacement method to large displacements is tested by using it in evolutionary algorithm-based optimization.

5.2 Background of RBF-Based Interpolation

The theory regarding RBF interpolation was already given in Section 2.2.3; herein, the method is extended with more notation, and issues arising from the different use of the mathematical tool are highlighted. An RBF network is a weighted linear combination of RBF kernels interpolating scattered data in the Q-dimensional space. In mesh displacement, in specific, the quantities to be interpolated are the known 3D displacements at the K surface mesh nodes or, generally, at K distinct source nodes. A 3D RBF kernel $\phi_k(\boldsymbol{x}) = \phi(r = ||\boldsymbol{x} - \boldsymbol{x}_k||)$ is a real-valued function depending on the distance r of a point $\boldsymbol{x} \in \mathbb{R}^3$ from the so-called RBF interpolation source $\boldsymbol{x}_k \in \mathbb{R}^3$. $||.||$ stands for the Euclidean distance.

To interpolate the given displacements $\boldsymbol{\delta}_k$, $1 \leq k \leq K$, K RBF kernels centered at the respective nodes must be used. The RBF interpolant $\boldsymbol{d} : \mathbb{R}^3 \to \mathbb{R}^3$ takes the form:

$$\boldsymbol{d}(\boldsymbol{x}) = \sum_{k=1}^{K} \boldsymbol{w}_k \phi_k(\boldsymbol{x}), \tag{5.1}$$

where the weights $\boldsymbol{w}_k \in \mathbb{R}^3$ are computed so as to exactly reproduce the known displacements $\boldsymbol{d}(\boldsymbol{x}_k) = \boldsymbol{\delta}_k$ at the K source nodes; this requires the numerical solution of a $K \times K$

linear system, with different right-hand side (r.h.s.) arrays.

Equation 5.1 is often modified by adding a polynomial term to preserve affine motion, i.e., translation, rotation and scaling [32, §6]. In 3D, the RBF interpolant, including a degree-one polynomial, takes the following form:

$$\boldsymbol{d}_\pi(\boldsymbol{x}) = \sum_{k=1}^{K} \boldsymbol{w}_k \phi_k(\boldsymbol{x}) + \boldsymbol{a}_0 + \boldsymbol{a}_1 x + \boldsymbol{a}_2 y + \boldsymbol{a}_3 z, \tag{5.2}$$

where $\boldsymbol{a}_q \in \mathbb{R}^3,\ 0 \leq q \leq 3$ are the polynomial coefficients. To compute the new degrees of freedom, new conditions are introduced; assuming that the sources are not co-planar these conditions are:

$$\sum_{k=1}^{K} \boldsymbol{w}_k = 0 \quad \text{and} \quad \sum_{k=1}^{K} \boldsymbol{w}_k \odot \boldsymbol{x}_k = \boldsymbol{0}\,, \tag{5.3}$$

where $\boldsymbol{0} \in \mathbb{R}^3$ denotes the zero vector and $\odot$ the entry-wise product operator. Then, if

$$\begin{gathered}\boldsymbol{\Phi} = \begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_K(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_K) & \cdots & \phi_K(\boldsymbol{x}_K) \end{bmatrix}, \quad \boldsymbol{P} = \begin{bmatrix} 1 & \boldsymbol{x}_1^\top \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_K^\top \end{bmatrix}, \\ \boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_1^\top \\ \vdots \\ \boldsymbol{w}_K^\top \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_0^\top \\ \vdots \\ \boldsymbol{a}_3^\top \end{bmatrix}, \quad \boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\delta}_1^\top \\ \vdots \\ \boldsymbol{\delta}_K^\top \end{bmatrix}. \end{gathered} \tag{5.4}$$

the $(K + 4) \times (K + 4)$ linear system to be solved is

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{P} \\ \boldsymbol{P}^\top & \boldsymbol{0} \end{bmatrix} \begin{pmatrix} \boldsymbol{W} \\ \boldsymbol{A} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Delta} \\ \boldsymbol{0} \end{pmatrix}. \tag{5.5}$$

For large K values, the computation of $\boldsymbol{W}$ and $\boldsymbol{A}$ by solving equation 5.5 becomes expensive. It exhibits poor scalability if implemented naively, due to both the complexity of linear solvers and its stiffness. Solving equation 5.5 is referred to as the training phase of the interpolation, whereas computing the displacements $\boldsymbol{d}_\pi(\boldsymbol{x})$ for all mesh nodes or targets, by equation 5.2, is the interpolation phase. More about the theory on the solvability of these types of interpolations can be found in [104].

The behavior of the RBF interpolation is profoundly influenced by the chosen kernel ϕ [31], being of either local or global support. The former requires the definition of a local support radius r_s, which determines the region of influence of the kernel around each source node. Since $\phi(r) \neq 0$ if and only if $r < r_s$, displacements imposed on the source nodes affect only mesh nodes lying inside the region of influence of the corresponding kernel. A low-valued support radius leads to a better conditioned and sparser matrix $\mathbf{\Phi}$, whereas deformation is dissipated over a smaller portion of the interior mesh. By appropriately selecting the kernel $\phi(r)$, see Section 5.3, matrix $\mathbf{\Phi}$ becomes symmetric and positive definite. However, the block-matrix on the left-hand side of equation 5.5 is generally non-positive definite.

5.3 The Two-Step Strategy

The proposed two-step strategy divides the mesh displacement problem into two successive, both RBF-based, interpolation steps:

- In the first step (predictor), all mesh nodes (surface and interior) are targets of a global RBF interpolation, and a coarsened set of source nodes is generated by a data reduction method. The latter takes both the spatial distributions of mesh nodes and the displacement field to be interpolated into account. In this step, the interpolant takes the form of equation 5.2, including degree-one polynomial terms. After displacing the entire mesh, though, the boundary nodes do not generally respect the known displacements.
- The second step (corrector) corrects the position of the surface mesh nodes through local deformations. All surface mesh nodes become sources, and only the internal nodes in a small volume close to the surface become targets. In this step, the RBF interpolant takes the form of equation 5.1.

The two steps are more manageable than the original problem. In fact, the first step generates a "small" but dense coefficient matrix (its rank might be by orders of magnitude lower than the number of surface nodes) whereas the second step generates a "big" (rank equal to the number of surface nodes), though very sparse, matrix. This strategy allows for a noticeable reduction in the computational cost of displacing a mesh.

5.3.1 Step 1: Predictor

The predictor is based on data reduction according to which the source nodes set is coarsened by clustering. The objective is to generate a reduced set of sources that are representative of the displacement field of the surface mesh. For this purpose, an adaptive octree data-structure that recursively splits the Cartesian space is employed. By considering the surface mesh nodal density and the spatial gradient of the displacements, more sources are generated in areas of rapid variation in the imposed surface nodes displacements. In detail, the algorithm starts from a single parent box containing all surface mesh nodes. Each parent box is recursively split based on the number of contained surface nodes (to ensure proper resolution in densely populated zones) and the maximum difference in the displacements of the contained surface nodes (to ensure appropriate resolution where the displacement field rapidly changes). Empty boxes are ignored, and parent boxes, the division of which would yield only one child box, are subdivided irrespective of the above criteria until a maximum depth limit is reached. The centers of leaf (childless) octree boxes are used as RBF interpolation sources in the predictor training phase. Each source takes on the averaged displacement of the surface mesh nodes contained in the corresponding leaf box of the octree. Figures 5.1 and 5.2 illustrate, respectively, an example of selected interpolation sources on the CFD surface mesh of a duct and the octree structure used to identify such sources.

The approximation to the displacement introduced by the predictor is measured by the nodal error, which is defined at each (i^{th}) surface mesh node $\boldsymbol{x}_i$, as:

$$E_i = \sqrt{\Delta\boldsymbol{x}_i^{\mathsf{T}}\Delta\boldsymbol{x}_i}\,, \tag{5.6}$$

where, in the predictor step, $\Delta\boldsymbol{x}_i = \boldsymbol{\delta}_i - \boldsymbol{d}_{\pi,\text{Predictor}}(\boldsymbol{x}_i)$ is the difference between the known displacement $\boldsymbol{\delta}_i$ of the i^{th} surface mesh node and that computed within the predictor step, $\boldsymbol{d}_{\pi,\text{Predictor}}(\boldsymbol{x}_i)$. In contrast to greedy methods that usually minimize the surface error norm (summing up all surface nodal errors), in the proposed method, any deviation from the known boundary displacements is resolved in the corrector step. The role of the predictor step is to generate a reduced set of source points and approximate the displacement field.

A global support RBF kernel is chosen by considering characteristics such as mesh quality preservation [78], flop count and condition number of the linear system to be solved. In the predictor step, the inverse multi-quadric kernel, Table 2.1, is used, where σ is the parameter regulating the decay of the kernel, selected to be equal to half of the mesh bounding box diagonal length.

The linear systems (one per Cartesian direction) in equation 5.5 (including the polynomial term), assembled with the reduced set of RBF sources, are solved by an iterative method. Since the coefficient matrix in equation 5.5 is non-positive definite, the selected solver is based on the Bi-Conjugate Gradient Stabilized (BiCGStab) algorithm (Appendix F), coupled with the SPAI preconditioner. The iterative method solves systems with multiple r.h.s. arrays (the displacements along the three Cartesian directions) at once using parallel matrix-matrix multiplications. After solving equation 5.5, the displacement field is

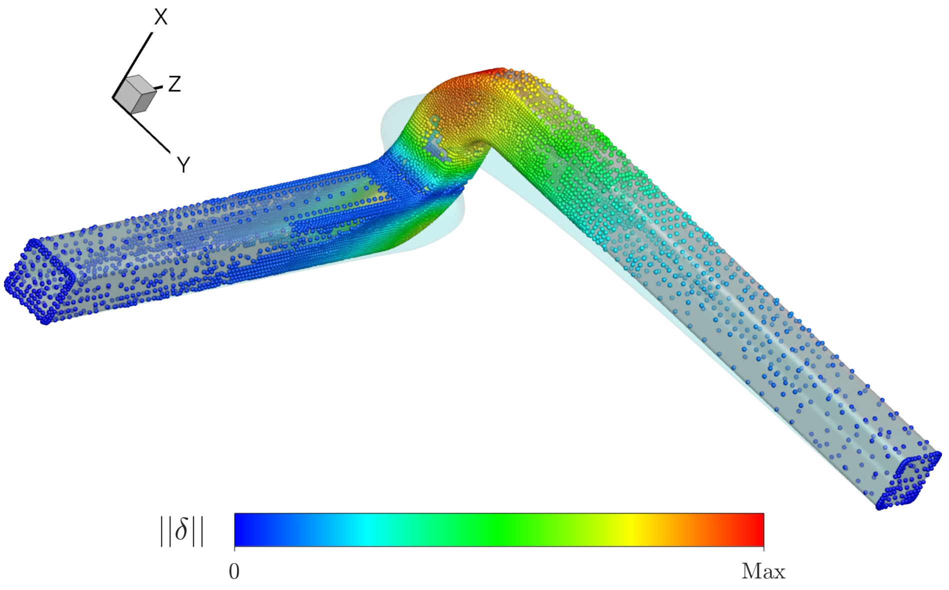


Figure 5.1: Double elbow duct. RBF sources (spheres) generated by the data reduction algorithm in the predictor step for the duct studied in Section 5.5. Colors from blue to red represent small to large displacements $\boldsymbol{\delta}$. The reference and displaced duct shapes are depicted in gray and blue, respectively (see also Figure 5.10). Most RBF sources lie in the area of the high spatial gradient of the displacements. The computed sources do not necessarily lie on the mesh surface.

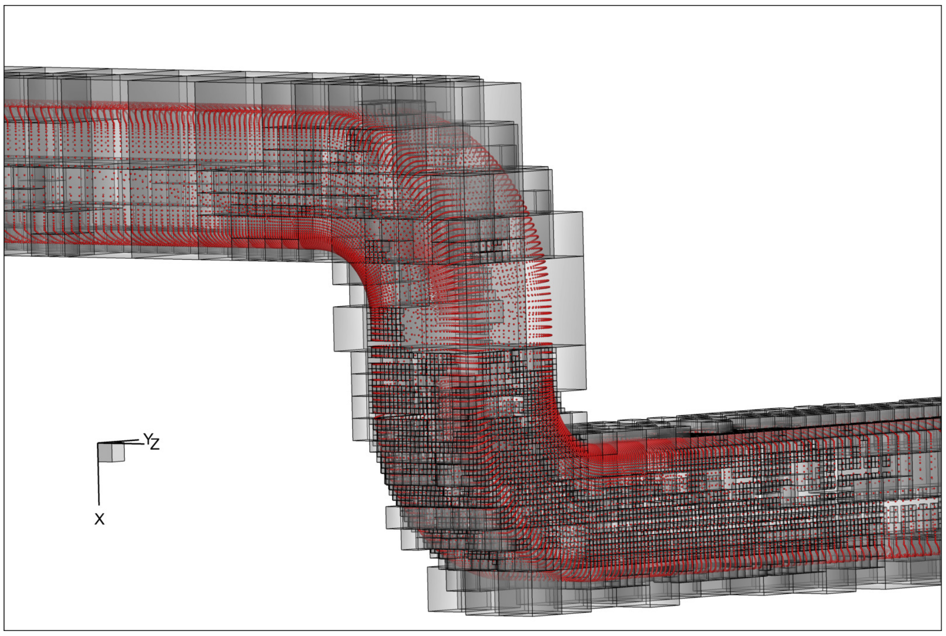


Figure 5.2: Double elbow duct. Octree used by the data reduction algorithm in the predictor step for the duct studied in Section 5.5. The barycenters of the leaf bins are used as RBF sources. The surface mesh nodes are depicted in red.

obtained by evaluating equation 5.2 at all mesh nodes. The FMM, Section 5.4.2, is used to speed up this evaluation.

5.3.2 Step 2: Corrector

The corrector step is based on a local RBF interpolation method. The kernel used in this step is the Wendland C0 function, Table 2.1. A tradeoff among the smooth propagation of the deformations in the volume mesh, computational cost and memory requirements, depending on the choice of the support radius, is expected. In the corrector, the interpolation sources coincide with the surface mesh nodes with prescribed displacements. Since the predictor has already displaced the surface mesh nodes close to their known target positions, the remaining surface displacements $\Delta \boldsymbol{x}_i$ are minor. As a rule of the thumb, the support radius should be at least three times larger than the largest error E_i of all surface nodes.

The linear system in equation 5.5 (without the polynomial term, this time) is solved by taking into account the new nodal positions computed by the predictor and imposing the displacements on the surface nodes to be equal to the already computed differences $\Delta \boldsymbol{x}_i$. The system is sparse and positive definite for the Wendland C0 activation function; however, due to the non-symmetric SPAI preconditioner, the BiCGStab algorithm is used. Then, the displacement field is obtained by evaluating equation 5.1. The method presented in Section 5.4.3 is used to speed up this evaluation.

5.4 Acceleration Methods for the Two-Step Strategy

Techniques to efficiently carry out the training and interpolation phases, in both the predictor and corrector steps, are proposed. These methods are:

- The Sparse Approximate Inverse preconditioner [105] for the acceleration of the training phase in both steps.
- The Fast Multipole Method [106] for the acceleration of the interpolation phase in the predictor step.
- An integer lattice-based technique for the acceleration of the interpolation phase in the corrector step.

5.4.1 The SPAI Preconditioner

Preconditioning is essential to quickly solve the linear system in equation 5.5, during the training phase in both steps. Recently, preconditioning techniques based on the SPAI have been developed [107]. Their main advantage is that they are inherently parallel since many independent small linear systems must be solved. SPAI preconditioners work properly for a wide range of applications [108] and are immune to numerical difficulties such as pivot breakdowns as well as instabilities, which might occur in incomplete LU (ILU) [109].

In the literature, SPAI preconditioners are applied to sparse linear systems, although a few applications with dense systems can also be found. For instance, [110] compared diagonal, symmetric successive overrelaxation, ILU and SPAI preconditioners coupled with a Generalized Minimal Residual (GMRES) solver and found that the SPAI preconditioner gives better convergence rates, for a dense matrix arising from the discretization of an electromagnetic scattering problem.

The SPAI preconditioner $\boldsymbol{M}$ is a sparse approximate inverse of a sparse approximation to $\boldsymbol{\Phi}$ (equation 5.4). The method assumes that a sparse matrix can effectively approximate the inverse of a full (predictor) or sparse (corrector step) matrix. Recall that the inverse of a sparse and, certainly, a dense matrix is generally dense. Nevertheless, for a decaying RBF kernel, many of the entries in $\boldsymbol{\Phi}$ are small, and many of the entries in $\boldsymbol{\Phi}^{-1}$ are also expected to be small [111]. All these small entries are neglected, so that sparse (or sparser, if $\boldsymbol{\Phi}$ is already sparse) approximations to $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}^{-1}$ are employed.

The computation of the preconditioner $\boldsymbol{M}$ requires the minimization of the Frobenius norm[2]:

$$\min_{\boldsymbol{M}} ||\boldsymbol{S}\boldsymbol{M} - \boldsymbol{I}||_F^2\,, \tag{5.7}$$

where $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{S}$ is a sparse matrix approximating $\boldsymbol{\Phi}$; $\boldsymbol{S}$ is formed by the largest entries in $\boldsymbol{\Phi}$. In equation 5.7, the Frobenius norm $||.||_F$ is employed since it allows decomposing the minimization problem into K (rank of $\boldsymbol{\Phi}$) independent linear problems. In fact, a property of the Frobenius norm allows splitting it into a sum of Euclidean norms [112]:

$$||\boldsymbol{S}\boldsymbol{M} - \boldsymbol{I}||_F^2 = \sum_{k=1}^{K} ||\boldsymbol{S}\boldsymbol{m}_k - \boldsymbol{e}_k||_2^2\,, \tag{5.8}$$

where $\boldsymbol{m}_k$ is the k^{th} column of $\boldsymbol{M}$, and $\boldsymbol{e}_k$ is the k^{th} row of $\boldsymbol{I}$. Each summand in equation 5.8 constitutes a linear system to be solved, the rank of which is reduced based on the sparsity of $\boldsymbol{S}$ and $\boldsymbol{M}$. For this reason, $\boldsymbol{S}$ and $\boldsymbol{M}$ are subject to sparsity constraints to balance the quality of the preconditioner (the preconditioned system should converge rapidly) and its construction and application time. A complete overview of the theory of the SPAI preconditioner is provided in [105].

[2]The Frobenius norm is of a matrix is defined as the square-root of the sum of the absolute squares of its elements.

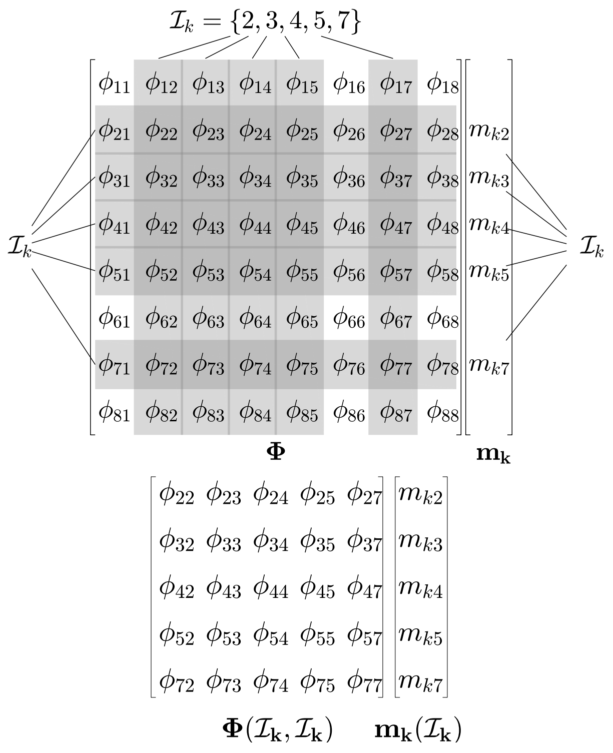

$\mathcal{I}_k = \{2,3,4,5,7\}$
$\mathcal{I}_k$
$\mathcal{I}_k$
$\mathbf{\Phi}$
$\mathbf{m_k}$
$\mathbf{\Phi}(\mathcal{I}_\mathbf{k}, \mathcal{I}_\mathbf{k})$
$\mathbf{m_k}(\mathcal{I}_\mathbf{k})$

Figure 5.3: Graphical representation of the definition of the sub-matrix $\mathbf{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ to solve the linear system of each (k^{th}) summand of equation 5.8. The integer lattice determines the set of indices $\mathcal{I}_k$ (see example in Figure 5.4). Top: full matrix $\mathbf{\Phi}$ and sparse k^{th} column $\boldsymbol{m}_k$ of the preconditioner $\boldsymbol{M}$. Bottom: the reduced matrix $\mathbf{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ that is solved to find $\boldsymbol{m}_k(\mathcal{I}_k)$.

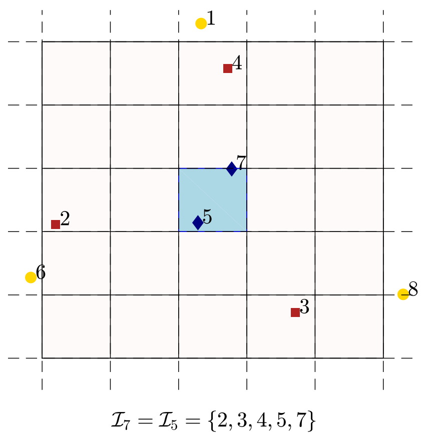


Figure 5.4: Regular Cartesian 2D integer lattice built over a cloud of sources. Square- and diamond-sources are the third-level lattice neighbors of the diamond-sources. Round marks are considered as far-away sources and, therefore, excluded from the neighbors of the diamond-sources. The integer lattice is used to define a priori the sparsity pattern of the SPAI preconditioner, i.e., sets of indices $\mathcal{I}_k$. The two diamond sources yield the same set of indices $\mathcal{I}_k$. In 3D, the lattice is formed by regular Cartesian cubes.

The criteria for selecting the sparsity (non-zero) pattern of $\boldsymbol{M}$ are discussed below. The strategy is to maintain low the number of non-zero entries in $\boldsymbol{M}$ while capturing the largest entries in $\boldsymbol{\Phi}^{-1}$ that are the most significant regarding the quality of the preconditioner. Reliable results can be obtained using a precomputed sparsity pattern [113]. Thus, strategies to identify a suitable sparsity structure for $\boldsymbol{M}$ dynamically are not used. They have general applicability but are time-consuming, and the procedure to identify the preconditioner sparsity structure cannot easily be parallelized [105, 109, 114].

By first assuming that identical sparsity patterns for $\boldsymbol{M}$ and $\boldsymbol{S}$ are given, it is possible to define a set of indices pinpointing the non-zero entries in each column of the preconditioner $\boldsymbol{m}_k$, as follows:

$$\mathcal{I}_k = \{\forall j \in [1, K] \text{ s.t. } \boldsymbol{m}_k(j) \neq 0\}\,. \tag{5.9}$$

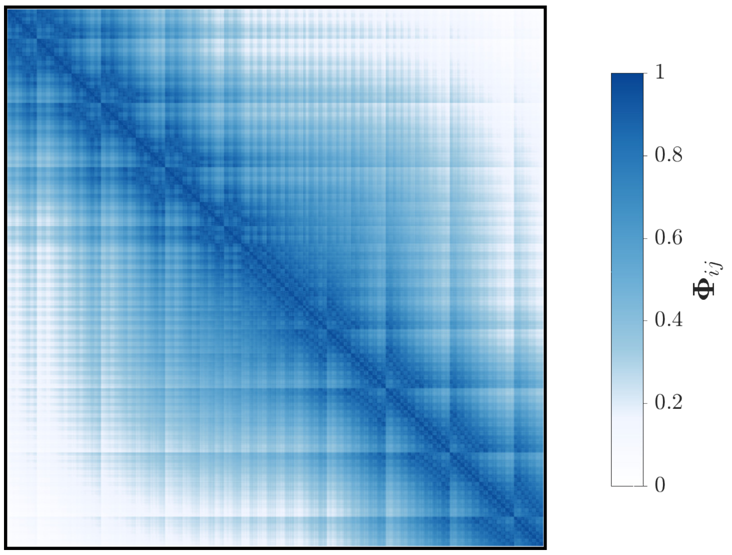
Then, $\mathcal{I}_k$ identifies the columns of $\boldsymbol{\Phi}$ to be kept in the linear system of the k^{th} summand of equation 5.8. $\mathcal{I}_k$ also identifies the rows of $\boldsymbol{\Phi}$ that are kept, in order to reduce the number of rows of each linear system in such summands. The retained non-zero entries in each column of the preconditioner $\boldsymbol{m}_k$ are computed by solving the following K linear systems:

$$\boldsymbol{\Phi}(\mathcal{I}_k, \mathcal{I}_k)\boldsymbol{m}_i(\mathcal{I}_k) = \boldsymbol{e}_i(\mathcal{I}_k). \tag{5.10}$$

in which the coefficient matrices are square, symmetric and positive definite. Figure 5.3 illustrates graphically how the sub-matrix $\boldsymbol{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ is formed by a known set of indices $\mathcal{I}_k$ (as in the example of Figure 5.4) to compute the entries of the k^{th} column of the preconditioner. With a sparser matrix $\boldsymbol{M}$, smaller sub-matrices $\boldsymbol{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ are derived so that Cholesky factorization can efficiently solve the linear systems of equation 5.10, to compute $\boldsymbol{m}_k$. In this regard, if the indices of the rows of $\boldsymbol{\Phi}$ forming the sub-matrices in equation 5.10 were not the same with the indices of the columns, QR decompositions would be needed being about four times more costly than Cholesky.

Using the same sparsity pattern for $\boldsymbol{S}$ and $\boldsymbol{M}$ is justified by the fact that the position of the large entries in $\boldsymbol{\Phi}^{-1}$, which contribute the most to the quality of the preconditioner, tend to be at the location of the large entries in $\boldsymbol{\Phi}$. This is theoretically supported by the work presented in [111]: for a banded positive definite matrix, its inverse entries decay exponentially away from the bands. This behavior is illustrated for two RBF coefficient matrices in Figures 5.5 and 5.6.

The description of the strategy used to a priori define the sparsity pattern of $\boldsymbol{S}$ and $\boldsymbol{M}$, namely the sets of indices $\mathcal{I}_k$, $1 \leq k \leq K$, previously assumed to be given, follows. The sparsity pattern is the sets of indices $\mathcal{I}_k$ identifying the entries with the highest value in



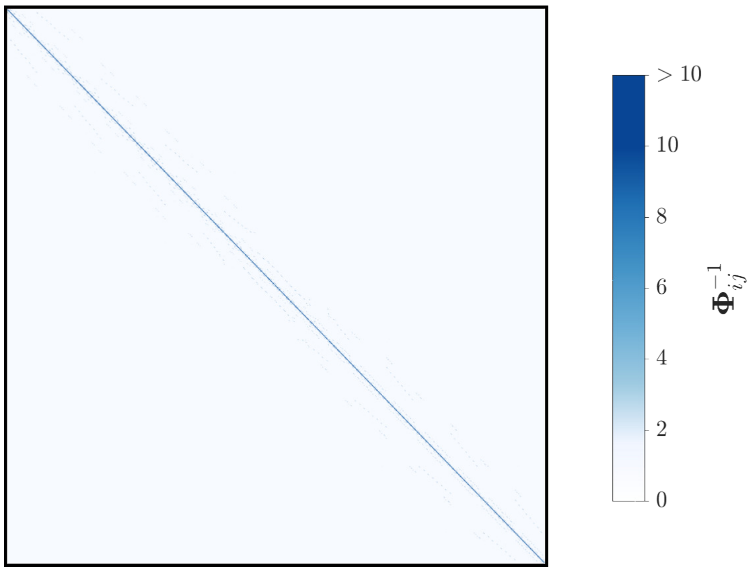
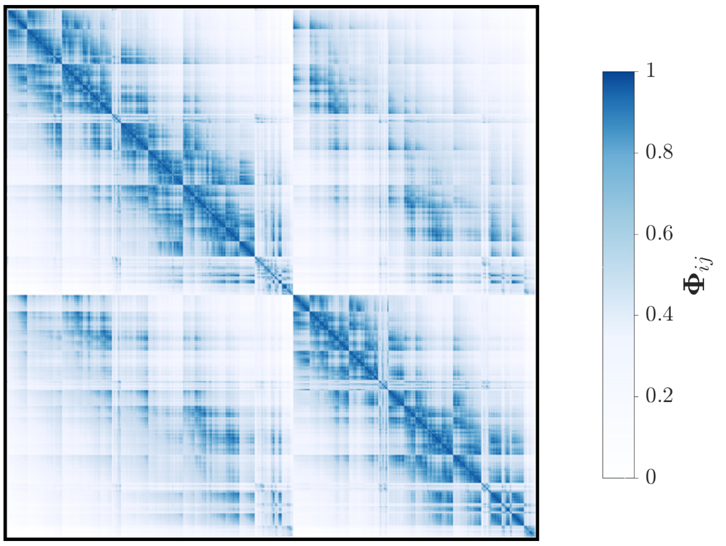


Figure 5.5: Double elbow duct. Patterns of an RBF training matrix $\mathbf{\Phi}$ (top) and its inverse $\mathbf{\Phi}^{-1}$ (bottom) of the predictor. $\mathbf{\Phi}$ is originated from the reduced set of sources of the duct case of Figure 5.1. Both matrices are symmetric, and their rank is ~10^4. Large to small entries are depicted in blue to white. The range $(0, 1]$ of the entries of $\mathbf{\Phi}$ corresponds to the image of the multiquadric activation function. In $\mathbf{\Phi}^{-1}$, only entries close to the diagonal (but not just the diagonal itself) are predominant since the largest entries in $\mathbf{\Phi}$ are close to the diagonal.



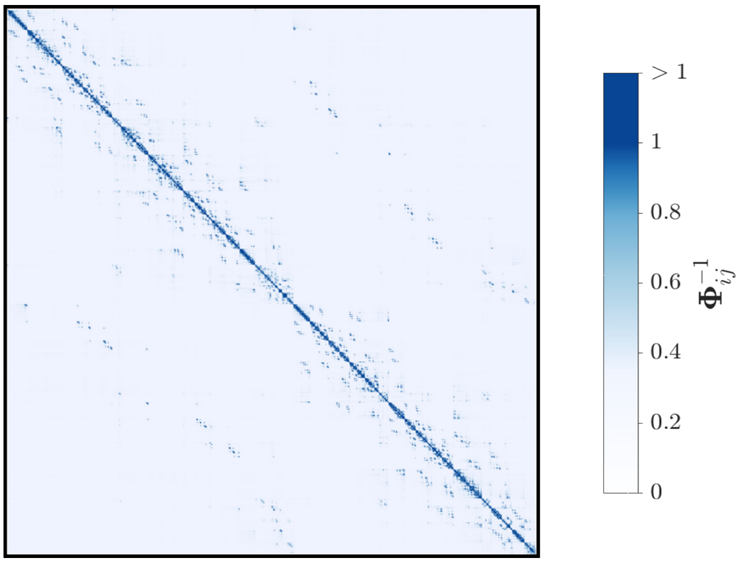


Figure 5.6: RRD Turbine stator. Patterns of an RBF training matrix $\mathbf{\Phi}$ (top) and its inverse $\mathbf{\Phi}^{-1}$ (bottom) of the predictor. $\mathbf{\Phi}$ is originated from the reduced set of sources of the reference shape of the turbine stator case of Figure 5.12. Both matrices are symmetric, and their rank is ~10^4. Large to small entries are depicted in blue to white. The range $(0, 1]$ of the entries of $\mathbf{\Phi}$ corresponds to the image of the multiquadric activation function. In $\mathbf{\Phi}^{-1}$, predominant entries far from the diagonal also match the predominant entries of $\mathbf{\Phi}$.

each row of $\mathbf{\Phi}$. For each RBF interpolation source, the largest entries in $\mathbf{\Phi}$ arise from the other sources in the neighborhood, due to the behavior of the RBF kernel. The sets $\mathcal{I}_k$ are, then, formed by the indices of the rows generated by the neighbor sources, including the k^{th} source itself. To facilitate the neighbors' searching procedure, a regular Cartesian integer lattice tessellating the space is constructed. The neighbors of each RBF source are defined through the lattice neighbors. Several levels of lattice neighbors can be determined. First-level neighbors are all nodes in the box containing the source itself. Second-level ones are nodes contained in the neighboring boxes of the first-level box in addition to the first-level itself and so forth. In Figure 5.4, a 2D explicative lattice is illustrated with three levels of neighbors. The distance between lattice points and the number of levels that define the neighbor's nodes are used to adjust the sparsity pattern of $\boldsymbol{S}$ and $\boldsymbol{M}$.

Using the integer lattice to compute a priori the sparsity pattern, has a beneficial effect. In fact, all nodes in a lattice box have the same set of neighbor nodes $\mathcal{I}_k$. The same set of neighbor nodes $\mathcal{I}_k$ leads to the same reduced matrix $\mathbf{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$, which is factorized only once to build all the columns of the preconditioner corresponding to the nodes contained in the lattice box having the $\mathcal{I}_k$ set of neighbors; this reduces significantly the number of Cholesky decompositions needed.

Figure 5.7 illustrates the time required for the linear solver to converge, in the training phase of the predictor step of the duct case (Section 5.5.1), including the set-up time for various preconditioners. The SPAI preconditioner reduces the time needed to converge to a reasonably accurate solution by one order of magnitude. This saving in time becomes higher for larger matrices. The influence of the neighbor grouping strategy based on the integer lattice is also illustrated: for the preconditioner with density 5%, the grouping strategy reduces the number of decompositions needed from $\sim 10^4$ to just $\sim 7 \times 10^2$, reducing the set-up time by a factor of approximately 15.

As previously mentioned, the SPAI preconditioner is non-symmetric, and a solver for non-symmetric matrices (BiCGStab) must be used. Since $\mathbf{\Phi}$ is a symmetric positive definite matrix, a symmetric positive definite preconditioner was also investigated to employ faster iterative solvers. One of them was Conjugate Gradient (CG), used in the corrector step and not in the predictor, due to the polynomial terms, which make the coefficient matrix non-positive definite. The Factorized Sparse Approximate Inverse (FSPAI) preconditioner [105] (positive definite) would exhibit much better performances compared to the SPAI preconditioner if the strategy to reduce the number of sub-matrices decompositions was not included. Since a sub-matrix must be factorized for each column of the preconditioner (factorizations cannot be re-used to build multiple columns), the set-up time of the FSPAI

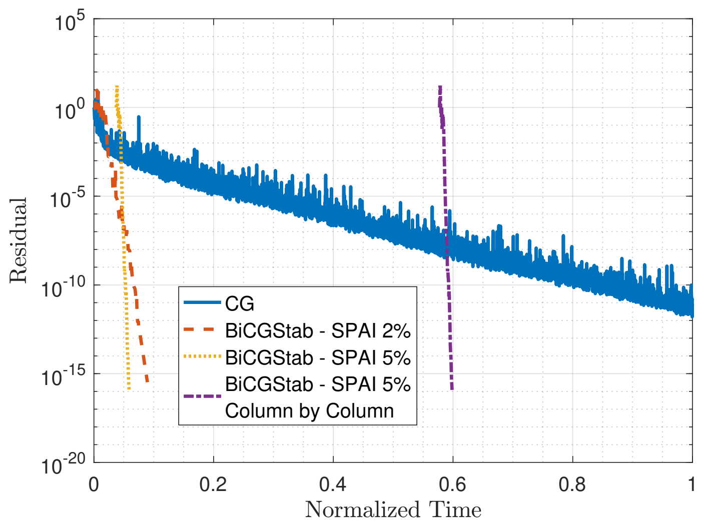


Figure 5.7: Double elbow duct. History of the residual of the full linear system (rank ~10^4) assembled from the reduced set of RBF sources (Figure 5.1) without polynomial terms for various combinations of iterative solvers and SPAI preconditioners plotted as a function of normalized time. For the sake of fairness, the set-up time for the preconditioners, which appears as a delay before the solvers take over, is also considered. Percentages in the legend refer to the density (which is equal to 1 minus the sparsity of the matrix) of the preconditioners. The non-preconditioned CG solver does not have any set-up time, but the convergence rate is severely affected by ill-conditioning. Two different SPAI preconditioners were used with different densities, to demonstrate that a correlation exists between density and quality but, of course, a denser preconditioner costs more. The preconditioner named “Column by Column” is built without the neighbor’s grouping strategy based on the integer lattice: the preconditioners built with the grouping strategy have similar performance and much lower set-up time.

becomes predominant compared to the time for solving the system, similarly to the “SPAI column by column” in Figure 5.7. On the other hand, methods for the symmetrization of SPAI can be used in conjunction with solvers for symmetric non-positive definite matrices. For instance, [115] suggested a symmetrization strategy in conjunction with the Symmetric Quasi-Minimal Residual (SQMR) method, finding better convergence rates that non-symmetric SPAI used in combination with GMRES.

5.4.2 The Fast Multipole Method

The FMM is used to speed up the interpolation phase of the predictor step. Its role is to accelerate the computation of summations involved in equation 5.1; the FMM approximates the $\boldsymbol{\Phi W}$ product, whereas $\boldsymbol{PA}$ and $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}$, which appear in equation 5.2, are considered separately. The FMM was initially developed in [116] to approximately solve N-body gravitational and electrostatic potential simulations, within any user-defined precision, with runtime complexity $O(N)$ instead of $O(N^2)$ of the direct computation. The same algorithm is applied to RBF interpolations to compute interactions of RBF kernels [117], thus reducing also the computational complexity of the RBF interpolation. The theory and performances of the FMM are described in Chapter 6. The FMM computes the displacement of each target node considering the real contribution of the nearby source mesh nodes and low-rank approximations for the remaining far-away contributions. In this work, the black-box Fast Multipole Method (bbFMM) [106] is adopted; according to this, the low-rank approximation is based on polynomial interpolation on Chebyshev nodes implemented to simultaneously compute displacements along the three Cartesian directions.

There is a trade-off between computational complexity and approximation error and whether this approach becomes advantageous depends not only on the mesh size but also on the minimum nodal distance, which determines the maximum allowed error due to the approximations made by FMM. In fact, the risk is to introduce a significant error in the interpolated displacements that could damage mesh quality. An accuracy analysis is also reported in Chapter 6.

The matrix-matrix products required by the BiCGStab iterative solver applied to equation 5.5 were replaced by the FMM to take advantage of the lowered multiplication complexity. However, issues regarding the accuracy and convergence of the BiCGStab solver arose: nearly exact matrix-matrix products were required for the solver not to converge to a wrong solution, [117]. The approach proposed in [118], based on nested GMRES solvers employing FMM-based matrix-matrix multiplication approximations with different accuracies, was investigated to remedy the issues of the BiCGStab solver using the FMM-based matrix-matrix multiplication. In this nested scheme, the inner solver was used as a preconditioner for the outer solver. The inner solver was using FMM-based matrix-matrix multiplications with reduced accuracy (low cost), whereas the outer solver was using accurate ones (at a higher cost). The inner solver was preconditioned by SPAI. Even if this approach worked well, the cost has increased, concluding that such an approach seems attractive only for huge dense linear systems, which is not the case in either step of the proposed method.

5.4.3 Integer Lattice-Based RBF Interpolation

The FMM is not used in the corrector since the piecewise definition of the locally supported RBF kernel makes the low-rank approximation by the bbFMM unreliable. Instead, a faster method is proposed for the exact interpolation of RBF networks with small local support radii. A neighbors' search procedure based on an integer lattice is used to accelerate

Table 5.1: Double elbow duct. Quality metrics for the reference and displaced meshes, using standard RBF and the proposed two-step strategy. Lower aspect ratios are favorable regarding CFD solution accuracy.

	Reference	Standard RBF	Two-Step Strategy
Max. aspect ratio	6574	6810	6597
Avg. aspect ratio	4.12	4.02	4.02
Min. cell volume	2.87×10^{-11}	8.99×10^{-12}	1.45×10^{-11}

Table 5.2: Double elbow duct. Maximum and average surface error norm E_i for the reference and displaced CFD surface mesh illustrated in Figures 5.1 and 5.10. The first column lists the values of the error function prior to mesh displacement. The column labeled "Predictor Step" gives the same norms upon completion of this step. The corrector merely drops both error norms to zero.

	Reference	Predictor Step
Max. E_i	2.01×10^{-1}	1.77×10^{-3}
Avg. E_i	5.76×10^{-2}	7.68×10^{-5}

the interpolation phase of the corrector step by avoiding the computation of zero-valued summands in equation 5.1. The integer lattice used in this section is similar to the one explained in Section 5.4.1 (see also Figure 5.4) but contains both sources and targets and is scaled so that the distance between lattice points be equal to the local support radius r_s of the RBF network. Then, the contributions from the sources on the first two level lattice neighbors are the only ones that must be evaluated to compute the displacement of the target nodes in each lattice box. In fact, due to the local support of the RBF kernel, only sources within a radius r_s from a target node contribute to its displacement.

This strategy reduces the interpolation cost by about two orders of magnitude for the duct and RRD turbine stator cases of Section 5.5. Moreover, this procedure is readily amenable to parallelization since the computed displacements are independent of each other. Additionally, the same neighbors' search procedure is used to reduce the coefficient matrix filling time in equation 5.5.

5.5 Parametric Studies and Results

Below, four tests, on three 3D CFD meshes, are reported. In Section 5.5.1, a comparison of mesh displacement performed with the standard RBF model, and the proposed method is presented for a double elbow duct. The same double elbow duct and a turbine stator, named RRD, are used in Section 5.5.2 to assess the software performance and scalability w.r.t.

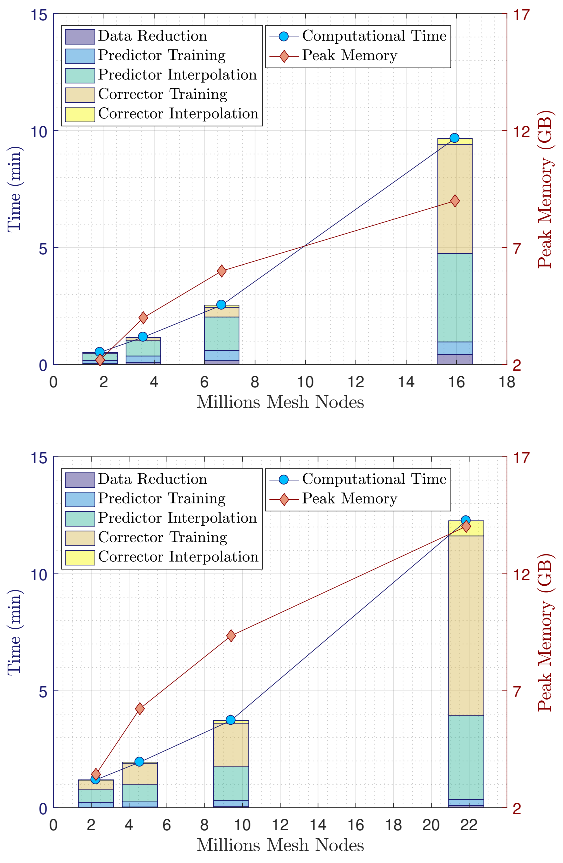


Figure 5.8: Wall-clock time and RAM requirements for the double elbow duct (top) and RRD turbine stator (bottom) mesh displacements, for different mesh sizes. Cost is broken down into five main steps in the bar chart: data reduction, predictor training and interpolation (Section 5.3.1) as well as corrector training and interpolation (Section 5.3.2). The bar chart and computational time refer to the left vertical axis, whereas the peak memory curve points to the right one.

mesh sizes. In Section 5.5.3, the performance of the software while varying the predictor step size is investigated for the displacement of a polyhedral mesh, with up to 22-face polyhedral cells. Finally, in Section 5.5.4, the effect of increasing deformations on the mesh quality of the turbine stator CFD mesh is investigated in the frame of evolutionary algorithm-based optimization. In all cases, mesh quality assessment is based on commonly used metrics such as the maximum skewness[3] and the minimum orthogonality[4]. The goal is to minimize the degradation of mesh quality after the displacement. Performance is measured on a computational node with two 6-core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz processors.

[3]The skewness of two faces of a mesh element is computed as the dot product of the two face unit normals.

[4]The orthogonality of a face of a mesh element is computed as one minus the absolute value of the dot product of two unit vectors which point in the direction of two adjacent edges of the face.

5.5.1 Two-Step Strategy vs. Standard RBF Interpolation

The quality of the mesh within a double elbow duct, deformed using the programmed software, is compared with the outcome of standard RBF with the inverse multiquadric activation function. The displacement is representative of a shape variation that could occur in evolutionary-based shape optimizations. The mesh, having ~2.50×10^6 nodes, is suitable for viscous flow simulations and is comprised of tetrahedra, pyramids, prisms and hexahedra. Reference and final shapes are illustrated in Figure 5.10. Table 5.1 presents quality metrics for the reference and displaced meshes, which indicate the ability of the proposed method to achieve results of better quality, for the considered case and metrics, w.r.t. standard RBF interpolation. This is justified by the fact that, with the two-step strategy, the deformation is interpolated sequentially, which has a beneficial effect on the mesh quality preservation [31]. In fact, the maximum cell aspect ratio has increased just by 0.3% using the two-step strategy, against the 3.6% increase by the standard model; the average aspect ratio has increased by 2.4% using either method.

Table 5.2 and Figure 5.10 illustrate the nodal error norms (equation 5.6) of the reference and displaced surface mesh at each step of the two-step procedure. In the first step, the RBF interpolation is carried out using $\sim 10^4$ nodes, Figure 5.1, as a consequence of the data reduction algorithm. The first step significantly reduces the nodal error norms, but the resulting surface mesh does not perfectly fit the new geometry. The second step uses all the ~6×10^4 surface mesh nodes to perform the RBF interpolation that corrects all boundary nodal displacements. In the second step, the sparsity of the linear system that needs to be solved to perform the RBF interpolation is ~99.8%.

5.5.2 Scalability Studies on the Mesh Size

Figure 5.8 illustrates the results of the investigation of the time and memory requirements for performing the displacement of increasingly larger RRD turbine stator and double elbow duct CFD meshes. The turbine stator mesh is displaced from the reference to the second deformed shape of Figure 5.12, whereas the shape modification of the double elbow duct is that of Figure 5.10. The time required to morph the meshes is similar in both cases for similar mesh sizes. In both cases, the predictor interpolation and the corrector training are the most expensive phases. The former scales linearly with the mesh size, thanks to the FMM – the latter scales super-linearly due to the increased matrix size. The predictor training time is almost constant since the imposed surface mesh displacements are similar for all mesh sizes. The corrector interpolation phase also scales super-linearly with the

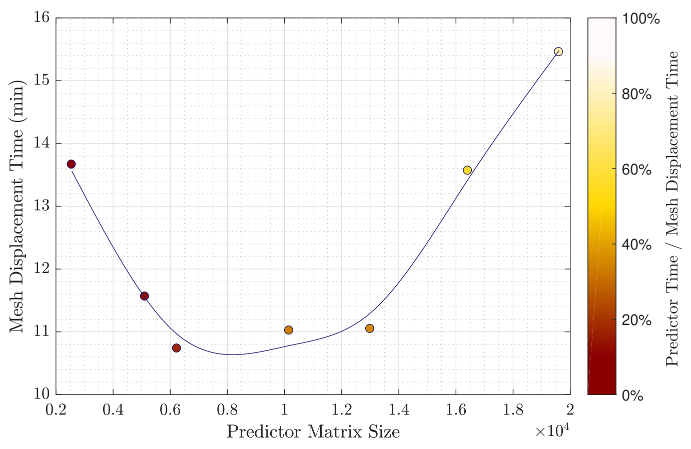


Figure 5.9: DrivAer car model. Wall-clock time for the mesh displacement, as illustrated in Figure 4, by varying the coarsening in the predictor. Wall-clock time measurements are represented as circles, whereas the curve represents the wall-clock time trend. The time spent in the predictor step against the overall mesh displacement time is also marked with colors from red to white.

mesh size due to the increased number of RBF kernel evaluations, but contributes just a little to the total computational cost, thanks to the integer lattice-based strategy.

5.5.3 Parametric Study on the Predictor Matrix Size

The DrivAer car geometry is a test case developed by the Technical University of Munich [119] that is herein used in the fastback configuration with mirrors, wheels and a smooth under-body. The mesh with ~4.20×10^6 nodes, ~3.58×10^5 of which are surface mesh nodes, consists of various types of elements, with up to 22-face polyhedra. The reference mesh is displaced to the deformed one, as a result of shape optimization for minimizing drag [120] (not included in this thesis). Mesh quality metrics are listed in Table 5.3. The results exhibit small differences in the maximum aspect ratio, which increases by 10% with almost the same maximum skewness. Moreover, no degenerated elements are observed.

Figure 5.9 reports an investigation of the time required to displace a mesh as a function of the predictor training matrix size. Additionally, the same figure shows the time spent in the predictor against the overall time required to displace the mesh. The results display that, in this case, an optimal size for the coarsening exists. More importantly, the trend of the overall time required, around the optimal predictor training matrix size, is nearly flat, and non-optimal predictor sizes yield an increase in computational cost below 50%, for this case. In fact, the lowest required time found is ~11 min, while the highest is ~15 min. The predictor training matrix size is controlled by the user-defined data reduction parameters.

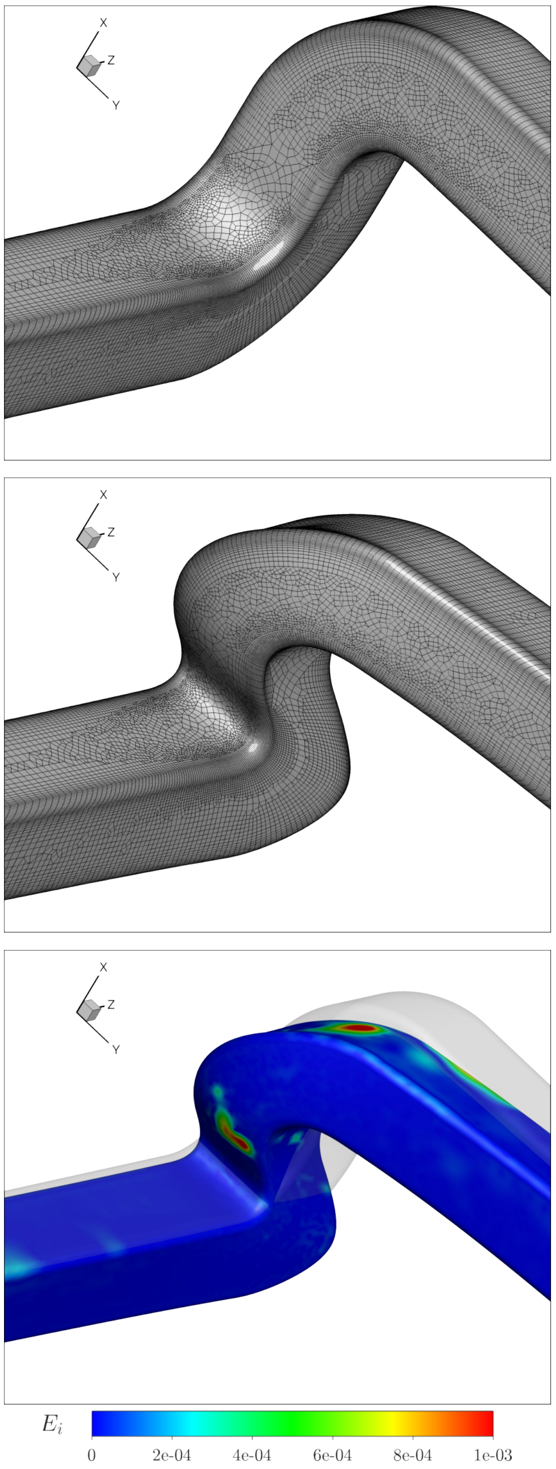
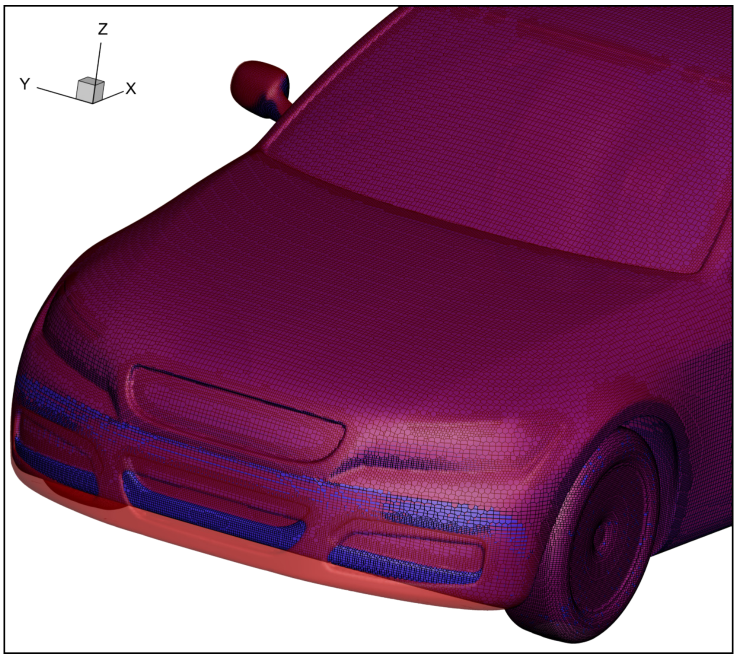


Figure 5.10: Double elbow duct. Detail of the reference (top) and displaced (middle) CFD unstructured meshes. On the bottom, the two shapes are compared and the low-to-high nodal error norms of the CFD mesh, after the 1^{st} step of the two-step strategy, are illustrated on the displaced shape with colors ranging from blue (low absolute values) to red (high absolute values).



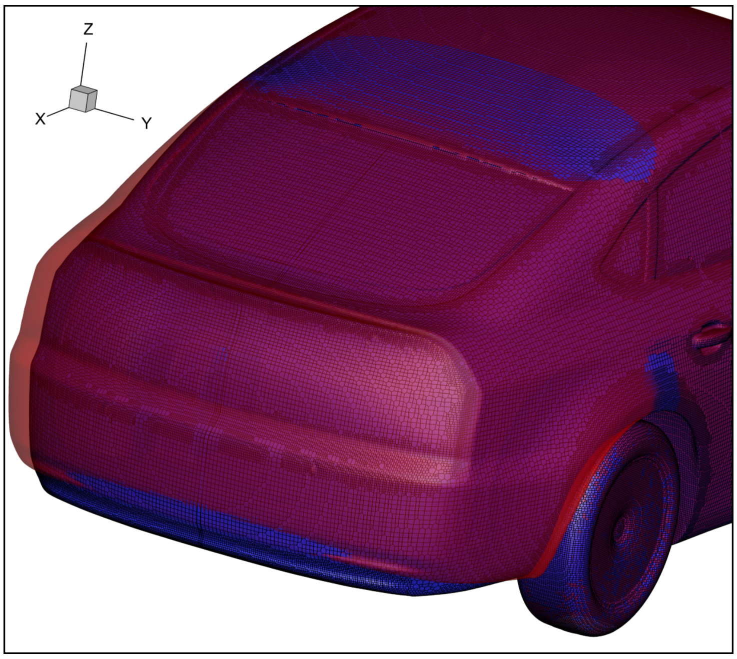


Figure 5.11: DrivAer car model. Reference (blue) and optimal (red) shape as a result of drag optimization. The most important shape modifications are located at the front (top figure) and rear (bottom figure) parts of the car model.

Table 5.3: DrivAer car model. Quality metrics of the unstructured polyhedral mesh around the reference and deformed designs illustrated in Figure 5.11.

	Reference	Deformed
Min. Jacobian	>0	>0
Max aspect ratio	36.0	39.4
Max skewness	3.5	3.5

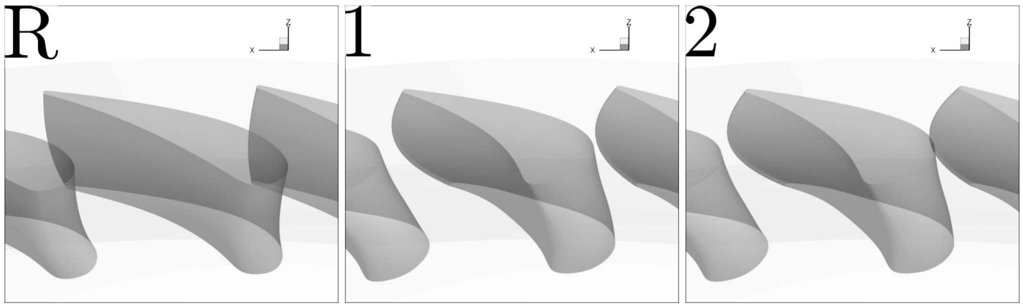


Figure 5.12: RRD turbine stator blade. From left to right: reference, 1^{st} and 2^{nd} deformed shapes selected from the front of non-dominated solutions of the two-objective optimization of the turbine stator.

5.5.4 RRD Turbine Stator CFD Shape Optimization

In this section, the proposed mesh displacement strategy is incorporated into an EA-based shape optimization of the turbine stator, using the software EASY (Appendix D). A (10; 20)-EA is used to optimize the shape. The computational budget is limited to 250 solver calls. The transonic flows, for the various designs, are resolved with the PUMA solver (Appendix E), employing the Spalart-Allmaras turbulence model. The simulation takes $\approx$ 70 minutes on 2 NVIDIA Tesla K40 GPUs.

The blade has a maximum chord length of 60 mm. Hub and shroud average radii are 221 mm and 268 mm, respectively. Inlet and outlet conditions are provided in the form of radial profiles, corresponding to 565 K average inlet total temperature, 190 kPa average inlet total pressure, 0.1° and 0.4° average inlet peripheral and radial flow angles respectively and 114 kPa average outlet static pressure. The blade shape is parameterized with the B-Rep-Morpher tool, and there are overall 8 design variables. The surface mesh conforming to each new boundary is obtained by inverting and displacing nodes in the NURBS parametric space, taking special care of trimmed surfaces, as described in Section 4.1; in detail, Figure 4.2 and 4.3 present the current case and a part of the computational mesh. The volume mesh is deformed using the two-step method presented in Section 5.3 with additional treatments for the matching periodic boundaries. The mesh is block-structured with viscous layers and

Table 5.4: RRD turbine stator blade. Quality metrics and objective function values for the block-structured volume mesh of the turbine stator reported for the reference and two non-dominated designs illustrated in Figure 5.12. The sign of the Jacobian is used to check the validity of the mesh. Larger orthogonality metric values and lower normal skewness values are favorable regarding CFD solution accuracy. $y^+ < 1$ of the first nodes off the wall is desirable to guarantee that the mesh near the wall is adequate for CFD simulations with a low-Re turbulence model. In the second deformed mesh, the max. y^+ is higher than 1 only for a small number of nodes, so the results are still considered reliable. Surface mesh quality metrics are also reported since they represent bounds for the quality metrics of the volume mesh.

	Reference	1^{st} Def.	2^{nd} Def.
Capacity $[ms\sqrt{K}]$	5.23×10^{-4}	7.45×10^{-4}	7.43×10^{-4}
Total Pressure Losses	6.82×10^{-2}	4.92×10^{-2}	9.67×10^{-2}
Min. Jacobian	>0	>0	>0
Min. min. orthogonality	0.17	0.14	0.08
Avg. min. orthogonality	0.74	0.68	0.62
Max. max. normal skewness	0.84	0.86	0.93
Avg. max. normal skewness	0.27	0.32	0.39
Max. y^+	0.67	0.95	1.19
Min. surface orthogonality	0.17	0.12	0.07
Max. max. surface normal skewness	0.84	0.89	0.93

~2.20×10^6 nodes, ~1.20×10^5 out of which are surface nodes.

The optimization aimed at minimizing the mass-averaged total pressure losses and maximizing row capacity. The total pressure losses between the inlet S_I and outlet S_O are expressed in the form of the non-dimensional quantity

$$\Delta P_t = \frac{\overline{p_t}|_{S_I} - \overline{p_t}|_{S_O}}{\overline{(p_t - p)}|_{S_I}}, \tag{5.11}$$

where p is the pressure, p_t the total pressure, and operators $\overline{\cdot}|_{S_I}$ and $\overline{\cdot}|_{S_O}$ represent mass-flow-averaging at inlet and outlet, respectively. Capacity is defined as

$$C = \dot{m} \left. \frac{\sqrt{\overline{T_T}}}{\overline{p_T}} \right|_{S_O}, \tag{5.12}$$

where $\dot{m}$ is the mass flow and T_t the total temperature. A geometric constraint to keep the axial chord of the stator blade constant is also imposed. The row is composed of 34 blades.

Figure 5.13 illustrates the front of non-dominated solutions and Figure 5.12 illustrates the shapes for the reference and two improved designs. The mass flow, total pressure and total temperature distributions are illustrated in Figure 5.14, 5.15 and 5.16, respectively.

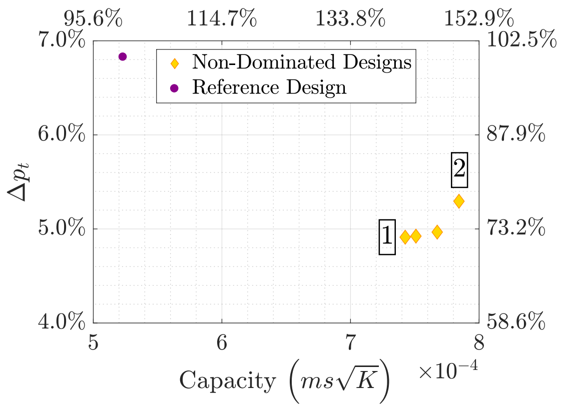


Figure 5.13: RRD turbine stator blade. EA-based optimization with the B-Rep-Morpher parameterization method. The shapes of the two non-dominated solutions, marked with integers, are compared to the reference design in Figure 5.12.

To evaluate the quality of the obtained deformed meshes, adapted to the improved shapes, various quality metrics for structured meshes were computed, as presented in Table 5.4, along with the results of the optimization. The results indicated the absence of degenerated elements (minimum Jacobian > 0) even for large deformations. A negative impact on the worse quality metrics (min. min. orthogonality and max. max. normal skewness) resulted, which is attributed to just a few elements since averaged quality metrics (avg. min. orthogonality and avg. max. normal skewness) are preserved. Moreover, it should be taken into account that, when a volume mesh is deformed, the quality of surface deformations represents a bound for the volume mesh quality. For this reason, in Table 5.4, the quality metrics for the surface mesh are also tabulated, exhibiting the same trend as the volume mesh quality metrics.

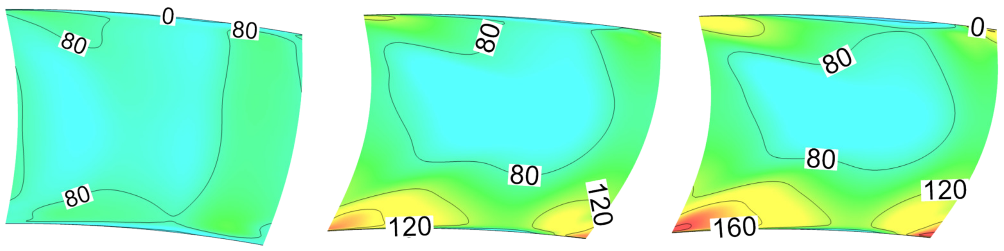


Figure 5.14: RRD turbine stator blade. ρV_a (kg/m^{2}s) iso-areas at the stator outlet for the reference (left), designs marked with 1 (middle) and designs marked with 2 (right) in the front on non-dominated solutions of Figure 5.13. Higher values imply higher capacity.

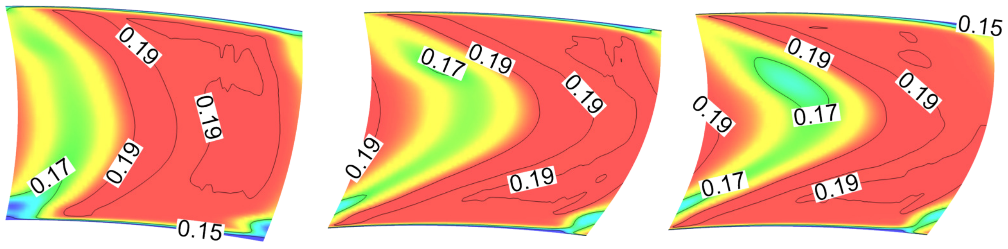


Figure 5.15: RRD turbine stator blade. p_t (MPa) iso-areas at the stator outlet for the reference (left), designs marked with 1 (middle) and designs marked with 2 (right) in the front on non-dominated solutions of Figure 5.13. Higher values imply lower total pressure losses and lower capacity.

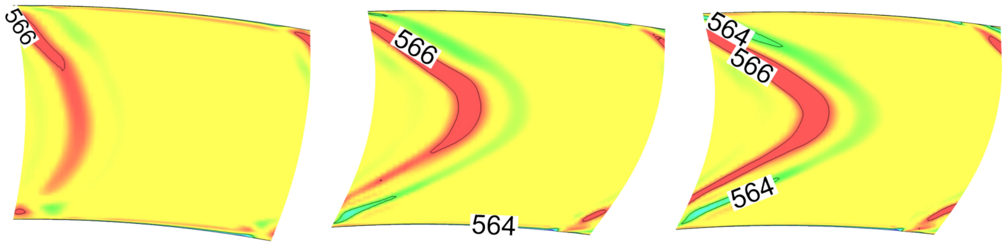


Figure 5.16: RRD turbine stator blade. T_t (K) iso-areas at the stator outlet for the reference (left), designs marked with 1 (middle) and designs marked with 2 (right) in the front on non-dominated solutions of Figure 5.13. Higher values imply higher capacity.

Chapter 6

The Fast Multipole Method

In Chapter 5, the FMM [116] has been used to speed-up the interpolation phase of the predictor step of the mesh displacement tool. In this chapter, the FMM is described in more detail and the performance of its implementation are analyzed.

6.1 Introduction

As previously noted, the FMM is a technique to compute sums such as:

$$d(\boldsymbol{x}_n) = \sum_{k=1}^{K} w_k \phi(\boldsymbol{x}_n, \boldsymbol{y}_k), \quad n = 1, \ldots, N \tag{6.1}$$

with computational complexity $\mathcal{O}(K + N)$ instead of $\mathcal{O}(KN)$, at the cost of committing a predictable error ϵ[1]. The computation of these kinds of sums is closely related to a number of problems involving the interaction of K sources to N targets, such as gravitational systems in astrophysics and electrostatic potential evaluation in physics. $d(\boldsymbol{x}_n)$ represents a field value computed at a target point $\boldsymbol{x}_n$; the value is computed considering the influence of all the sources located at $\boldsymbol{y}_k$. $\phi(\boldsymbol{x}_n, \boldsymbol{y}_k)$ is the kernel function governing the interaction of the k^{th} source with the n^{th} target. The most straightforward approach is to evaluate all pairwise interactions among sources and targets. While this direct approach is accurate within machine precision, the computational complexity of the solution is $\mathcal{O}(KN)$. The direct approach is appropriate only for small numbers of sources and targets or in cases in which very high accuracy is needed. For larger numbers of sources and targets, more efficient algorithms have been devised, such as treecodes [121] and the FMM. The principal

[1]In equation 5.1 the function $\boldsymbol{d}(\boldsymbol{x})$ is $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ however, for easiness of the exposition of the FMM, herein the function $d(\boldsymbol{x}_n) \ : \ \mathbb{R}^3 \rightarrow \mathbb{R}$ is considered instead.

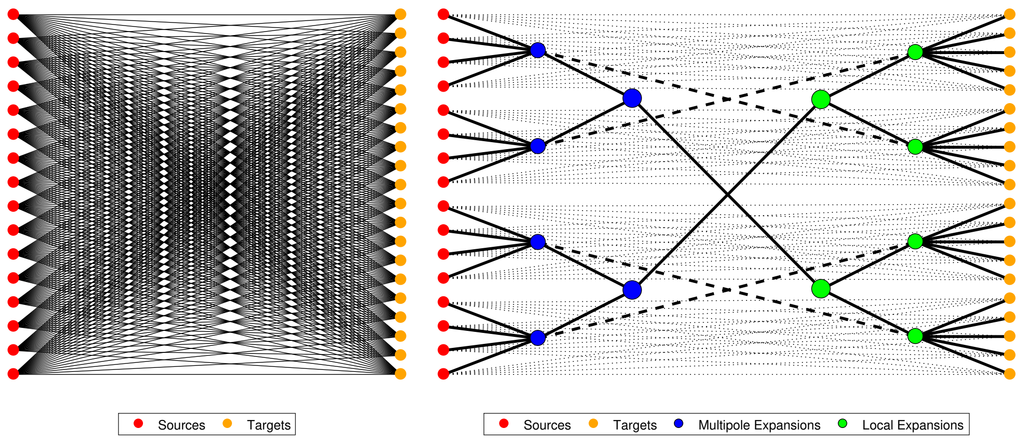

● Sources ● Targets
● Sources ● Targets ● Multipole Expansions ● Local Expansions

Figure 6.1: Schematic of the interactions of K sources and N targets. Left: Interactions using the direct method with $O(KN)$ complexity. Right: Schematic of the interactions for the FMM with $O(K + N)$ complexity. The approximation introduced by the FMM reduces the number of operations at the price of damaging accuracy.

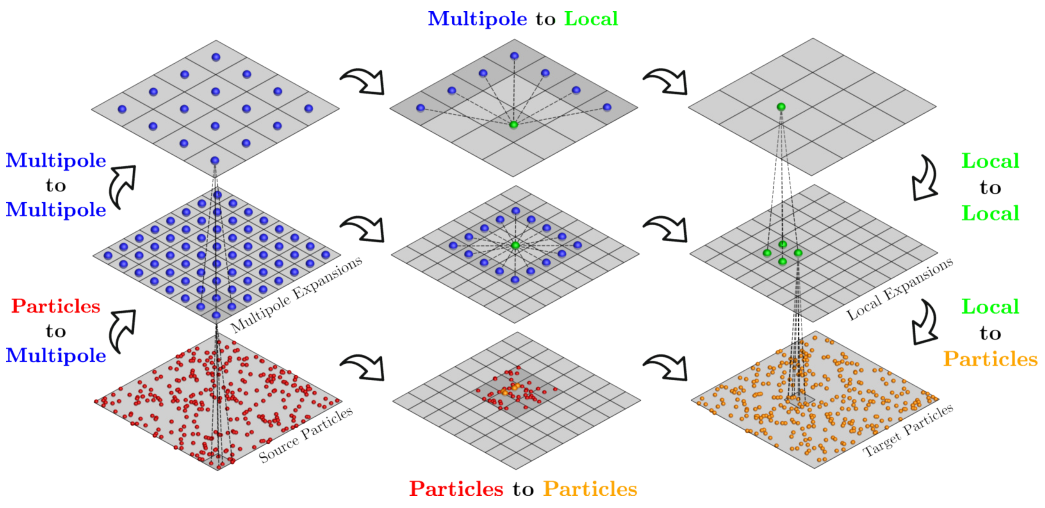


Figure 6.2: Schematic of the interactions of sources and targets for the FMM from the geometric point of view of the hierarchical decomposition. The various operators of the FMM are illustrated.

idea of these fast algorithms is to separately handle near-field ($\left\|\boldsymbol{x}_n - \boldsymbol{y}_k\right\| \to 0$) and far-field ($\left\|\boldsymbol{x}_n - \boldsymbol{y}_k\right\| \to \infty$) contributions to $d(\boldsymbol{x}_n)$. The near-field is computed directly, and the far-field can be approximated and hence computed with lower complexity (and accuracy).

Figure 6.1 demonstrates how the source "particles" interact with the target "particles" with the direct method and the FMM. Figure 6.2 illustrates by schematic how the FMM approximation is achieved by clustering far-away sources and targets into successively larger groups and aggregating values in multipole and local expansions. A multipole expansion represents the "potential" generated by sources contained inside a bin; a local expansion of a bin represents the potential of all well-separated sources. In the FMM, the sequence of computations is as follows. The algorithm requires first an upward pass of the octree. The source particles are aggregated to multipole expansion employing the Particles to Multipole (P2M) operator. Then, multipole expansions are aggregated into greater groups applying the Multipole to Multipole (M2M) operator. Then, during the downward pass, the multipole expansions are translated to local expansions between well-separated bins using the Multipole to Local (M2L) operator. Local expansions of larger parent[2] bins are accumulated to children bins with the Local to Local (L2L) operator. Values $d(\boldsymbol{x}_n)$ at the target particles are computed by considering the contribution of neighbor sources employing the Particles to Particles (P2P) operator and the contribution of all other sources with the local expansion accumulated in each leaf bin with the Local to Particles (L2P) operator. Both Figures 6.1 and 6.2 show that larger bins interact with others being significantly far-away, and smaller bins interact with closer ones.

Common to each FMM method are two fundamental ingredients:

- An octree data-structure used to spatially organize the RBF sources and targets as well as to differentiate near and far nodes at various levels, which is illustrated in Section 6.2.
- A low-rank approximation method to build the operators needed by the FMM, which is illustrated in Section 6.3.

The performance of the implemented FMM code is analyzed in Section 6.4.

6.2 The Octree Data-Structure

The FMM requires the definition of a spatial hierarchy: the domain comprising sources and targets is hierarchically partitioned into increasing levels of refinement and, then, the near- and far-fields interactions are identified at each level. A quadtree in 2D or an octree in 3D

[2]Bins that contain other bins.

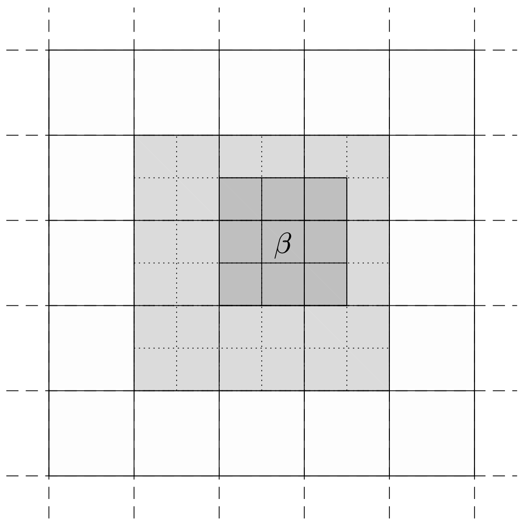


Figure 6.3: 2D quadtree. Neighbors of a bin β are illustrated in dark-gray (each bin is neighbor of itself) and colleagues in light-gray. The white bins are well-separated from β. In 3D, a bin has up to 27 colleagues, and its interaction list may comprise up to 189 bins ($6^3 - 3^3$).

is used for this purpose, and for each bin of the tree, an interaction list w.r.t. the other bins is defined.

Source and target distributions encountered in the mesh displacement application[3] are highly non-uniform, especially if meshes for viscous simulations are employed, due to the presence of viscous layers. Therefore, the octree used to build the interaction lists must adapt to such distributions, also taking the difference between sources and targets into account.

The data-structure is built as follows: the computational domain is defined to be the smallest cube containing all sources and targets. This cube is the octree bin at level $l = 0$. Then, a hierarchy of children bins, that refine the computational domain into smaller regions is built. Levels of refinement are added with a top-down algorithm dividing bins recursively in eight cubic bins of equal size. Before going into details of the octree subdivision criteria, the following definitions must be introduced:

- A bin α is defined to be a child of a bin β if α is obtained by a single subdivision of β. The bin β is defined to be the parent bin of α. A bin is set to be a leaf if it is childless.
- Two bins are defined to be colleagues if they are at the same octree level l and share at least a boundary point; a bin is a colleague of itself.
- Each bin β is associated with an interaction list comprising the children of the colleagues of its parents, which are not colleagues of β.

[3]For instance, the sources are the surface mesh nodes and the targets the internal volume nodes.

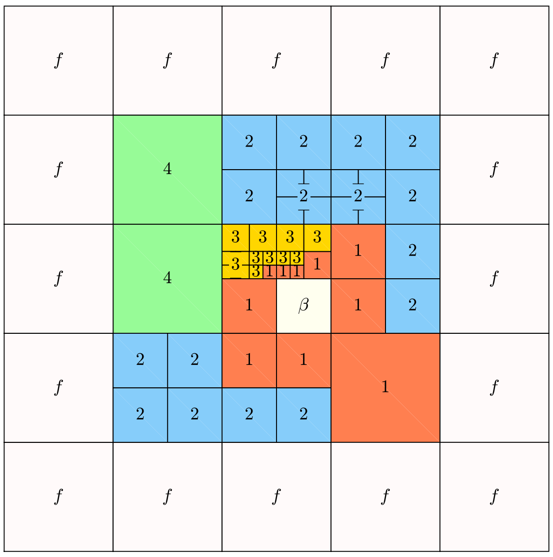


Figure 6.4: Adaptive list-1,-2,-3 and -4 for the bin β. Bins marked with f (far) are not considered in the adaptive lists of β.

Figure 6.3 illustrates the colleagues and interaction list for a bin in a quadtree. A bin is divided following the subsequent rules:

- A bin that does not contain sources and targets is ignored.
- A bin that contains fewer sources or targets than a user-defined threshold is set as a leaf; otherwise, it is considered a parent bin, and is divided into 8 children bins.

The following adaptive lists, namely sets of bins, are defined to be used by the FMM operators in Section 6.3:

List-1 of leaf bin β, $\mathscr{L}_1^\beta$, is the set comprising all leaf bins at any octree refinement level that share at least a boundary point with β; β is included in this set. If β is a parent box, its list-1 is empty.

List-2 of a bin β, $\mathscr{L}_2^\beta$, is its interaction list, namely the children of the colleagues of β's parents which are not colleagues of β.

List-3 of a leaf bin β, $\mathscr{L}_3^\beta$, is the set comprising all descendants of β's colleagues that do not share any boundary point with β but whose parent boxes share at least a boundary point with β. If β is a parent box, its list-3 is empty.

List-4 of a bin β, $\mathscr{L}_4^\beta$, is the set comprising the bins on which β is in their list-3.

Figure 6.4 illustrates such lists for a bin in a quadtree. Each bin containing sources is associated with a multipole expansion; each bin containing targets is associated with a local expansion. Figure 6.5 illustrates the leaf bins of the octree constructed on the CFD mesh of a compressor stator.

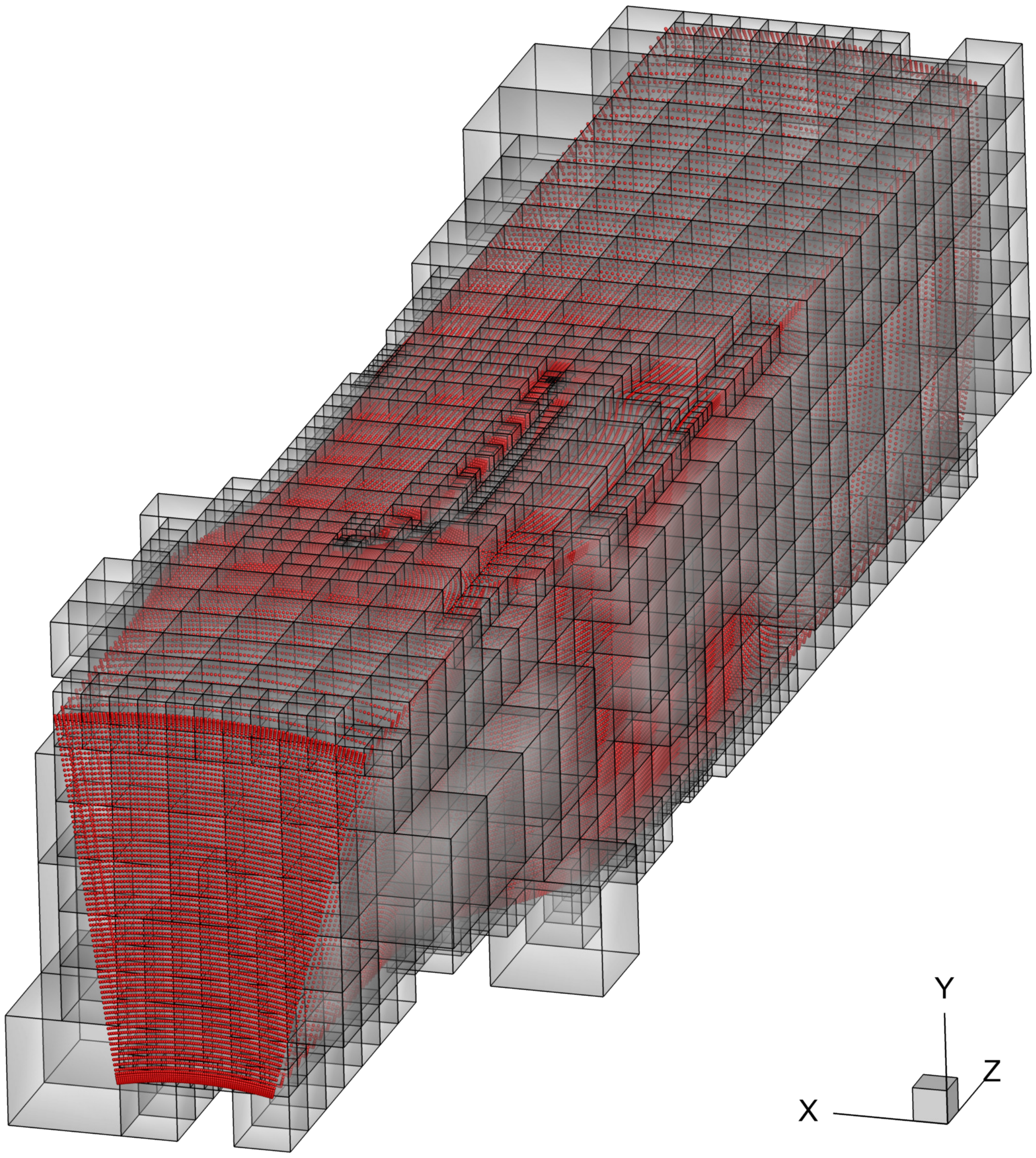


Figure 6.5: TUB stator (Chapter 7). Leaf bins of an octree used by the FMM. The octree is constructed on the CFD mesh. The surface mesh nodes are depicted in red.

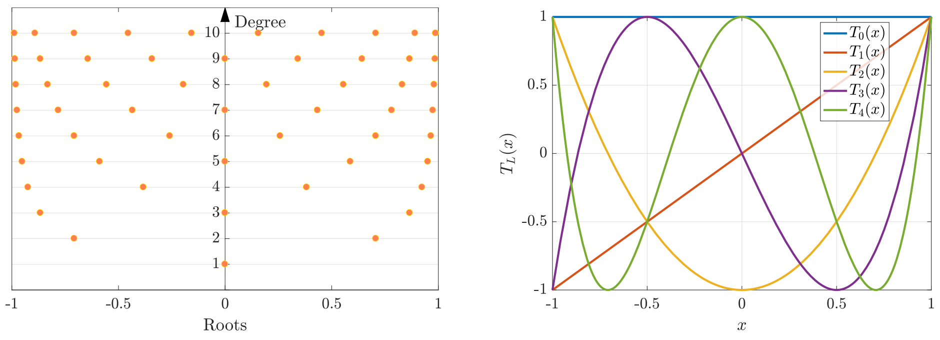


Figure 6.6: Chebyshev Polynomials. Left: Roots of the first 11 Chebyshev polynomials. Right: First five Chebyshev polynomials.

6.3 The Black-Box Fast Multipole Method

[116] first developed an FMM technique for the kernel $\phi(\boldsymbol{x}_i, \boldsymbol{y}_k) = 1/r$ with $r = \|\boldsymbol{x}_i - \boldsymbol{y}_k\|$; the approximation of ϕ for r sufficiently large (well-separated sources and targets) was obtained by using analytical expansions of ϕ. Therefore, the extension of this method to other kernels requires the development of analytical expansions, which was done in [122–125]. FMM techniques that rely only on discrete values of ϕ have also been investigated. For instance, in [126], interpolation based on the Legendre polynomials and the Singular Values Decomposition (SVD) method are used to build a kernel-independent (black-box) FMM technique with a controllable error. The FMM technique used by the mesh displacement tool presented in Chapter 5 is based on the black-box Fast Multipole Method (bbFMM) [106]. The advantage of the bbFMM is that the only input needed is the ability to evaluate ϕ at various points, with the only requirement that ϕ must be asymptotically smooth [127]. The bbFMM is based on the approximation of the kernel function through polynomial interpolation on Chebyshev nodes. Before going into the details of the bbFMM in Section 6.3.2 and 6.3.3, Section 6.3.1 recalls some properties of Chebyshev polynomials.

6.3.1 Interpolation Based on Chebyshev Polynomials

The Chebyshev polynomials are a sequence of orthogonal polynomials. The Chebyshev polynomials T_L are polynomials of degree L satisfying the relation

$$T_L(x) = \cos(L \arccos(x)) \tag{6.2}$$

on the interval $x \in [-1, 1]$. The image of $T_L(x)$ is also in the interval $[-1, 1]$. Generic intervals $[a, b]$, other than $[-1, 1]$, are considered by scaling the variable x, that is $x \rightarrow$

$\frac{1}{2}((b-a)x + a + b)$. From equation 6.2 the recursive definition generating the polynomial expressions is built:

$$\begin{aligned} T_0(x) &= \cos(0) = 1 \\ T_1(x) &= \cos(\arccos(x)) = x \\ T_L(x) &= 2xT_{L-1}(x) - T_{L-2}(x)\,, \quad L = 2, 3, \ldots \end{aligned} \tag{6.3}$$

T_L has L roots located at the nodes

$$\hat{x}_l = \cos\left(\frac{2l-1}{2L}\pi\right)\,, \quad l = 1, \ldots, L \tag{6.4}$$

which are called Chebyshev nodes. These cluster near the end of the domains, as illustrated in Figure 6.6. Polynomial interpolation on the Chebyshev nodes improves the stability of the interpolation scheme, compared to polynomial interpolation using equally-spaced nodes. The latter suffers from Runge's phenomenon; namely, the interpolation error does not converge uniformly as the number of interpolation nodes L increase [128, §13]. Polynomial interpolation on the Chebyshev nodes ensures uniform convergence to the interpolated function. Another advantage of the interpolation on the Chebyshev nodes is that it minimizes the maximum interpolation error and spreads it across the whole domain $[-1; 1]$ [128, §8]. Using the Chebyshev nodes as interpolation nodes, the approximating polynomial to a function $g(x)$ can be expressed as a Chebyshev series [128, §4]:

$$p_{L-1}(x) = \sum_{\lambda=0}^{L-1} c_\lambda T_\lambda(x) \tag{6.5}$$

with

$$c_\lambda = \begin{cases} \frac{1}{n}\sum_{l=1}^{L} g(\hat{x}_l) & \text{for } \lambda = 0 \\ \frac{2}{n}\sum_{l=1}^{L} g(\hat{x}_l)T_\lambda(\hat{x}_l) & \text{for } \lambda > 0 \end{cases} \tag{6.6}$$

For convenience, equation 6.5 can be written as

$$p_{L-1}(x) = \sum_{l=1}^{L} g(\hat{x}_l) S_L(\hat{x}_l, x) \tag{6.7}$$

with

$$S_L(\mathsf{x}, \mathsf{y}) = \frac{1}{L} + \frac{2}{L}\sum_{\lambda=1}^{L-1} T_\lambda(\mathsf{x})T_\lambda(\mathsf{y}) \tag{6.8}$$

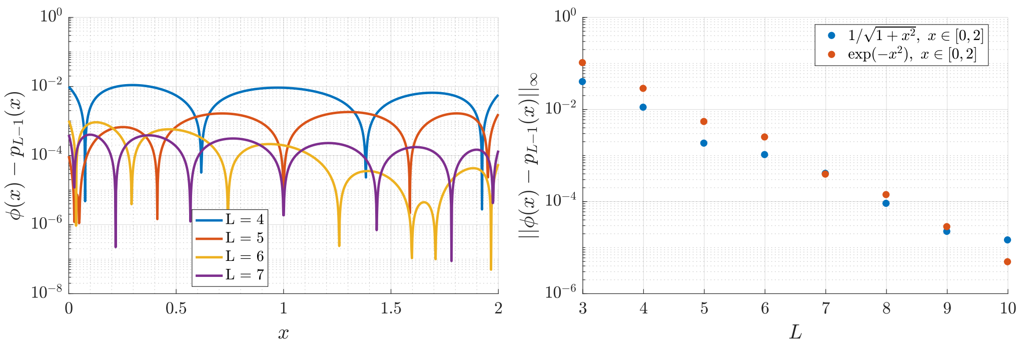


Figure 6.7: Chebyshev Polynomials. Left: Interpolation error for $\phi = 1/\sqrt{1+x^2}$, in the domain $x \in [0, 2]$ for a various number of Chebyshev nodes L. Right: Maximum interpolation error for two kernels in the domain $x \in [0, 2]$ in function of the number of Chebyshev nodes.

The convergence study for the Chebyshev series of equation 6.7 to the continuous and differentiable function $g(x)$ is given in [129, §5.7].

The interpolation can be extended to higher dimensions using the tensor product of the functions S_L, one for each dimension. For instance, in 3D, the polynomial approximation $p_{L-1}(\boldsymbol{x}) : \mathbb{R}^3 \to \mathbb{R}$ of the function $g(\boldsymbol{x}) : \mathbb{R}^3 \to \mathbb{R}$ with $\boldsymbol{x} = (x_1, x_2, x_3)^\mathsf{T}$ is expressed as

$$p_{L-1}(\boldsymbol{x}) = \sum_{l=1}^{L^3} g(\hat{\boldsymbol{x}}_l) R_L(\hat{\boldsymbol{x}}_l, \boldsymbol{x}) \tag{6.9}$$

with $\hat{\boldsymbol{x}}_l = (\hat{x}_{l1}, \hat{x}_{l2}, \hat{x}_{l3})^\mathsf{T}$, $l_i \in \{1, \ldots, L\}$, being the vectors of Chebyshev roots and

$$R_L(\hat{\boldsymbol{x}}_l, \boldsymbol{x}) = S_L(\hat{x}_{l1}, x_1) S_L(\hat{x}_{l2}, x_2) S_L(\hat{x}_{l3}, x_3)\ . \tag{6.10}$$

The vectors of Chebyshev roots $\hat{\boldsymbol{x}}_l$, $l \in 1, \ldots, L^3$ are properly defined to span all L^3 combinations of L (1D) roots.

L determines the accuracy of the polynomial interpolation; higher accuracy requires higher L values. Figure 6.7 illustrates the interpolation error as a function of the number of Chebyshev nodes used.

6.3.2 Interpolation-Based Low-Rank Approximation

The construction of an interpolation-based low-rank approximation works as follows. By introducing the function μ_l and $\nu_l : \mathbb{R}^3 \to \mathbb{R}$, a low-rank separable approximation of the kernel ϕ can be constructed as:

$$\phi(\boldsymbol{x}_n, \boldsymbol{y}_k) \approx \sum_{l}^{L} \mu_l(\boldsymbol{x}_n) \nu_l(\boldsymbol{y}_k) \,. \tag{6.11}$$

Substituting equation 6.11 in equation 6.1 the result is:

$$d(\boldsymbol{x}_n) = \sum_{l=1}^{L} \left(\mu_l(\boldsymbol{x}_n) \sum_{k=1}^{K} w_k \nu_l(\boldsymbol{y}_k) \right), \quad n = 1, \ldots, N \,. \tag{6.12}$$

In equation 6.12, $d(\boldsymbol{x}_n)$, $n = 1, \ldots, N$ may be computed in two steps:

- Transform the sources using the interpolation basis function:

$$\mathcal{W}_l = \sum_{k=1}^{K} w_k \nu_l(\boldsymbol{y}_k), \quad l = 1, \ldots, L \,. \tag{6.13}$$

- Compute $d(\boldsymbol{x})$ at the target points:

$$d(\boldsymbol{x}_n) = \sum_{l=1}^{L} \mu_l(\boldsymbol{x}_n) \mathcal{W}_l, \quad n = 1, \ldots, N \,. \tag{6.14}$$

These two steps together have computational complexity $O(L(K + N))$.

In the bbFMM, the low-rank approximation of ϕ is constructed with an interpolation scheme. The functions μ_l and ν_l are built as follows. A Λ-point interpolant of a function $g(x)$, $x \in [-1, 1]$ can be expressed as:

$$p_{L-1}(\boldsymbol{x}) = \sum_{l=1}^{\Lambda} g(\boldsymbol{x}_l) \ell_l(\boldsymbol{x}) \,, \tag{6.15}$$

where ℓ_l represent the interpolating functions, i.e. a polynomial, and $\hat{\boldsymbol{x}}_l$ the interpolation nodes, i.e. the Chebyshev nodes. Equation 6.15 is used to build a low-rank approximation of the kernel ϕ by first fixing the variable $\boldsymbol{y}$ which makes $\phi(\boldsymbol{x}, \boldsymbol{y})$ only function of $\boldsymbol{x}$:

$$\phi(\boldsymbol{x}, \boldsymbol{y}) \approx \sum_{l=1}^{\Lambda} \phi(\hat{\boldsymbol{x}}_l, \boldsymbol{y}) \ell_l(\boldsymbol{x}) \,. \tag{6.16}$$

Then, $\phi(\hat{\boldsymbol{x}}_l, \boldsymbol{y})$ is only a function of $\boldsymbol{y}$ and equation 6.15 is used again to give:

$$\phi(\boldsymbol{x}, \boldsymbol{y}) \approx \sum_{l=1}^{\Lambda} \sum_{\lambda=1}^{\Lambda} \phi(\hat{\boldsymbol{x}}_l, \hat{\boldsymbol{y}}_\lambda) \ell_l(\boldsymbol{x}) \ell_\lambda(\boldsymbol{x}) \,, \tag{6.17}$$

which is a low-rank approximation of ϕ in the form of equation 6.11 with:

$$\begin{aligned} \mu_l(\boldsymbol{x}) &= \ell_l(\boldsymbol{x}) \\ \nu_l(\boldsymbol{y}) &= \sum_{\lambda=1}^{\Lambda} \phi(\hat{\boldsymbol{x}}_l, \hat{\boldsymbol{y}}_\lambda)\ell_\lambda(\boldsymbol{x})\,. \end{aligned} \tag{6.18}$$

The interpolating functions $\ell_l(\boldsymbol{x})$ and $\ell_\lambda(\boldsymbol{x})$ in equation 6.17 can be identified with the functions R_L of equation 6.10, and the index Λ with L^3 (in 3D), as in equation 6.9; $(L-1)$ is the polynomial interpolation degree. Therefore, the low-rank approximation of the kernel $\phi : \mathbb{R}^3 \to \mathbb{R}$ employing Chebyshev polynomials is:

$$\phi(\boldsymbol{x}, \boldsymbol{y}) \approx \sum_{l=1}^{L^3} \sum_{\lambda=1}^{L^3} \phi(\hat{\boldsymbol{x}}_l, \hat{\boldsymbol{y}}_\lambda) R_L(\hat{\boldsymbol{x}}_l, \boldsymbol{x}) R_L(\hat{\boldsymbol{y}}_\lambda, \boldsymbol{y}) \tag{6.19}$$

Replacing this expression into equation 6.1 and switching the order of summations the result is:

$$\begin{aligned} d(\boldsymbol{x}_n) &= \sum_{k=1}^{K} w_k \phi(\boldsymbol{x}_n, \boldsymbol{y}_k) \\ &\approx \sum_{k=1}^{K} w_k \sum_{l=1}^{L^3} \sum_{\lambda=1}^{L^3} \phi(\hat{\boldsymbol{x}}_l, \hat{\boldsymbol{y}}_\lambda) R_L(\hat{\boldsymbol{x}}_l, \boldsymbol{x}_n) R_L(\hat{\boldsymbol{y}}_\lambda, \boldsymbol{y}_k) \\ &= \sum_{l=1}^{L^3} R_L(\hat{\boldsymbol{x}}_l, \boldsymbol{x}_n) \sum_{\lambda=1}^{L^3} \phi(\hat{\boldsymbol{x}}_l, \hat{\boldsymbol{y}}_\lambda) \sum_{k=1}^{K} w_k R_L(\hat{\boldsymbol{y}}_\lambda, \boldsymbol{y}_k), \quad n = 1, \ldots, N \end{aligned} \tag{6.20}$$

Equation 6.20 is the basis to build the bbFMM. In fact, a fast summation method can be constructed with the following three steps:

- Compute the weights at the Chebyshev nodes $\hat{\boldsymbol{y}}_\lambda$ (multipole expansion):

$$\mathcal{W}_\lambda = \sum_{k=1}^{K} w_k R_L(\hat{\boldsymbol{y}}_\lambda, \boldsymbol{y}_k) \ , \quad \lambda = 1, \ldots, L^3 \tag{6.21}$$

- Compute $d(\boldsymbol{x})$ at the Chebyshev nodes $\hat{\boldsymbol{x}}_l$ (local expansion):

$$\mathcal{D}_l = \sum_{\lambda=1}^{L^3} \mathcal{W}_\lambda \phi(\hat{\boldsymbol{x}}_l, \hat{\boldsymbol{y}}_\lambda) \ , \quad l = 1, \ldots, L^3 \tag{6.22}$$

- Compute $d(\boldsymbol{x})$ at the targets $\boldsymbol{x}_n$:

$$d(\boldsymbol{x}_n) = \sum_{l=1}^{L^3} \mathcal{D}_l R_L(\hat{\boldsymbol{x}}_l, \boldsymbol{y}_n) \quad , \quad n = 1, \ldots, N \tag{6.23}$$

These three steps have overall computational complexity $O(L^3(K+N) + L^6)$ that is approximately $O(L^3K)$ for $K > N \gg L$

6.3.3 The FMM Based on Chebyshev Interpolation

In Section 6.3.2, a method to perform summations such as equation 6.1, trading computational complexity with precision, has been presented for continuous kernels based on a low-rank approximation using interpolation based on Chebyshev polynomials. For the low-rank approximation of equation 6.19 to accurately represent ϕ, the targets and source domains need to be well-separated since equation 6.19 is a far-field approximation of the kernel ϕ. Far-field interactions are computed at various levels of refinements; on each level, only targets that are sufficiently far-away from the sources, but that cannot be taken into account on less refined levels, are involved. On the most refined level, interactions that are not sufficiently far-away are computed directly. Thus, the need for an octree data-structure with the (adaptive) interaction lists, illustrated in Section 6.2. The combination of such an octree and the fast summation method illustrated in Section 6.3.2 gives rise to an approximate multilevel fast summation method. This is summarized in the following steps:

1. The octree data-structure and the adaptive list-1,-2,-3 and -4 are built. The adaptive octree is built by recursion from the whole domain on level 0 until the most refined level $\mathcal{I}$.
2. Multipole expansions $\mathcal{W}_\lambda^\beta$, $\lambda = 1, \ldots, L^3$ are formed for each bin β that contains sources (upward pass).

   2.1 For each bin β in the set of leaf bins B_{Leaf} (that contains sources) multipole expansions are formed by computing the weights $\mathcal{W}_\lambda^\beta$ at the Chebyshev nodes $\hat{\boldsymbol{y}}_\lambda^\beta$ from the weights associated with the sources w_k (P2M):

$$\mathcal{W}_\lambda^\beta = \sum_{\boldsymbol{y}_k \in \beta} w_k R_L(\hat{\boldsymbol{y}}_\lambda^\beta, \boldsymbol{y}_k) \quad , \quad \lambda = 1, \ldots, L^3, \;\; \forall \beta \in B_{\mathrm{Leaf}} \,. \tag{6.24}$$

   2.2 From the refined to coarse octree level, for each bin β in the set of parent bins at level η, B_{Parent}^η (that contains sources), multipole expansions are formed by computing the weights $\mathcal{W}^{\beta}_{\lambda}$ at the Chebyshev nodes $\hat{\boldsymbol{y}}^{\beta}_{\lambda}$ merging the multipole expansion of the children of β, that are the bins α in the sets B^{β}_{child} (M2M):

$$\mathcal{W}^{\beta}_{\lambda} = \sum_{\alpha \in B^{\beta}_{\text{child}}} \sum_{l=1}^{L^3} \mathcal{W}^{\alpha}_{\lambda} R_L(\hat{\boldsymbol{y}}^{\beta}_{\lambda}, \hat{\boldsymbol{y}}^{\alpha}_{l}) \ , \qquad (6.25)$$
$$\lambda = 1, \ldots, L^3, \ \ \forall \beta \in B^{\eta}_{\text{Parent}} \ \ \eta = \mathcal{I} - 1, \ldots, 2 \, .$$

3. Local expansions $\mathcal{D}^{\beta}_{\lambda}$, $\lambda = 1, \ldots, L^3$ are formed for each bin β that contains targets (downward pass):
$$\mathcal{D}^{\beta}_{\lambda} = \mathcal{D}^{\beta}_{\lambda}|_{\text{M2L}} + \mathcal{D}^{\beta}_{\lambda}|_{\text{P2L}} + \mathcal{D}^{\beta}_{\lambda}|_{\text{L2L}} \, . \qquad (6.26)$$

3.1 For each bin β (that contains targets) contributions to the local expansions $\mathcal{D}^{\beta}_{\lambda}$ from multipole expansions of bins in the list-2 of β (M2L) are added:

$$\mathcal{D}^{\beta}_{\lambda}|_{\text{M2L}} = \sum_{\alpha \in \mathscr{L}^{\beta}_{2}} \sum_{l=1}^{L^3} \mathcal{W}^{\alpha}_{l} \phi(\hat{\boldsymbol{x}}^{\beta}_{\lambda}, \hat{\boldsymbol{y}}^{\alpha}_{l}) \ , \quad \lambda = 1, \ldots, L^3, \ \forall \beta \, . \qquad (6.27)$$

3.2 For each bin β (that contains targets) contributions are added to the local expansions $\mathcal{D}^{\beta}_{\lambda}$ from multipole expansions of bins α in the list-4 of β if the sources in α are more than L^3. Otherwise, sources are accounted for directly (M2L/Particles to Local (P2L)):

$$\mathcal{D}^{\beta}_{\lambda}|_{\text{P2L}} = \sum_{\alpha \in \mathscr{L}^{\beta}_{4}} \begin{cases} \displaystyle\sum_{l=1}^{L^3} \mathcal{W}^{\alpha}_{l} \phi(\hat{\boldsymbol{x}}^{\beta}_{\lambda}, \hat{\boldsymbol{y}}^{\alpha}_{l}) & \text{if } \#(y_k \in \alpha) > L^3 \\ \displaystyle\sum_{\boldsymbol{y}_k \in \alpha} w_k \phi(\hat{\boldsymbol{x}}^{\beta}_{\lambda}, \boldsymbol{y}_k) & \text{if } \#(y_k \in \alpha) \leq L^3 \end{cases} , \qquad (6.28)$$
$$\lambda = 1, \ldots, L^3, \ \forall \beta \, .$$

3.3 From coarse to refined octree level, for each bin β (that contains targets) contributions are added from the local expansions $\mathcal{D}^{\alpha}_{\lambda}$ of β parent bin, α (L2L):

$$\mathcal{D}^{\beta}_{\lambda}|_{\text{L2L}} = \sum_{l=1}^{L^3} \mathcal{D}^{\alpha}_{l} R_L(\hat{\boldsymbol{x}}^{\beta}_{\lambda}, \hat{\boldsymbol{x}}^{\alpha}_{l}) \ , \qquad (6.29)$$
$$\lambda = 1, \ldots, L^3, \ \ \forall \beta \in B^{\eta}, \ \ \eta = 3, \ldots, \mathcal{I}, \ \ (\alpha : \alpha \in B^{\beta}_{\text{child}}) \, .$$

4. Values d_n approximating $d(\boldsymbol{x}_n)$ are evaluated at target nodes $\boldsymbol{x}_n$, $n = 1, \ldots, N$ consid-

ering approximate far-fields contributions accumulated in the local expansions and exact direct contributions from the near-fields:

$$d_n = d_n|_{\mathrm{L2P}} + d_n|_{\mathrm{P2P}} + d_n|_{\mathrm{M2P}} \, . \tag{6.30}$$

4.1 Far-fields contributions are added to d_n in each leaf bin β (that contains targets), adding the contributions from local expansions accumulated in β (L2P):

$$d_n|_{\mathrm{L2P}} = \sum_{\lambda=1}^{L^3} \mathcal{D}^{\beta}_{\lambda} R_L(\hat{\boldsymbol{x}}^{\beta}_{\lambda}, \boldsymbol{x}_n) \ , \quad n = 1, \dots, N, \quad (\beta : \beta \ni \boldsymbol{x}_n) \, . \tag{6.31}$$

4.2 Direct contributions are added to d_n in each leaf bin β (that contains targets), considering sources in the bins of the list-1 of β (P2P):

$$d_n|_{\mathrm{P2P}} = \sum_{\alpha \in \mathscr{L}^{\beta}_{1}} \ \sum_{\boldsymbol{y}_k \in \alpha} w_k \phi(\boldsymbol{x}_n, \boldsymbol{y}_k) \ , \quad n = 1, \dots, N, \quad (\beta : \beta \ni \boldsymbol{x}_n) \, . \tag{6.32}$$

4.3 Contributions to targets $\boldsymbol{x}_n$ in each leaf bin β, coming from sources in bins α in the list-3 of β are added to d_n considering the multipole expansions of each bin α if the sources in α are more than L^3 otherwise sources in α are accounted for directly (P2P/Multipole to Particle (M2P)):

$$d_n|_{\mathrm{M2P}} = \sum_{\alpha \in \mathscr{L}^{\beta}_{3}} \begin{cases} \displaystyle\sum_{\lambda=1}^{L^3} \mathcal{W}^{\alpha}_{\lambda} \phi(\boldsymbol{x}_n, \hat{\boldsymbol{y}}^{\alpha}_{\lambda}) & \text{if } \#(y_k \in \alpha) > L^3 \\ \displaystyle\sum_{\boldsymbol{y}_k \in \alpha} w_k \phi(\boldsymbol{x}_n, \boldsymbol{y}_k) & \text{if } \#(y_k \in \alpha) \leq L^3 \end{cases} , \tag{6.33}$$

$$n = 1, \dots, N, \quad (\beta : \beta \ni \boldsymbol{x}_n) \, .$$

The P2M, M2M, M2L, L2L, L2P and P2P operators are schematically illustrated in Figure 6.2. The P2L and M2P operators are introduced in the adaptive bbFMM for the following reason. Consider an interaction between a source bin and a target bin in its $\mathscr{L}_4$ list; if the source particles in a source bin are fewer than the Chebyshev nodes of the multipole expansion, L^3 in 3D, contributions to the local expansion of the target bin are computed directly from the sources, thanks to the P2L operator. Similarly, when targets in a target bin that is in the $\mathscr{L}_3$ list of a source bin with fewer sources than the number of Chebyshev nodes of the multipole expansion, the contributions to the target nodes are added by evaluating the multipole expansion directly at the target particles positions, thanks to the M2P operator.

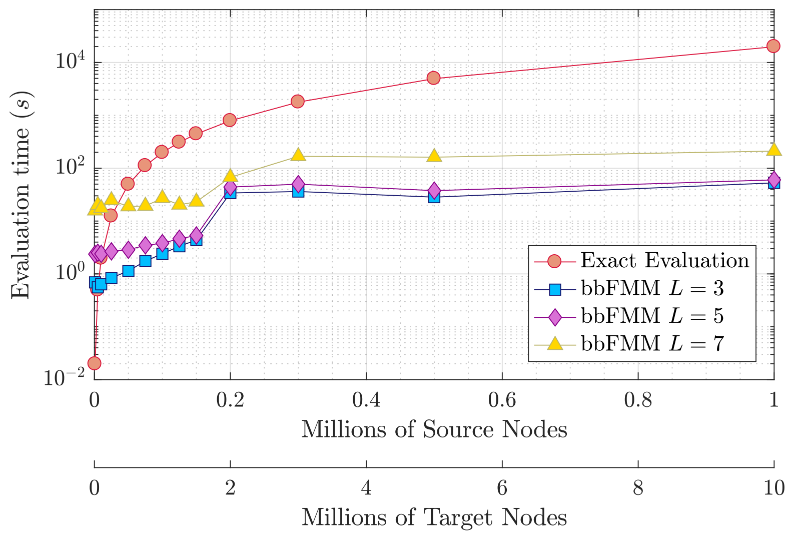


Figure 6.8: Wall-clock time for the evaluation of equation 5.1 by varying the number of sources and target nodes (retaining the ratio 1:10) for a uniform distribution randomly generated in a unit cube using the bbFMM method for the multiquadric RBF kernel. The FMM-based interpolations include the FMM set-up time. Three numbers of Chebyshev nodes L employed in the interpolation are illustrated.

6.4 Software Performance and Implementation

The benefits of code optimization, multi-threading, and parameter tuning are substantial to develop fast software for the FMM. In this section, the performance of the programmed software is analyzed. In detail, this section examines the software execution time scaling w.r.t. the number of particles, the effect on the performance of the maximum number of particles per octree leaf bin, and the time required by the various FMM operators. Software performance is measured on a computational node with two 6-core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz processors.

Figure 6.8 illustrates the time required to perform RBF interpolations for various numbers of particles, uniformly distributed in a unit cube, with and without the FMM. Computational time trends reflect the $\mathcal{O}(N + K)$ complexity of the FMM against the $\mathcal{O}(NK)$ complexity of the direct computation. The FMM-based RBF interpolation is more affordable for a large number of particles, even if the parallel implementation is more involved since it requires many synchronization points compared to the direct computation. The efficient parallelization of FMM codes is an active research area; for instance, see [130]. Some comments regarding the parallel implementation of the bbFMM used in this thesis are reported in Section 6.4.1.

Figures 6.9 and 6.11 illustrate the FMM execution time w.r.t. the number of Chebyshev nodes L and bin capacity β_C, for a uniform and non-uniform distribution, respectively. The uniform distribution of sources and targets is randomly generated, respectively, on the surface and into a unit cube whereas in the non-uniform distribution sources and targets correspond to the surface and volume nodes, respectively, of the CFD mesh on and within the compressor stator case (see Figure 6.5). Figures 6.10 and 6.12 illustrate the decomposition of the time required by the FMM into the time needed by the various operators and tree data structure for the two particle distributions and bin capacities. For both particle distributions, the execution time of the M2L operator quickly increases with L. For the non-uniform distribution, the time required by the P2P operator has a greater cost than in the uniform case, due to the increased $\mathscr{L}_3$ and $\mathscr{L}_4$ list sizes in favor of $\mathscr{L}_2$ lists. In fact, due to the decreased $\mathscr{L}_2$ list sizes, the M2L operator cost is much lower than the high-precision uniform distribution case. Moreover, the adaptive octree is optimized to be traversed efficiently when non-uniform distributions of particles are employed.

One of the reasons for the high cost of the M2L operator in the original algorithm is that, in 3D, there are up to 189 source bins per target bin; the source bins contain multipole expansions and the target bins contain local expansions. Therefore, the M2L operator requires up to 189 $L^3 \times L^3$ matrix-vector products, which result in the increased computational time for increasing L. In the programmed software, the matrices required by the M2L operator, named translation matrices, are precomputed. The kernel ϕ used for mesh displacement is translation-invariant, which allows precomputing the 316 translation matrices corresponding to the possible interactions between bins containing multipole expansions and a bin containing a local expansion for each octree level[4]. In Figures 6.10 and 6.12, the time needed for these computations is included in the set-up phase.

More efficient M2L operators for large L have been proposed. For instance, [106] proposed to use SVD compression to reduce the size of the matrix-vector products. If the kernel ϕ is homogeneous[5], it is possible to compute one SVD with size $(316 \cdot L^3) \times L^3$ containing all the translations of the M2L operator. In fact, in this case, the M2L operator is simply appropriately scaled to be used in each octree level. However, if the FMM-based summations are performed only once, the cost to compute the SVD is counterbalanced by the time saved in the M2L operations only for large K. Moreover, if the kernel ϕ is not homogeneous but just translation-invariant, which is the case for the kernels used for mesh displacement, the SVD must be computed for each octree level. In [127], the number

[4]In 3D, each target bin interacts with up to $6^3 - 3^3 = 189$ source bins. The union of the possible translation matrices for 8 bins comprises $7^3 - 3^3 = 316$ translation matrices.

[5]A function $\phi(r)$ is homogeneous of degree m if $\phi(\alpha r) = \alpha^m \phi(r)$ for any nonzero real α.

of translation matrices to be considered in each level is reduced to 16, instead of 316, by using symmetry planes. The remaining matrices are expressed as permutations thereof. This has a significant impact on the precomputation time when using SVD compression. As noted, the M2L cost in the non-uniform distribution has less impact, so that for mesh displacement, it is reasonable to use the original bbFMM algorithm without modifications.

Similarly to the M2L operator, other operators make use of matrices computed in the set-up phase and, then, reused for each bin appropriately. For instance, the M2M and L2L operators are a single matrix-vector product; on each level, 8 matrices correspond to the interactions of a parent bin with a child bin or vice versa. On the other hand, operators involving particles, such as P2M, L2P and P2P are implemented as matrix-free matrix-vector products since each bin generates a different matrix than the others. As expected, the P2P operator and tree construction costs are independent of L. This means that for low L, the P2P operator has a more significant relative impact on the total execution time, but it diminishes for high L, where the M2L operator becomes the most expensive one.

The execution time strongly depends on the maximum number of particles per bin β_C employed in the octree construction. Lower values, such as 20, lead to octrees with more levels, which in turn lead to higher computational cost due to the disequilibrium between the high cost of the M2L operator and the low cost of the P2P one. For higher β_C, such as 200, the execution time diminishes, and the M2L and P2P operators are more balanced. For even higher β_C, the execution time slowly increases.

6.4.1 Algorithm and Software Parallelization

The increasing degree of parallelism of modern hardware architectures and software scalability requirements involves the need to parallelize the implemented software.

As noted, the FMM-based RBF interpolation is more affordable for large meshes, even if the parallel implementation is more involved since it requires many synchronization points [130], compared to the direct computation. In fact, the direct computation is not affected by parallel slowdown effects because of the absence of synchronization overheads that are detrimental to achieve a good parallel efficiency[6]. In fact, in the direct computation, the summation for each target node, equation 6.1, can be carried out independently, which is without synchronization points.

Different parallelization schemes of the FMM algorithm have been proposed to exploit multicore architectures and increase parallel efficiency. [131] proposed a two-level strategy

[6]Parallel efficiency is defined as the speed-up divided by the number of units of execution, such as processors and cores.

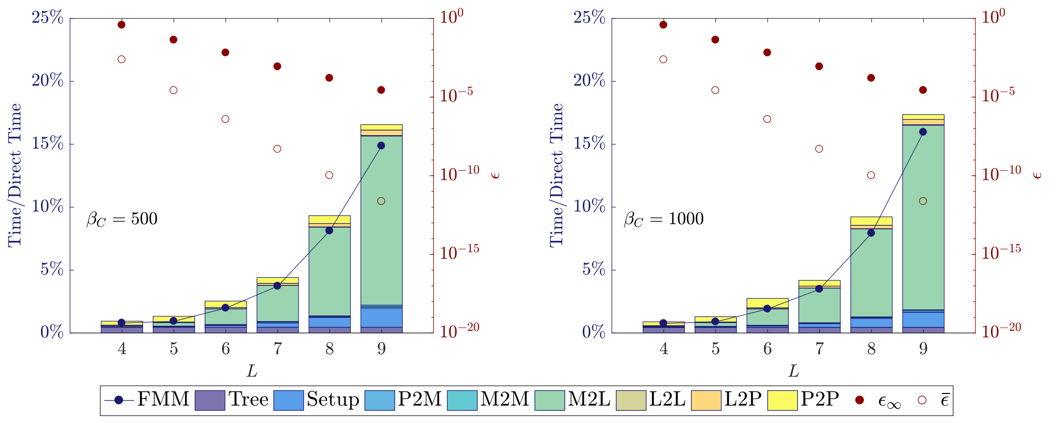


Figure 6.9: Uniform distribution. Time fractions and errors for the FMM-based summations for a uniform distribution of ~1.40 × 10^5 sources on the surface of a unit cube and ~2.31 × 10^6 targets in the volume of a unit cube for a various number of Chebyshev nodes L. Time appears as the fraction of the time needed to perform the interpolation using the bbFMM over the time needed for the standard RBF interpolation (evaluation of equation 6.1). The kernel used for the summation is $\phi = 1/\sqrt{r^2 + 1}$. The bar chart and FMM computational time refer to the left vertical axis, whereas the circles to the right one, which is in the logarithmic scale. The bar chart reports the time required by the main operators of the FMM. The sum of the time required by the operators is not the total time required by the FMM because some operators are computed in parallel. Left: Errors and times needed for the FMM-based summations using an octree with at most 500 sources and targets per leaf bin. Right: The same as the left chart, but using an octree with at most 1000 sources and targets per leaf bin. The direct computation takes 410 seconds. Times are averaged over 100 runs.

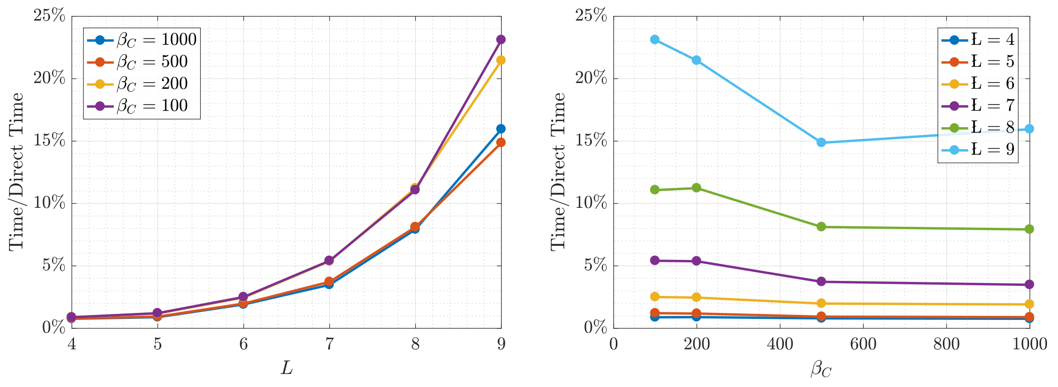


Figure 6.10: Uniform distribution. Time fractions and errors for the FMM-based summations for the same source and target distribution of Figure 6.9 and for a various number of Chebyshev nodes L and octree bin capacities β_C. Time appears as the fraction of the time needed to perform the interpolation using the bbFMM over the time needed for the standard RBF interpolation (evaluation of equation 6.1). The kernel used for the summation is $\phi = 1/\sqrt{r^2 + 1}$. Left: Time fractions for a various number of Chebyshev nodes L. Right: Time fractions for various octree bin capacities β_C.

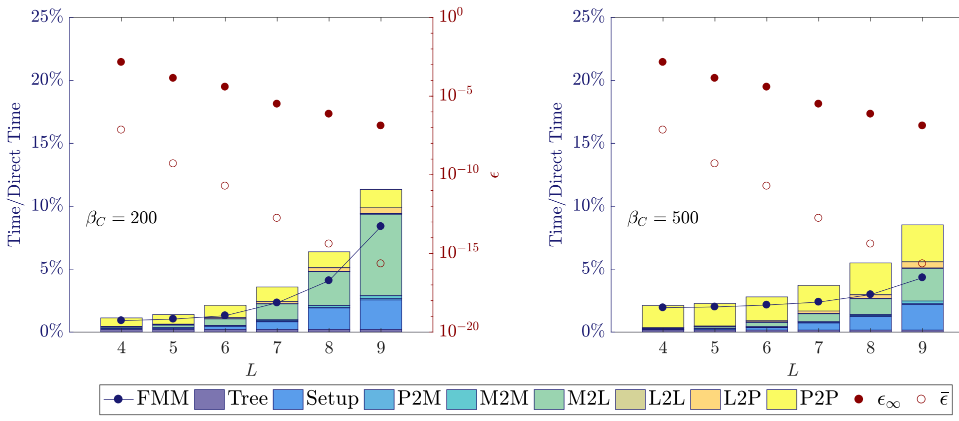


Figure 6.11: TUB compressor stator CFD mesh. Same as Figure 6.9 but for the TUB mesh with stretched mesh layers for viscous simulations with ~1.40×10^5 sources (surface mesh nodes) and ~2.31×10^6 targets (all mesh nodes). The octree bin capacity β_C is also different.

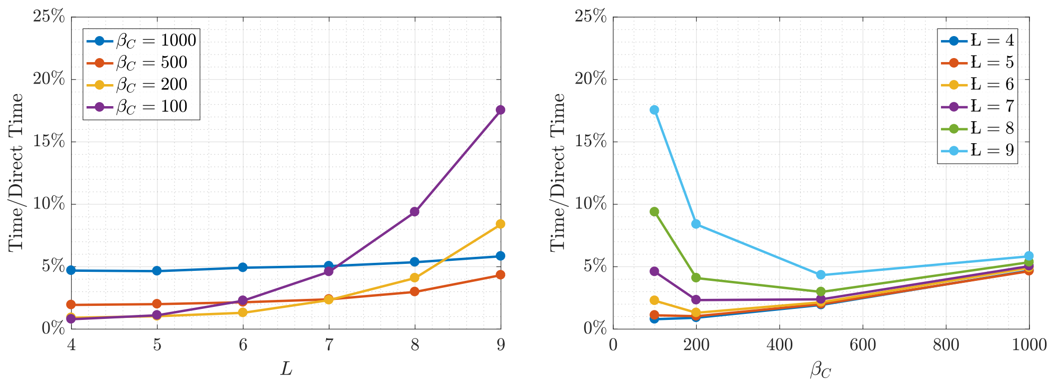


Figure 6.12: TUB compressor stator CFD mesh. Same as Figure 6.10 but for the TUB mesh.

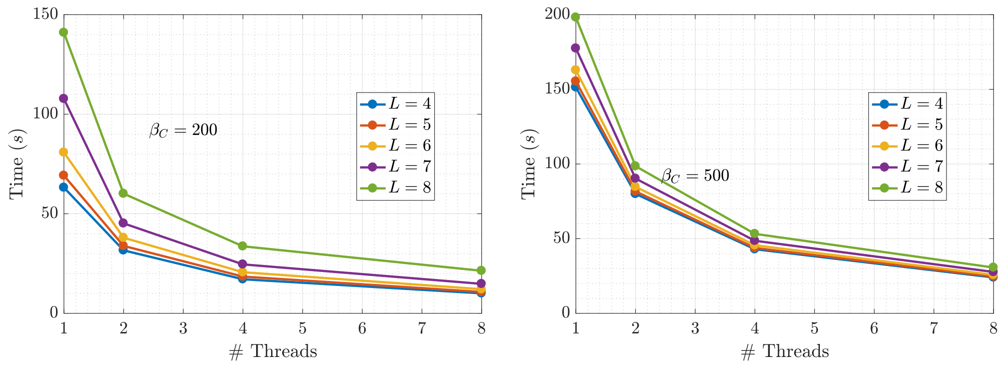


Figure 6.13: TUB compressor stator CFD mesh. Wall-clock times for the FMM-based summations for the TUB mesh for a various number of Chebyshev nodes L and octree bin capacity β_C as a function of the number of cores.

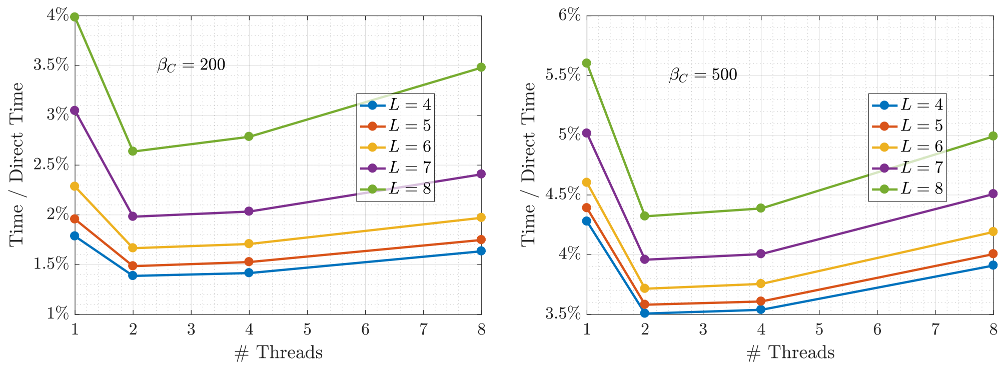


Figure 6.14: TUB compressor stator CFD mesh. Normalized wall-clock times for the FMM-based summations for the TUB mesh for a various L and β_C as a function of the number of cores. Time is normalized w.r.t. the wall-clock time needed for direct computation for the same number of threads.

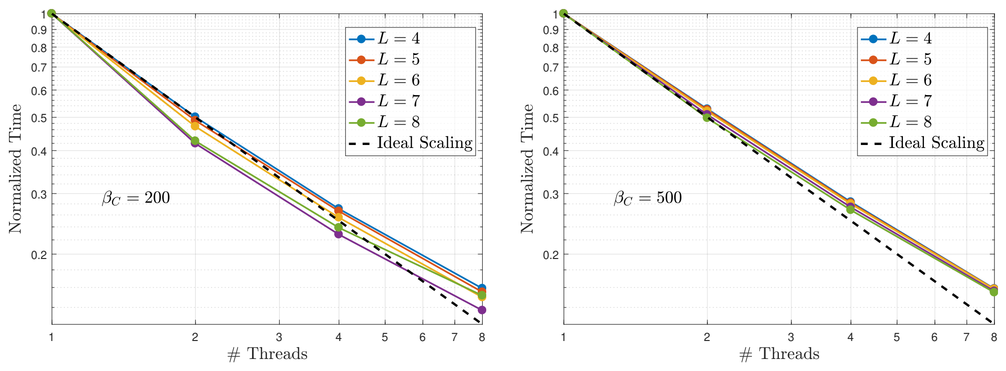


Figure 6.15: TUB compressor stator CFD mesh. Parallel speedup using OpenMP for the FMM-based summations for the TUB mesh for a various L and β_C as a function of the number of cores.

using POSIX threads at the coarser level and multithreaded BLAS routines at the finer level. [132] suggested OpenMP implementations based on the fork-join model. An "omp-parallel-for" pragma directive with static scheduling is used to loop over all bins on each level of the octree. In [133], both MPI and OpenMP are used. [132] also employs SIMD instructions to exploit processors parallelism for an adaptive FMM. A task-based approach is examined in [130].

The implemented software based on the bbFMM algorithm is parallelized with a task flow model for shared memory architectures. In this model, synchronizations are performed between successive levels of the octree and interleaving far-field and near field tasks. Diverse technologies can be used to implement the parallel software. The implemented one relies on SIMD vectorization (see [134]) and OpenMP directives (see [135]).

Near- and far-field are the two primary tasks of any FMM. In fact, near- and far-field contributions can be computed independently and then summed. In detail, the interaction of the various FMM is summarized as follows. The P2P, P2M and L2P operators, concerns only the leaf bins. The M2M and L2L operators always involve two octree levels, unlike the M2L, which operates only on one level at a time. A bottom-up octree traversal performs in order the P2M and M2M operators, then the M2L operator is performed on each level and bin independently, finally a top-down traverse performs the L2L and L2P operators. The P2P can be performed at any time, but when the L2P operator is being performed.

Such interactions lead to the following parallelization strategy. Considering that the M2L and P2P operators are the most expensive, "omp-sections" are used to compute in parallel the P2P operators and the sequence of operators P2M, M2M, M2L and L2L. The L2P operators follow these "omp-sections". Into these two "omp-sections", the FMM operators are further parallelized employing the "omp-task" directives. Synchronization points ("omp-taskwait directives") are required between the P2M, M2M, M2L and L2L operators and for each octree level when performing the M2M and L2L operators. However, the numerous synchronization points required by the far-field computation are compensated by the concurrent task insertion made by the near-field section.

Benefits are also obtained by the parallelization of the octree data structure set-up. During the octree construction, a divide and conquer technique is used. Such a strategy is implemented as a recursive algorithm. The root level is divided into eight octants, and each octant is treated as a new octree. After a "sub-octree" is constructed, it is merged into the parent octree. The procedure is repeated up to a certain octree level that guarantees a balance between the need to copy the data to initialize and merge a new "sub-octree" and the lowered complexity for building such a sub-octree on a reduced set of particles. Each sub-octree is, then, independent of the others at the same level, and it is constructed in parallel.

Figures 6.13, 6.14 and 6.15 illustrate the wall-clock time scaling as a function of the number of threads used. In detail, Figure 6.13 illustrates the wall-clock time for an increasing number of threads: the time needed to perform the FMM-based summation diminishes with the number of threads used. Figure 6.14 illustrates the wall-clock time for an increasing number of threads normalized w.r.t. the wall-clock time needed by the direct summation for the same amount of threads. Figure 6.15 illustrates the parallel speed-up of the programmed software compared to the ideal one: up to six threads, the programmed software scales almost ideally. For large L, the deviation from the ideal speed-up is larger.

Chapter 7

Optimization of a Compressor Stator

The TUB TurboLab Stator Blade is a typical turbomachinery optimization test case. Geometric constraints strongly influence the optimized shape. Previous chapters illustrated various test cases to demonstrate the method at hand; the test case presented in this chapter uses most of the methods previously exposed to perform the optimization.

Two parameterization methods are used: GMTurbo and the B-Rep-Morpher. The blade is imported in GMTurbo by means of the re-parameterization tool. The mesh is adapted to any shape change by using the surface mesh adaptation tool and the RBF-based mesh displacement tool. The blade is optimized with SQP-based and EA-based (EASY) methods. The fluid flow is computed using the in-house GPU-based RANS solver, PUMA. SQP-based methods use the gradients computed by the continuous adjoint to the RANS solver and the differentiated parameterizations. Results concerning shapes and fluid flows are illustrated for the various combinations of parameterization and optimization methods.

7.1 Introduction

The TurboLab Berlin compressor stator common research model has been tested in the measurement rig of the Chair for Aero Engines at TU Berlin. In the past, several research groups from academia and industry studied this case [136–139], and experiments were conducted on the original geometry and one optimized by I. Vasilopoulos [140, 141]. This stator is an aerodynamic model of a blade typically found in modern jet engine compressors. The geometry in CAD format can be downloaded from [142]. The annular reference cascade illustrated in Figure 7.1 is formed by 15 untwisted shrouded blades with

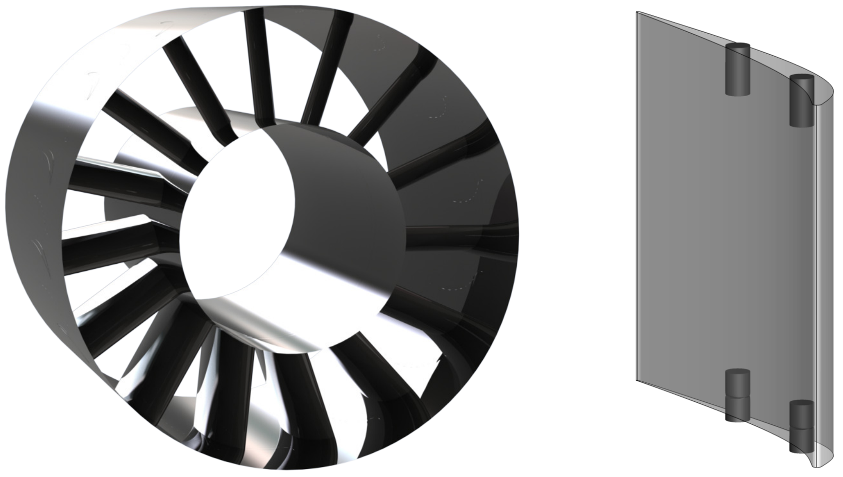


Figure 7.1: TUB stator. Left: Reference row assembly. Right: Reference geometry of a single blade with the placeholder cylinders for the fixture holes.

a spanwise constant chord. This case is used with both the GMTurbo, Section 7.3.1, and B-Rep-Morpher tools, Section 7.3.2, in the frameworks of a gradient-free multi-objective optimization, Section 7.4.1, and gradient-based constrained optimizations, Section 7.4.2. The optimization set-up, including the mesh, objective, and constraints, are presented in Section 7.2.

7.2 Optimization Set-Up

The functional considered are the mass-averaged total pressure losses between the inlet and outlet of the CFD domain and the deviation of the mass-averaged exit flow from the axial direction. The mass-averaged total pressure losses are defined in equation 2.16; the mass-averaged deviation of the exit flow from the axial direction is defined as

$$\alpha = \left(\frac{\displaystyle\int_{S_O} \left(\cos^{-1}\left(\frac{V_a}{\|\boldsymbol{V}\|} \right) \right)^2 \rho V_a \ \mathrm{d}S}{\displaystyle\int_{S_O} \rho V_a \ \mathrm{d}S} \right)^{\frac{1}{2}} \frac{180}{\pi} \,, \tag{7.1}$$

where V_a is the axial velocity component. These two functionals are contradicting since to achieve higher flow turning, the pressure losses are expected to increase. Higher flow turning is herein required to achieve a more axial outflow.

The shape is optimized for the operating point with the inlet total temperature, inlet total pressure and inlet whirl angle spanwise profiles of Figure 7.2; these lead to average

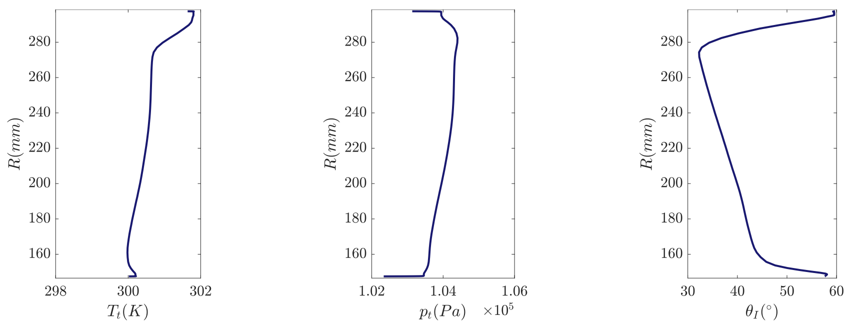


Figure 7.2: TUB stator. From left to right, radial inlet profiles for the total temperature, total pressure and whirl angle.

inlet total temperature of 300.52K, average inlet total pressure of 1.04×10^5 Pa and average inlet whirl angle of 39.83°. The pitch angle and inlet turbulence intensity are equal to 0° and 4%, respectively, being spanwise uniform. The back-pressure is adjusted so as to give the desired mass flow rate of 9.0 kg/s (full annulus). A steady RANS compressible flow solver with the Spalart-Allmaras turbulence model is used to compute the flow. This solver is part of the in-house PUMA flow solver suite (Appendix E).

The CFD mesh is block-structured with viscous layers; it consists of ~2.32×10^6 nodes, ~1.43×10^5 out of which are surface nodes. The surface mesh is illustrated in Figure 7.3. The mesh has been selected after carrying out a mesh independence study to achieve a nearly mesh-independent computation of the objectives. The maximum y^+ value of the first series of nodes off the walls is below one with a margin that allows morphed meshes to have max $y^+ < 1$ still.

The essential geometric characteristics are given in Figures 7.4 and 7.5. Figure 7.5 also illustrates some of the geometric constraints that must be fulfilled, specifically:

- There are 15 blades.
- The axial chord of the blade is kept constant.
- The leading and trailing edge radii must be greater than 1 mm.
- The thickness of the blade cannot decrease.
- The blade must be mountable in the casing (hub and shroud). Specifically, two holes for the fixture in the middle of the blade with a radius of 2.5 mm and a depth of 20 mm are required. Thus, the blade thickness at these positions has to accommodate a cylinder of material with a depth of 20 mm and a radius of 5 mm to allow for cutting of the thread at both hub and shroud. The two holes can be placed arbitrarily inside the profile shape but have to be at least 60 mm apart from each other.

Most geometric constraints are handled by imposing lower and upper limits on the design

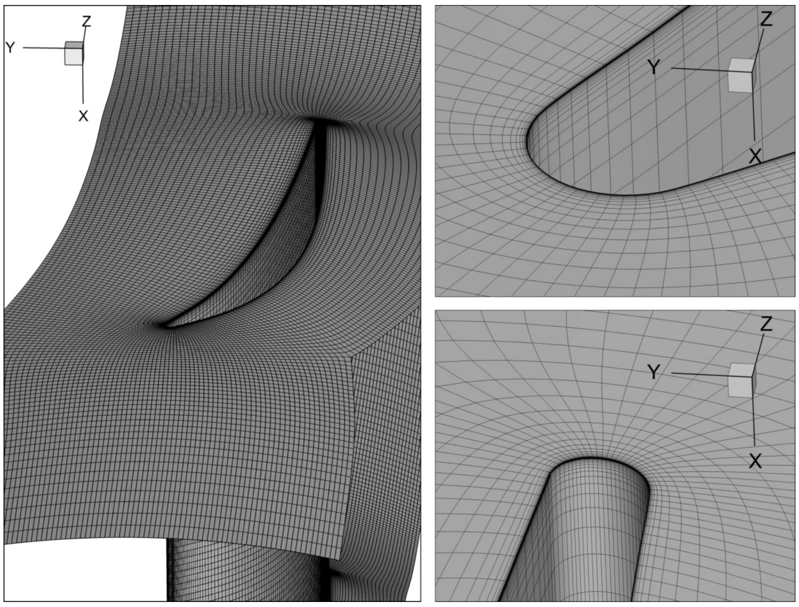


Figure 7.3: TUB stator. Computational surface mesh of the TUB stator baseline shape. Only solid walls are illustrated. Left: view from above the shroud surface. Right: details of the leading (top) and trailing (bottom) edges at the shroud.

variables. However, to fit the cylinders inside the blade, an appropriate constraint function is defined by exploiting the availability of the CAD description into the optimization loop. The way this constraint is imposed is illustrated in Section 7.3.3.

Before proceeding to the parameterization and optimization, a simulation of the reference geometry is carried out. Some results regarding the flow are illustrated in Figures 7.6, 7.7 and 7.8. The mass-averaged exit flow angle deviation α is equal to 6.27° for the reference geometry. Its distribution at the outlet is illustrated in Figure 7.8. The outlet flow angle highly deviates from the axial direction, which means that the reference stator does not achieve a satisfactory flow turning.

7.3 Shape Parameterizations

This section is organized as follows. Section 7.3.1 illustrates the TUB compressor row parameterization based on GMTurbo. Analogously, Section 7.3.2, illustrates the parameterization based on the B-Rep-Morpher. Section 7.3.3 illustrates the parameterization of the fixture holes used to handle the geometric constraint. Both GMTurbo and the B-Rep-Morpher parameterizations are differentiated, with the method illustrated in Section 4.2, to support gradient-based optimization using the adjoint method to compute the gradients. Although the cost of computing geometric sensitivities with such a method scales with the

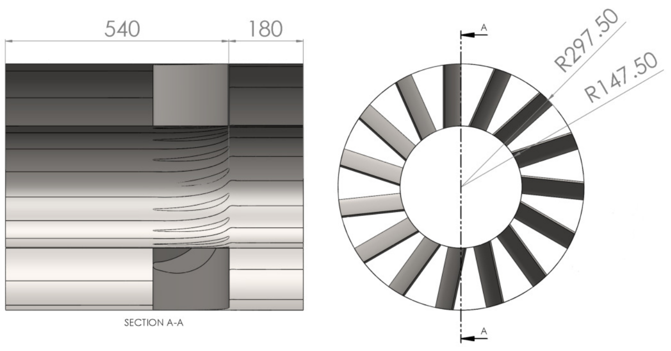


Figure 7.4: TUB stator. Geometric definitions illustrated for the baseline geometry. Dimensions are in millimeters.

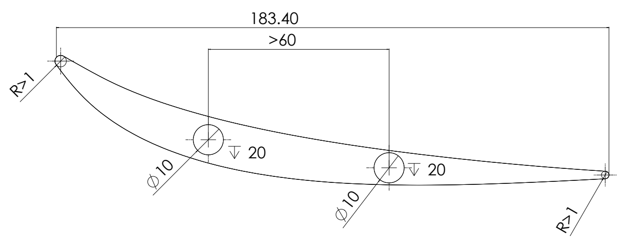


Figure 7.5: TUB stator. Geometric definitions illustrated for the baseline geometry at the hub and shroud. Dimensions are in millimeters.

number of design parameters, the total cost for the parameters used in this case is negligible compared to the cost of the primal and adjoint solutions. The surface mesh is adapted to shape changes of the NURBS surfaces by using the method illustrated in Section 4.1.

7.3.1 Parameterization with GMTurbo

The process described in Section 3.3 is used to extract the design parameters used by GMTurbo, Chapter 3, to generate the geometry. Two blade sections were used to extract the parameters; the original blade in CAD format is produced by extruding a planar airfoil so that two generatrices are sufficient to capture the geometry with the GMTurbo parameterization.

Figure 7.9 illustrates the input curves. Even if quantities are constant, few control points are inserted into the curves in order to add flexibility to the parameterization while keeping the blade representation compact. The total number of DoFs used by the optimization is 32 and distributed as follows: *(a)* 6 DoFs for β_{LE}, *(b)* 6 DoFs for β_{TE}, *(c)* 4 DoFs for θ_{LE}, *(d)* 4 DoFs for θ_{TE}, *(e)* 6 DoFs for ζ_{LE} and *(f)* 6 DoFs for ζ_{TE}. The first of the five control points of θ_{LE} is kept fixed to avoid mere displacement of the blade. The meridional projection, the casing and the thicknesses of the blade are kept constant.

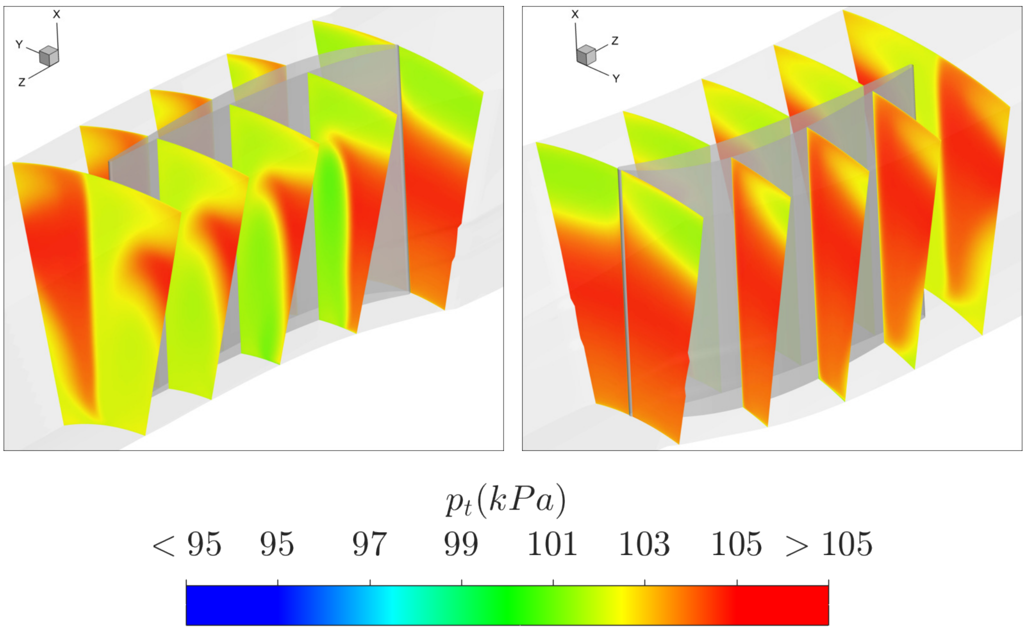


Figure 7.6: TUB stator. Total pressure contour plots on transversal planes.

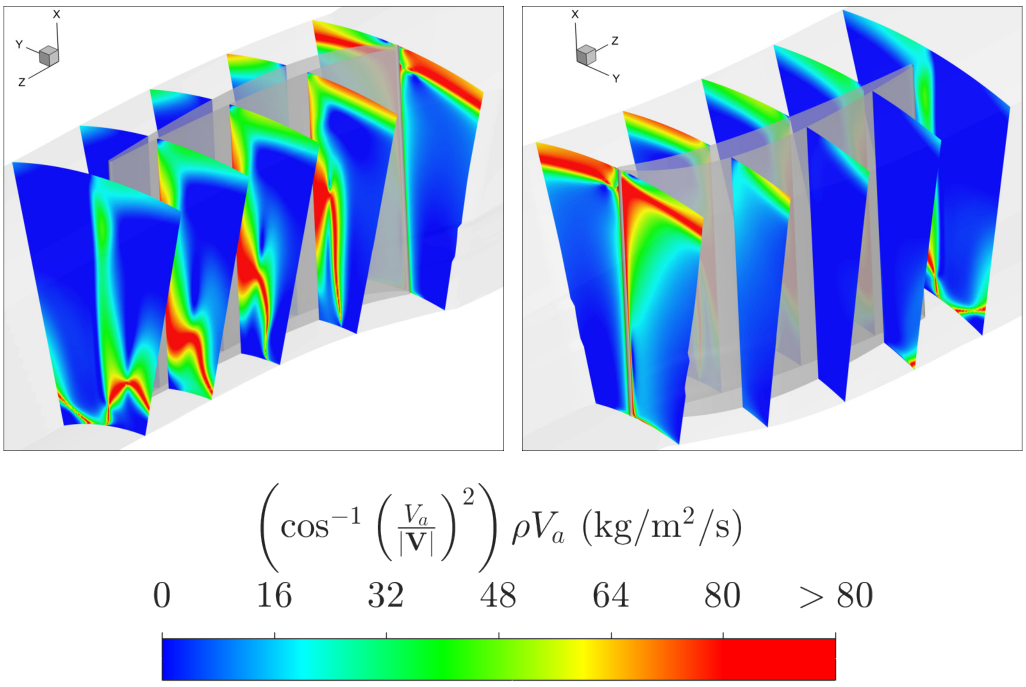

X
Y
Z
$\left(\cos^{-1}\left(\frac{V_a}{|\mathbf{V}|}\right)^2\right)\rho V_a\ (\mathrm{kg/m^2/s})$
0
16
32
48
64
80
> 80

Figure 7.7: TUB stator. Contour plots of the deviation of the flow angle from the axial direction.

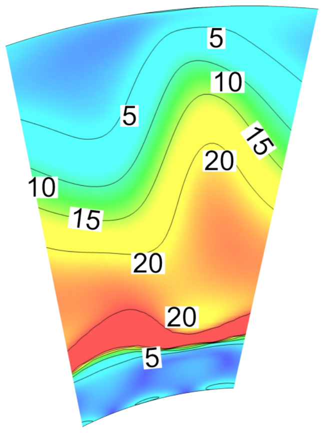


Figure 7.8: TUB stator. Contour plots of $\left(\cos^{-1}\left(\frac{V_a}{|\mathbf{V}|}\right)\right)^2 \rho V_a$ (kg/m^2/s) at the outlet of the baseline shape of the stator.

7.3.2 Parameterization with the B-Rep-Morpher

The B-Rep-Morpher is described in Chapter 2. The parameterization employs a two-step deformation method: the first step is controlled by 6 and the second by 24 RBF handles. Handles are displaced by changing their cylindrical coordinates in the azimuthal direction so that each one gives a DoF for the optimization. Displacing the blade only in the azimuthal direction of the cylindrical coordinates guarantees that the axial chord of the blade remains constant, as required by the geometric constraints. Figure 7.10 illustrates the handles. Hub and shroud must remain surfaces of revolution: because a non-rigid displacement of the NURBS control points would violate this constraint, these surfaces are excluded from the morphing action, and their p-curves are updated by re-trimming as described in Section 2.3.

7.3.3 Parameterization of the Fixture Holes

The fixture holes constraint has been described in Section 7.2. Checking if the constraint is fulfilled (both cylinders fit) or violated (one or both cylinders do not fit) sums up to checking if an intersection between the NURBS description of the holes and blade profiles exists. However, for better constraint handling, two continuous functions are used during the optimization, one for the hub and another for the shroud, defined as follows: *(a)* if the constraint is fulfilled, the function value is equal to the minimum distance between the cylinders and the blade profiles or *(b)* if the constraint is violated, the function value is equal to the additive inverse of the cubic root of the minimum volume intersection between the cylinders and the CFD domain. By naming C such a function (namely C_{HUB} for the holes at the hub and C_{SHROUD} for the holes at the shroud):

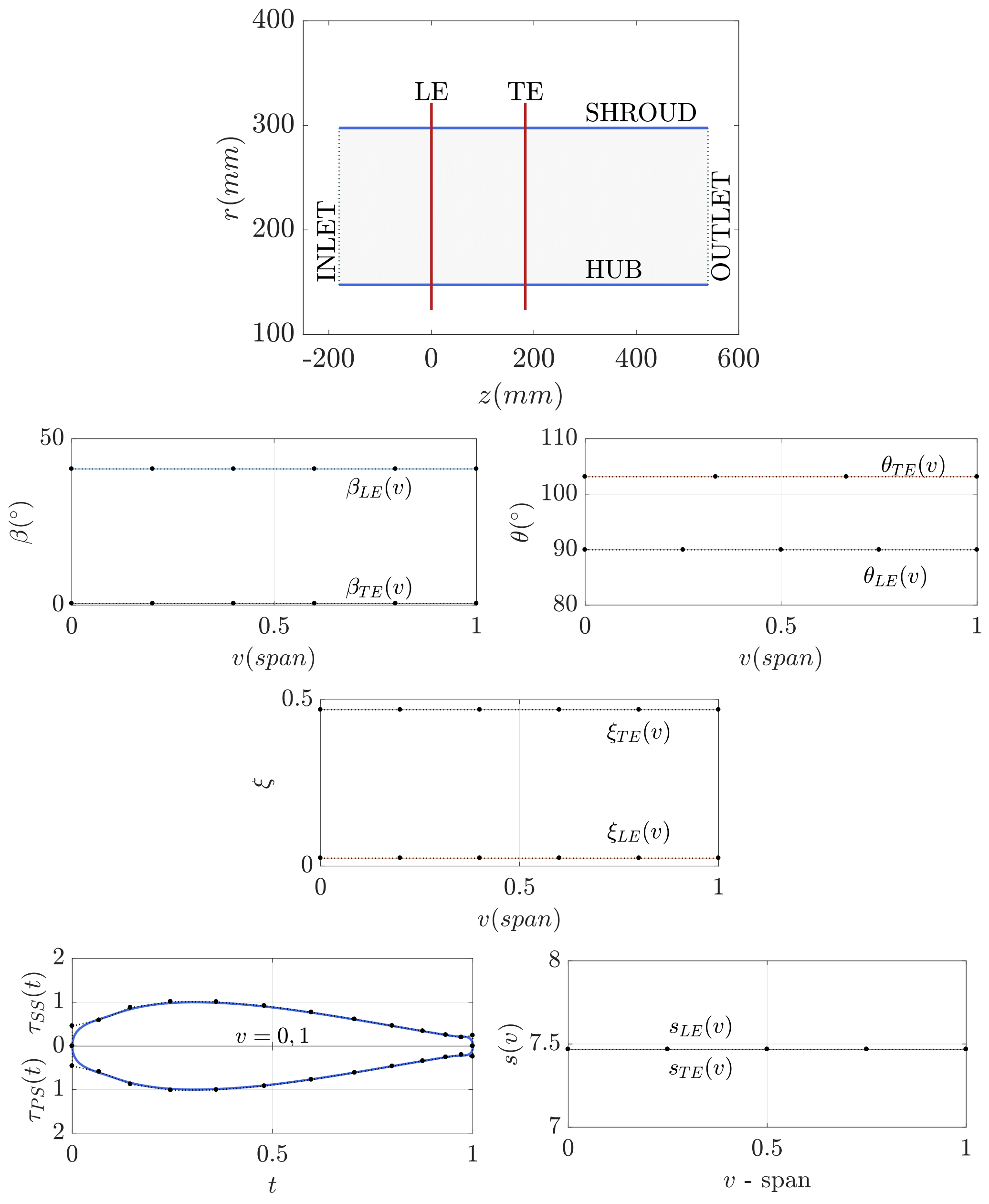


Figure 7.9: TUB stator. GMTurbo input curves. In the meridional shape (top), only the hub, shroud, TE and LE curves are inputs. The control points of the NURBS curves (black points) are also illustrated.

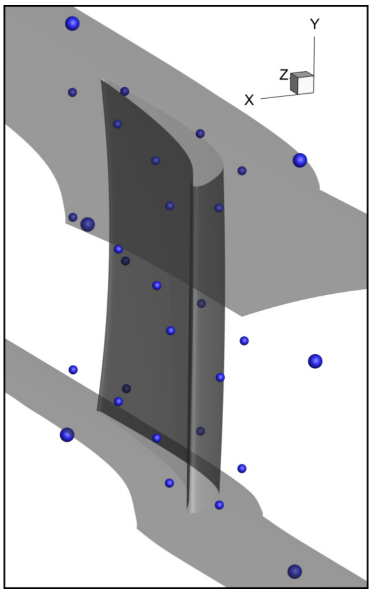


Figure 7.10: TUB stator. The B-Rep-Morpher parameterization using the handles (blue spheres) displaced in the azimuthal direction as DoFs is illustrated. The blade shape is altered by displacing the handles using the two-step approach. The first step is based on 6 handles (large spheres) whereas the second on 24 handles (smaller spheres).

$$C = \min_{u_1,v_1,u_2,v_2,\alpha_1,\alpha_2} c \tag{7.2}$$

where

$$c = \begin{cases} \sqrt[3]{V_\cap} & \text{if} \quad V_\cap > 0 \\ -d & \text{if} \quad V_\cap = 0 \end{cases} . \tag{7.3}$$

$V_\cap$ is the volume of the cylinders that fall outside the blade profiles, d is the minimum distance between the cylinders and the blade profiles. Computing C requires to minimize the quantity c: $u_1, v_1, u_2, v_2, \alpha_1$ and α_2 are the DoFs of the holes parameterization. Specifically, u_1, v_1 and u_2, v_2 are pairs of parametric coordinates on the hub or shroud NURBS surfaces, $\boldsymbol{S}$; they are used to determine the cylinder base's centers $\boldsymbol{S}(u_1, v_1)$ and $\boldsymbol{S}(u_2, v_2)$. α_1 and α_2 are the angles of inclination of the cylinder in the peripheral direction. Figure 7.11 illustrates these 6 DoFs. Constraints on the minimization problem are also considered: the cylinders must be at least 60mm apart, and the cylinder base centers must be inside the blade profiles. The constrained fitting problems are solved by EA-based optimizations.

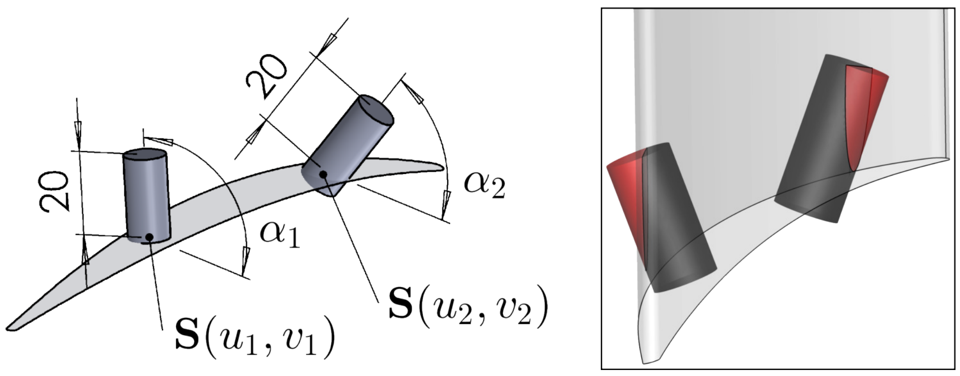


Figure 7.11: TUB stator. Left: Illustration of the six DoFs, u_1, v_1, u_2, v_2, α_1 and α_2, used to find the two holes for the fixture that fit in the blade profiles, if any. The airfoil at the bottom is the portion of the hub surface trimmed by the blade surfaces. Right: the volume of the cylinders that does not fit into the blade profile is illustrated in red.

7.4 Results

Section 7.4.1 illustrates the performance and shapes of the blades found by running an EA-based shape optimization with the software EASY (Appendix D). Similarly, Section 7.4.2 illustrates the result of gradient-based constrained optimizations. The outcomes of the optimizations are not comparable since they are run with different bounds on the design variables.

The literature contains results regarding the TUB shape optimization. For instance, [141] introduced 192 DoFs and optimized the blade for three operating points using a weighted combination of total pressure losses and stator exit flow angle (formulated differently than in this thesis) as objective function and slightly different test case conditions. By giving equal weights to the two normalized objectives, [141] decreased the exit flow angle from 4.56° to 2.8° and the total pressure losses by 3.2% w.r.t. the baseline. Comparatively, the Pareto member (resulting from the analysis illustrated in Section 7.4.1) marked with [2] in Figure 7.12 reduces the exit flow angle from 6.27° to 4.18° and total pressure losses by 0.8% (relative difference) w.r.t. the baseline, employing 30 DoFs. [139] optimized the TUB shape w.r.t. the total pressure losses and the exit whirl angle distribution by means of a differentiated CAD tool and a differentiated CFD solver, with 192 DoFs. The optimization reduced the exit whirl angle of about 2° while keeping the total pressure losses nearly constant. Comparatively, the results illustrated in Figure 7.19 (resulting from the analysis illustrated in Section 7.4.2) yields a reduction of the whirl angle of 3.75° while keeping the total pressure losses nearly constant and employing 32 DoFs. However, differences in the test case conditions and constraints used prevent a full comparison.

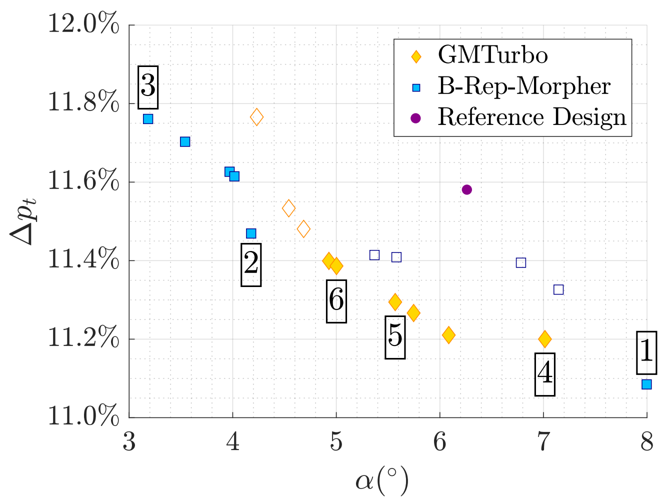


Figure 7.12: TUB stator. EA-based optimization with two parameterization methods: the fronts of non-dominated solutions computed using the B-Rep-Morpher and GMTurbo are compared. The two independent optimizations were performed at the same computational cost. The combined front (final set of non-dominated solutions by considering both fronts) is shown as filled symbols. The shapes of the six Pareto points, marked with integers, are compared to the reference design in Figure 7.13 and 7.14.

7.4.1 Gradient-Free Optimization

The objectives of both optimizations, using GMTurbo and the B-Rep-Morpher, are min. total pressure losses and min. deviation of the exit flow angle from the axial direction. A MAEA, provided by EASY, is used for the two-objective optimization: RBF networks are used as metamodels to speed-up the optimization process. The overall computational budget is restricted to 500 CFD solver calls. The EA is run with a population of 30 offspring, 10 parents and 7 elite members. The crossover operator (two-point crossover) uses 3 parents to generate one offspring and the mutation probability is 0.1%. The results of the optimizations are compared in Figure 7.12. The B-Rep-Morpher-based optimization computes a front of non-dominated solutions similar to that computed by the dedicated CAD kernel, GMTurbo. Even if the termination criteria prevent full convergence, both optimizations captures different parts of the front of non-dominated solutions.

Some of the shapes resulting from the optimization are shown in Figure 7.13 and 7.14. Figures 7.15 and 7.16 illustrate fields related to the two objectives at the outlet and midspan of the CFD domain.

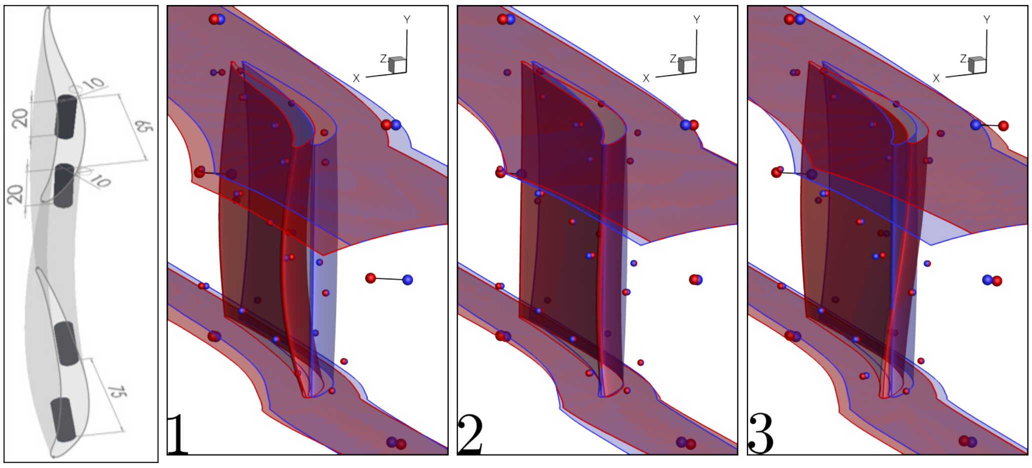


Figure 7.13: TUB stator. B-Rep-Morpher. Left: Compliance with the geometric constraint regarding the fixture holes for the shape marked with “1” (dimensions are in millimeters). Right: Three shapes (red) from the front on non-dominated solutions in Figure 7.12, in comparison with the reference shape (blue), with the corresponding handle displacements. The shape-morphing action is based on a two-step deformation: the handles of the first step are shown with bigger symbols.

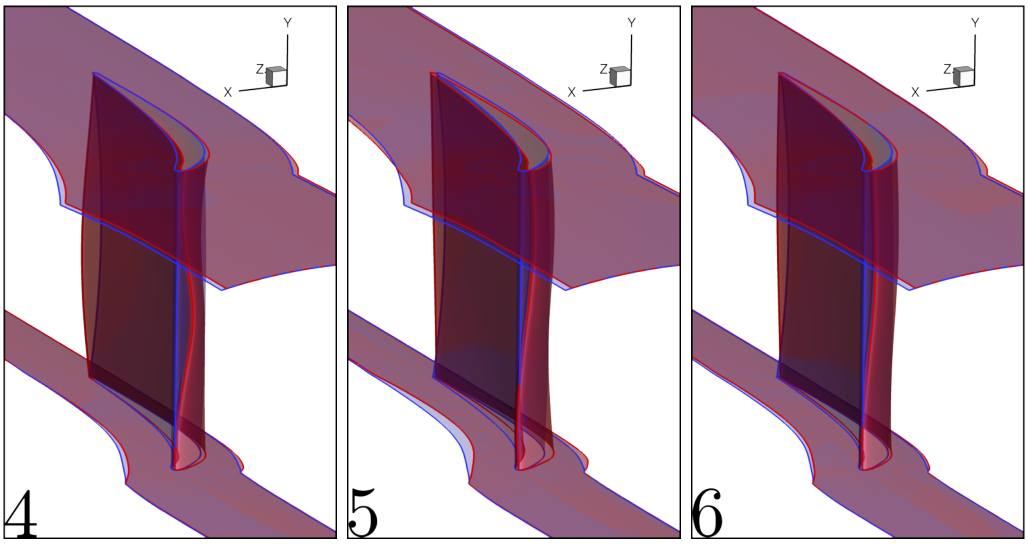


Figure 7.14: TUB stator. GMTurbo. Right: Three shapes (red) from the front on non-dominated solutions in Figure 7.12, in comparison with the reference shape (blue).

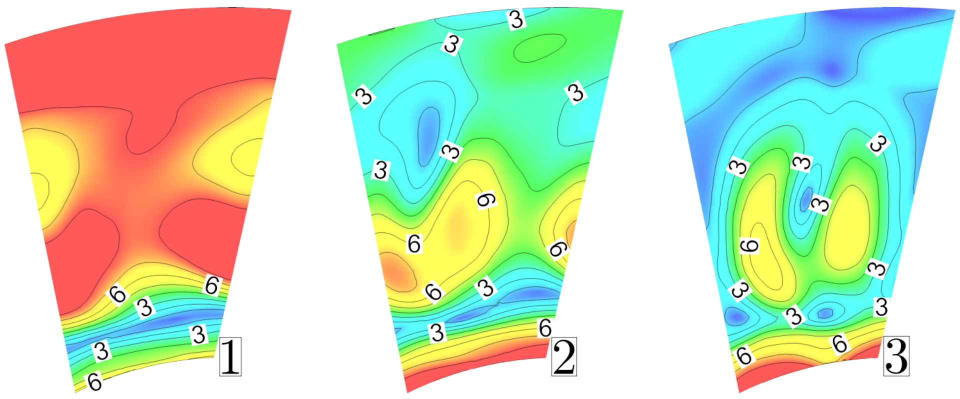


Figure 7.15: TUB stator. B-Rep-Morpher. $\left(\cos^{-1}\left(\frac{V_a}{|\mathbf{V}|}\right)\right)^2 \rho V_a$ (kg/m^2/s, see equation 7.1) iso-areas at the stator outlet for three designs from the front on non-dominated solutions of Figure 7.12. Higher values imply higher objective function values.

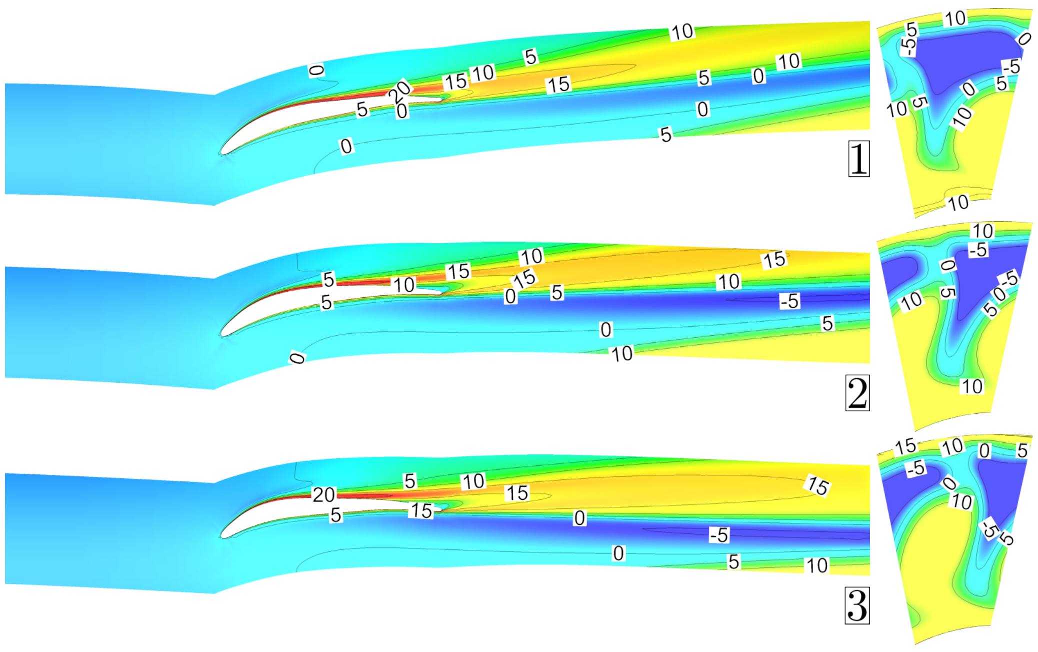


Figure 7.16: TUB stator. B-Rep-Morpher. $\left(\left(p_t - \int_{S_I} p_t \rho V_a \, dS \,/\int_{S_I} \rho V_a \, dS\right) / \left(\int_{S_I} (p_t - p) \rho V_a \, dS \,/\int_{S_I} \rho V_a \, dS\right)\right) \rho V_a$ (kg/m^2/s, see equation 2.16) iso-areas at the stator midspan (left) and outlet (right) for three designs from the front on non-dominated solutions of Figure 7.12. Higher values imply higher objective function values.

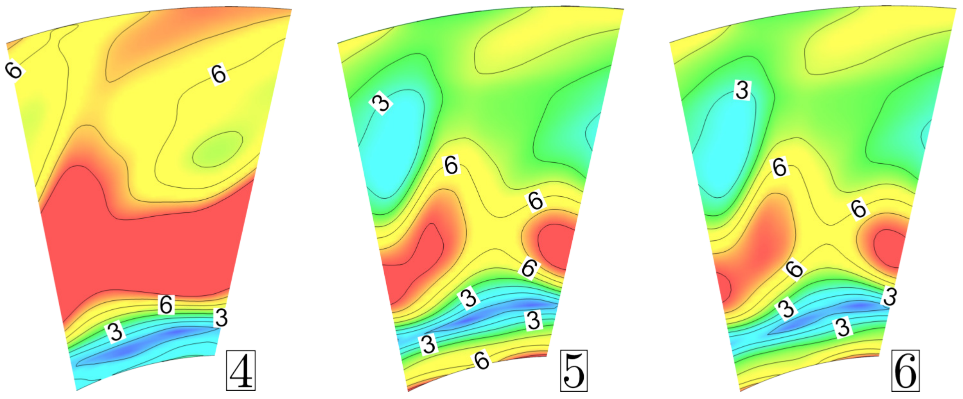


Figure 7.17: TUB stator. GMTurbo. $\left(\cos^{-1}\left(\frac{V_a}{|\mathbf{V}|}\right)\right)^2 \rho V_a$ (kg/m^2/s, see equation 7.1) iso-areas at the stator outlet for three designs from the front on non-dominated solutions of Figure 7.12.

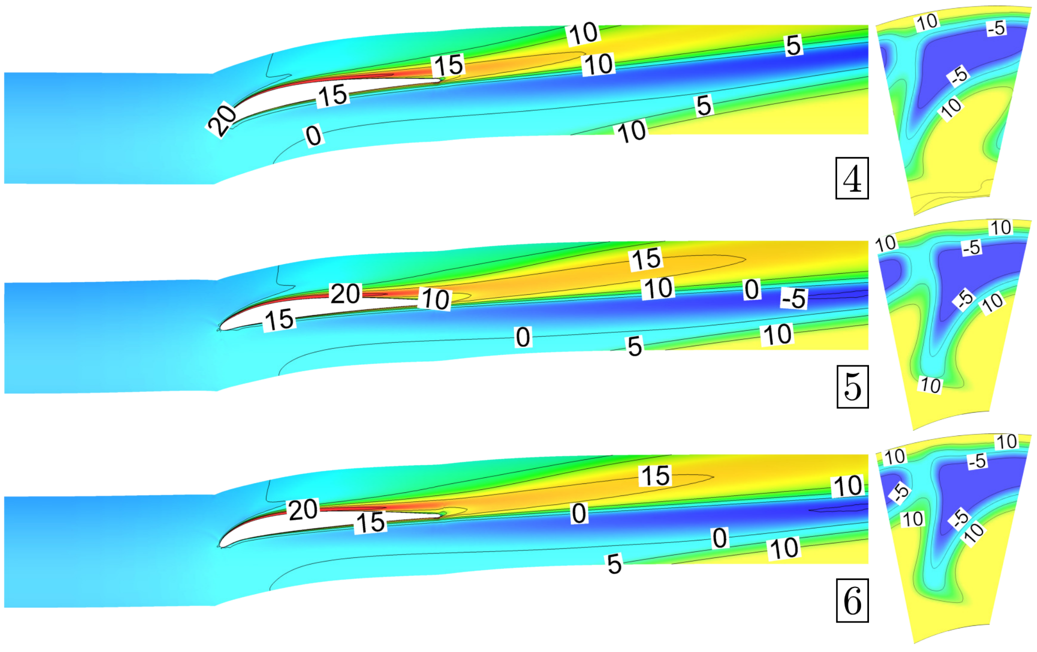


Figure 7.18: TUB stator. GMTurbo.
$\left(\left(p_t - \int_{S_l} p_t \rho V_a\, dS \,/ \int_{S_l} \rho V_a\, dS\right) / \left(\int_{S_l} (p_t - p)\rho V_a\, dS \,/ \int_{S_l} \rho V_a\, dS\right)\right) \rho V_a$ (kg/m^2/s, see equation 2.16) iso-areas at the stator midspan (left) and outlet (right) for three designs from the front on non-dominated solutions of Figure 7.12.

7.4.2 Gradient-Based Optimization

Here, gradient-based optimization deals with single-objective problems. An algorithm based on SQP is used with bounds for each design variable. Gradients are assessed by combining sensitivities computed by the continuous adjoint method available in the PUMA flow solver suite and those computed by differentiating the parameterization method.

Four studies are performed:

- use the GMTurbo parameterization to minimize α such that $\Delta p_t \leq \Delta p_{t0} + 1.5\%$. Δp_{t0} is the value of Δp_t for the initial shape.
- use the GMTurbo parameterization to minimize Δp_t such that $\alpha \leq \alpha_0$. α_0 is the value of α for the initial shape.
- use the B-Rep-Morpher parameterization to minimize α such that $\Delta p_t \leq \Delta p_{t0} + 1.5\%$.
- use the B-Rep-Morpher parameterization to minimize Δp_t such that $\alpha \leq \alpha_0$.

When considering Δp_t as a constraint, a tolerance is added, which allows finding more "interesting" geometries that stress more the software and methods under test due to higher mesh and shape displacements involved.

Optimizations converge to a stationary point without the necessity to regenerate the mesh due to invalid elements. Convergence histories are illustrated in Figure 7.19, 7.20, 7.21 and 7.22. SQP requires 8 to 12 iterations to reach a minimum.

The shapes of the blades resulting from the optimizations are illustrated in Figure 7.23, 7.24, 7.25 and 7.26. The optimal design variables w.r.t. α of the GMTurbo parameterization are illustrated in Figure 3.10. Figure 7.27 illustrates the cylinders' holes fitted into the blades' optimal shapes.

The optimization of α using GMTurbo reaches a value of 2.51° compared to 2.98° employing the B-Rep-Morpher. However, the total pressure losses of the GMTurbo-based optimization are 13.10%, which is close to the constraint value, in contrast to 12.58% of the shape found with the B-Rep-Morpher parameterization. Analogously, the optimization of Δp_t using GMTurbo reaches a value of 11.15% compared to 11.22% using the B-Rep-Morpher. However, the deviation of the exit flow angle from the axial direction of the GMTurbo-based optimization is 6.22°, which is, again, close to the constraint value, in contrast to 5.75° of the shape found with the B-Rep-Morpher parameterization.

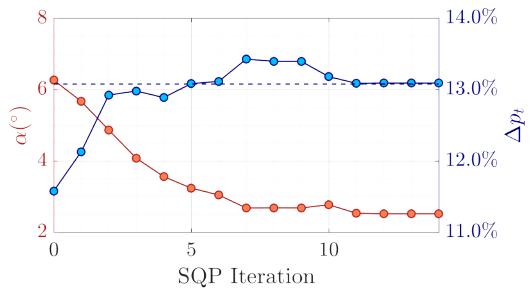


Figure 7.19: GMTurbo. SQP convergence history for the optimization case: $\min \alpha,\ \Delta p_t \leq \Delta p_{t0} + 1.5\%$. The corresponding final shape is illustrated in Figure 7.23.

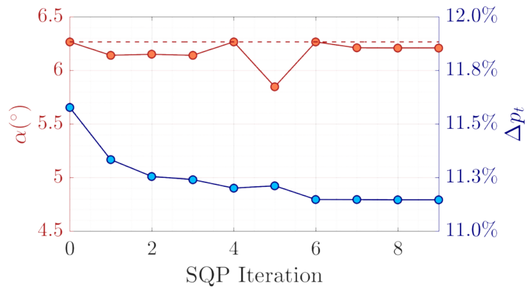


Figure 7.20: GMTurbo. SQP convergence history for the optimization case: $\min \Delta p_t,\ \alpha \leq \alpha_0$. The corresponding final shape is illustrated in Figure 7.24.

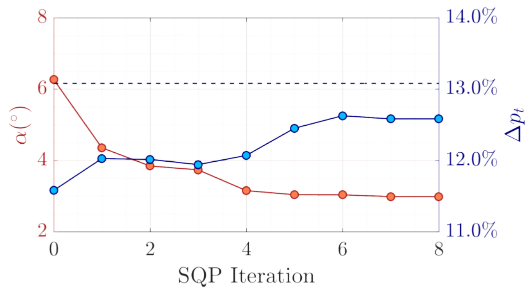


Figure 7.21: B-Rep-Morpher. SQP convergence history for the optimization case: $\min \alpha,\ \Delta p_t \leq \Delta p_{t0} + 1.5\%$. The corresponding final shape is illustrated in Figure 7.25.

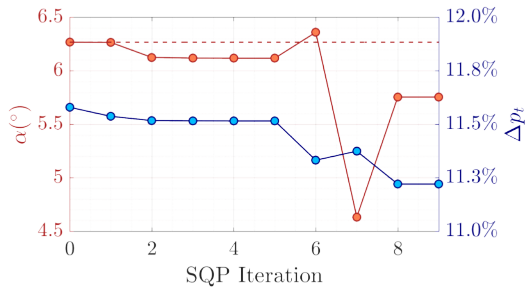


Figure 7.22: B-Rep-Morpher. SQP convergence history for the optimization case: $\min \Delta p_t,\ \alpha \leq \alpha_0$. The corresponding final shape is illustrated in Figure 7.26.

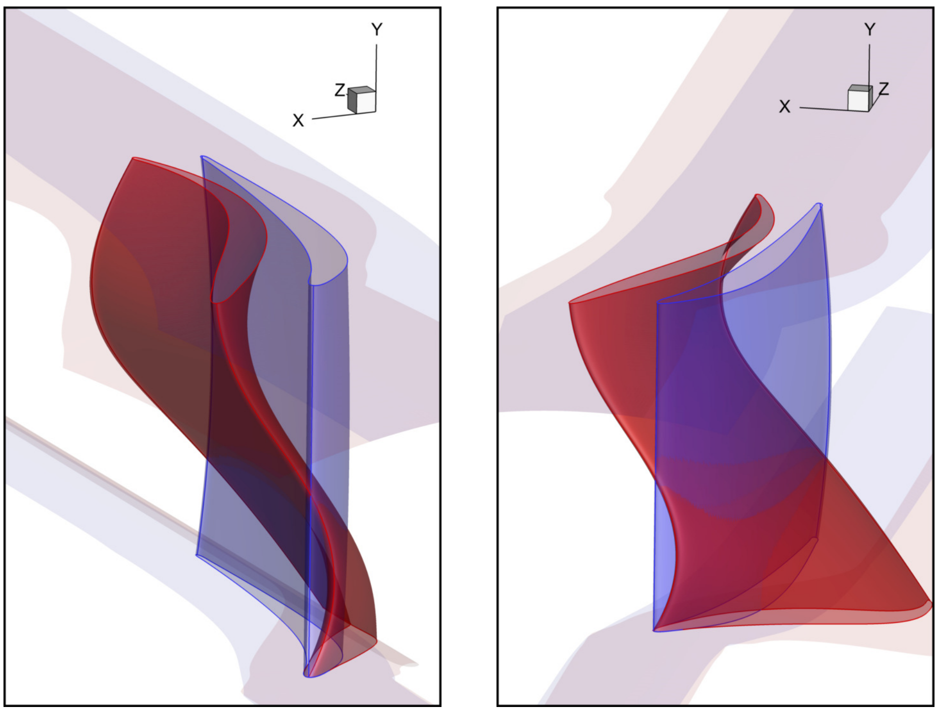


Figure 7.23: GMTurbo. Shape resulting from the optimization case: min α, $\Delta p_t \leq \Delta p_{t0}$+1.5%. The corresponding SQP convergence history is illustrated in Figure 7.19.

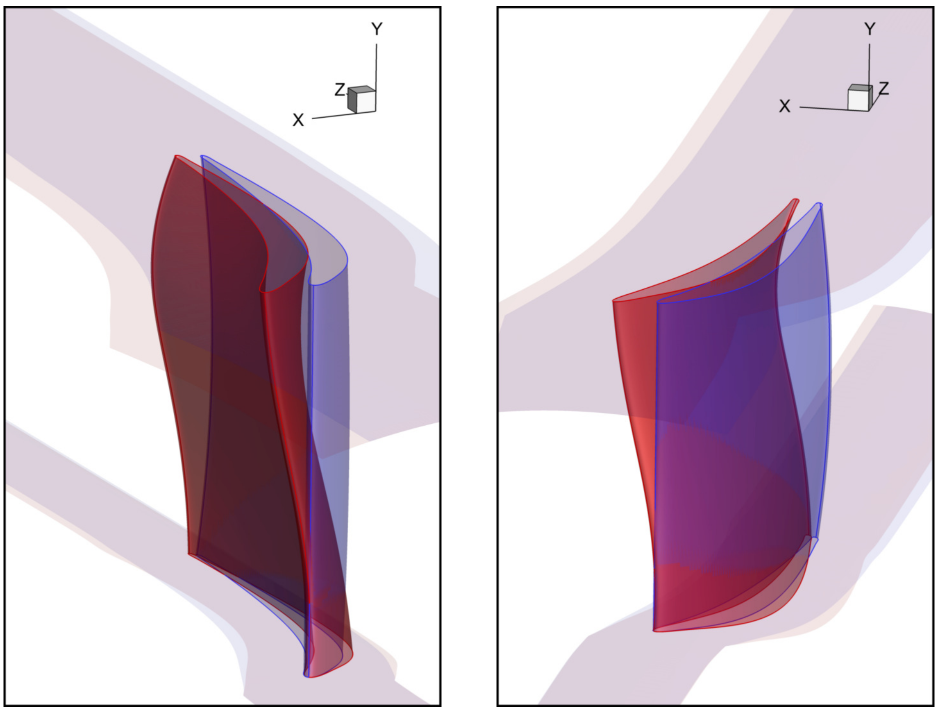


Figure 7.24: GMTurbo. Shape resulting from the optimization case: min Δp_t, $\alpha \leq \alpha_0$. The corresponding SQP convergence history is illustrated in Figure 7.20.

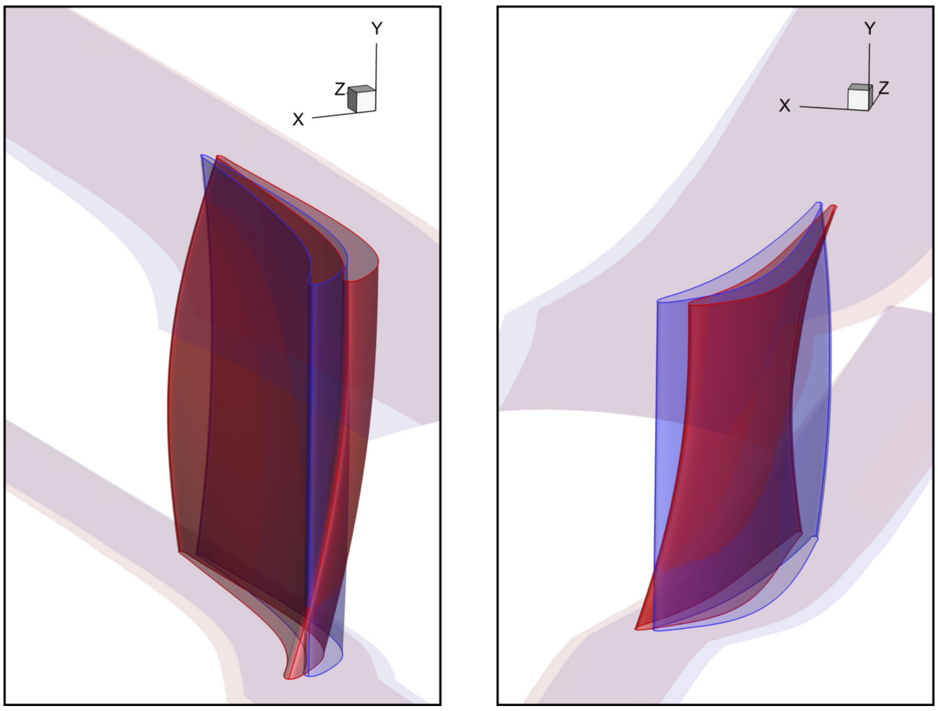


Figure 7.25: B-Rep-Morpher. Shape resulting from the optimization case: $\min \alpha,\ \Delta p_t \leq \Delta p_{t0} + 1.5\%$. The corresponding SQP convergence history is illustrated in Figure 7.21.

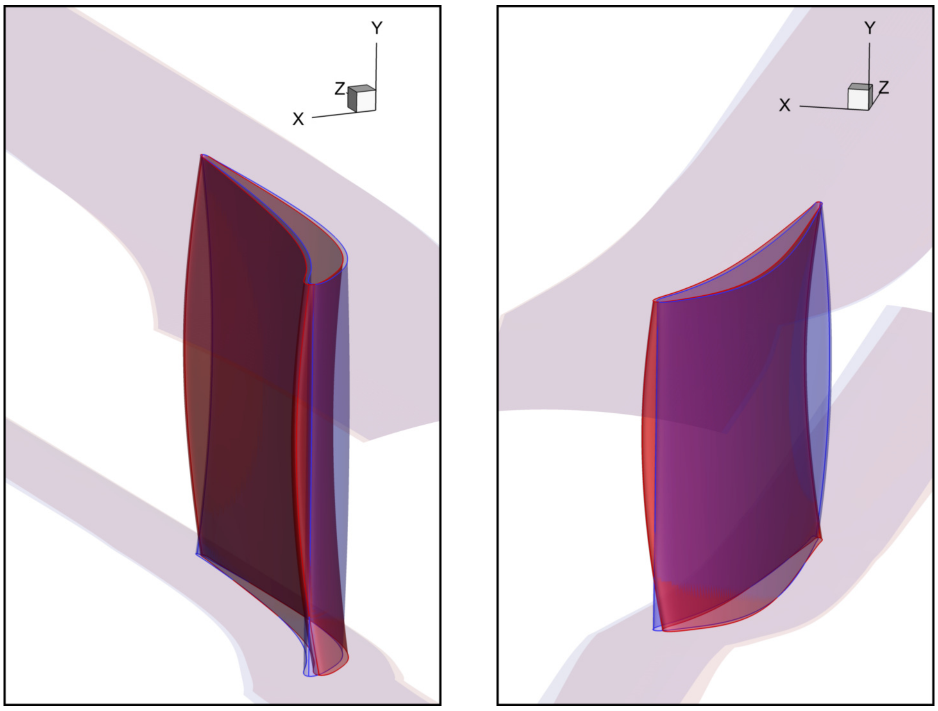


Figure 7.26: B-Rep-Morpher. Shape resulting from the optimization case:$\min \Delta p_t,\ \alpha \leq \alpha_0$. The corresponding SQP convergence history is illustrated in Figure 7.22.

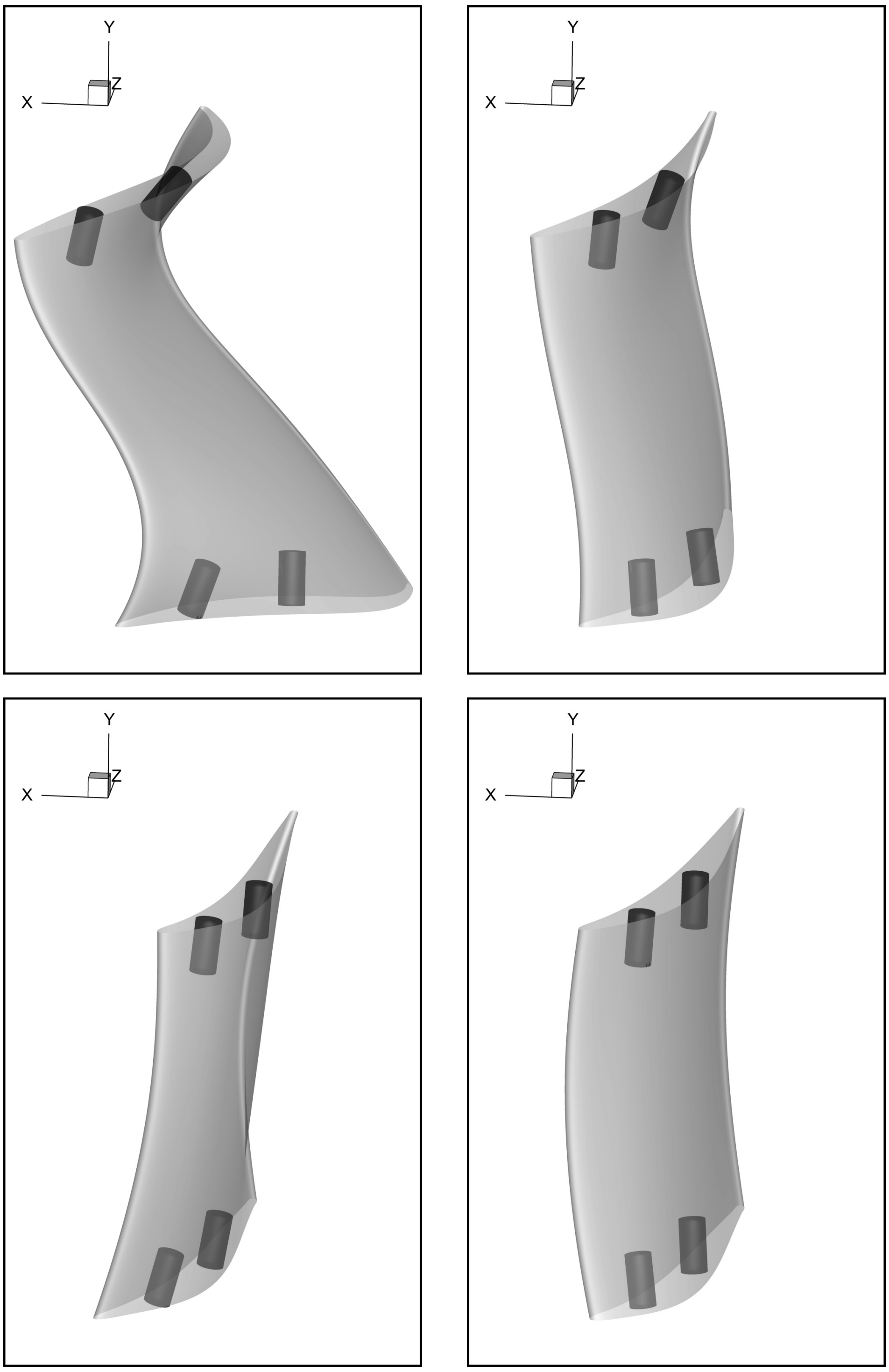


Figure 7.27: Cylinders fitted into the optimal shape of the TUB found by the gradient-based optimizations. Top: GMTurbo. Bottom: B-Rep-Morpher. Left: $\min \alpha,\ \Delta p_t \leq \Delta p_{t0} + 1.5\%$. Right: $\min \Delta p_t,\ \alpha \leq \alpha_0$.

Chapter 8

Closure

The aim of this thesis was the development of an automated and adaptable workflow for aerodynamic shape optimization. This dissertation focuses on CAD-based geometry parameterization, as well as mesh displacement, and their application in gradient-based (adjoint) and gradient-free CFD-based shape optimization problems.

Chapter 2 presented a new shape-morphing method, named B-Rep-Morpher, that acts directly on the NURBS-based B-Rep models of the bodies to be designed, which can thus be integrated into any aerodynamic (or hydrodynamic) shape optimization loop by ensuring that the optimal solution to be designed remains CAD compatible. The morphing action involves a small number of user-defined "handles" placed around or on the B-Rep shapes to be optimized. Their displacements are controlled by the optimization loop in a hierarchical or multi-step manner and affect the NURBS control points pertaining to the B-Rep model through RBF-based interpolations, with local or global support, depending on the step. Compared to B-Rep deformation methods that use the position of NURBS control points as design variables directly, the proposed method enables the parameterization of generic shapes provided in standard CAD format with a relatively small number of parameters. For instance, in Section 2.4.3, a wing with tens of thousands of NURBS control points was parameterized with some dozens of design variables, enabling the carrying out of efficient gradient-free optimization. The adoption of NURBS surfaces and curves enables the use of B-Rep models in standard formats, such as STEP format, to import and export shapes for further analysis in the design workflow. It also allows their use in optimization loops, by assisting for instance the imposition of geometric constraints, as demonstrated in the compressor stator in Chapter 7. An additional advantage of the proposed method is that thanks to the multi-step deformations, it acts as a smooth multi-frequency shape-morphing tool. Moreover, it provides a scalable number of shape parameters, enabling to enrich design spaces sequentially, as shown in Section 2.4.2 for a duct shape.

It is always possible to reduce the number of DoFs considerably, but parametric effectiveness needs to be considered too. In Section 2.4.1, the PE of the B-Rep-Morpher in a 2D case was compared with the standard NURBS parameterization. Even if the proposed method reduced the number of DoFs by 60%, the PE was reduced by just 22%, which is very satisfactory. As demonstrated in the compressor stator case, Chapter 7, the proposed B-Rep-Morpher enables the exploration of shape variations that constructive CAD methods, despite their indisputable robustness, might “miss”.

An existing geometric modeler for the design and shape parameterization of turbomachinery rows, namely the GMTurbo tool introduced in the PhD thesis of K. Tsiakas [8], is briefly presented in Chapter 3. The software was used to produce and optimize turbomachinery blades successfully. The compactness of such parameterization results in shape representations working with a controllable number of DoFs. Thanks to the extension implemented in this thesis, the tool can retain CAD compatibility throughout the optimization, providing the shapes in an appropriate format, such as STEP format, for subsequent stages of a design workflow. Furthermore, Section 3.3 presented software that converts blade geometries given in B-Rep formats, to GMTurbo formats, through a series of geometric computations. Geometries have been successfully imported and then optimized, such as in Chapter 7.

Chapter 4 illustrated two key components enabling the integration of GMTurbo and the B-Rep-Morpher software into automatic shape optimization loops. In detail, Section 4.1 illustrated a framework that allows for the deformation of surface meshes based on parameter changes of a reference CAD-based design model, namely GMTurbo or B-Rep-Morpher models. A key idea of this framework is to exploit the flexibility of RBF-based interpolation techniques in order to morph the NURBS parametric coordinates of the surface nodes of a volumetric simulation mesh according to displaced trimming curves. When coupled with the volumetric mesh displacement method presented in Chapter 5, this component allows for the implementation of CAD-based design optimization processes with no need for re-meshing. Another indispensable part of the gradient-based optimization is the differentiation of the parameterization: this is performed by means of FDs, which give satisfactory results in terms of time and accuracy, as illustrated in Section 4.2.

Chapter 5 illustrated a new RBF-based mesh displacement method capable of deforming CFD volumetric meshes, given the displacements of all surface mesh nodes. The proposed method places itself among a series of recent efforts to develop faster algorithms for mesh displacement based on RBF. It combines a two-step strategy with an effective preconditioner based on SPAI to accelerate the training phases in both steps. The FMM and an integer lattice are used to speed up the predictor and corrector interpolation phases, respectively.

The predictor reduces the problem size by solving an interpolation problem on a reduced dataset generated by a fast spatial decomposition method based on octree. This results in a geometric approximation of the boundaries, which is corrected by the local deformation in the second step. For instance, in the double elbow duct case, Section 5.5, the standard RBF requires the solution of a linear system approximately of rank 10^5 whereas, with the proposed method, the RBF training phase is subdivided into the solution of a dense linear system approximately of rank 10^4 and a sparse linear system approximately of rank 10^5, although with just 0.5% of non-zero entries. The SPAI preconditioner reduces the number of iterations needed by the iterative solver by more than one order of magnitude. The computation of the SPAI preconditioner is easily performed in parallel and a strategy, based on geometric considerations, to compute the non-zero structure and reduce the number of dense decompositions, by a factor 15, for the duct case, was proposed.

The performance of the FMM-based interpolation is assessed against standard RBF interpolation, resulting to order of magnitudes saving in computational cost for large CFD meshes. For example, by using the FMM for an RBF model with 1.25×10^5 source and 2×10^6 target nodes, the time needed to compute the displacements of the internal nodes is reduced by 10 times for parallel implementations yielding 10^{-7} maximum relative error or 100 times for 10^{-3}. Using the integer lattice to avoid the computation of the zero-valued RBF kernels is beneficial for both the training and interpolation phase of the corrector step. For instance, for the duct case, it was possible to cut the time needed to assemble the training matrix and perform the RBF interpolation by a factor of 100. The resulting mesh displacement tool operates regardless of the mesh type: it is fast compared to the typical time needed to perform the CFD simulations and preserves mesh quality as well as viscous layers even for large deformations.

The duct and RRD turbine stator cases, Section 5.5, were used to indicate the scalability of the software for increasing mesh sizes. For example, for the RRD turbine stator, the time needed to displace a mesh with $\sim 2.20 \times 10^6$ nodes is less than 2 min while the same displacements applied to a mesh with $\sim 2.20 \times 10^7$ nodes require ~ 12 min. Software performance is measured on a computational node with two 6-core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz processors. The performance of the mesh displacement software was also examined by varying the coarsening of the predictor step. Results indicate that the performance differs by varying the predictor size. However, for the DrivAer car case, Section 5.5.3, the time required to displace the mesh is nearly constant for a wide range of predictor training matrix sizes, and non-optimal parameters yield an increase in computational time of at most 50%. In Section 5.5.4, multi-objective evolutionary optimization of the RRD turbine stator was performed to assess the robustness of the software over large displacements. Optimal solutions were found that reduce the total pressure losses up to 27% and increase the capacity up to 88% without the need to generate new meshes even for large shape modifications.

Chapter 6 illustrated the FMM method and analyzed its performance both in terms of accuracy and computational time. An efficient data structure allows using the FMM with CFD applications involving meshes with highly non-uniform spatial mesh nodes distributions. The parallel implementation based on the task-flow paradigm allows the parallelization of the FMM; using 8 cores, the parallel speedup is close to the ideal one. The parametric studies allowed the tuning of the parameters of the method to achieve the best performance. For instance, it was found that the execution time is strongly dependent on the maximum number of particles in each octree bin and a near-optimal value of 200, for the cases under study, was determined.

Finally, in Chapter 7, the TUB stator was successfully optimized using the methods presented in Chapters 2 ,3, 4 and 5. For instance, the optimization was conducted with both EAs and the SQP method by combining them with both the B-Rep-Morpher and the GMTurbo parameterization methods, as well as their differentiation. The blade was imported in GMTurbo with the re-parameterization tool. The surface and volumetric mesh have been displaced with the surface mesh displacement method and the two steps RBF method, respectively. Moreover, the geometric constraint regarding the fixture holes has been integrated into the optimization process. Drastically different shapes have been found that improve the total pressure losses, deviation of the exit flow angle from the axial direction or both. In detail, the total pressure losses are improved up to 0.5% (absolute difference) and the deviation of the exit flow angle up to 3.75°. This showed that the coordination of the developments discussed above led to a modular shape optimization framework for computationally intensive CFD-based optimization problems with the CAD into-the-loop.

8.1 Contributions

The main contributions of this thesis are listed below:

- Developed a modular CAD-based shape optimization framework. Assessed its performance and demonstrated its effectiveness in various test cases, using different parameterizations and gradient-free and -based optimization algorithms.

- Investigated the underlying principle of GMTurbo and extended it to make it more powerful and automatic. Extensions include:
  - The possibility to build shapes with NURBS and export them in IGES or STEP format; previously, the shape was built discretely (non-CFD surface mesh) and only STL format was supported.
  - Implementation of a method to compute geometric sensitivities.
  - Implementation of a strategy allowing for the deformation of surface meshes based on parameter changes of a reference GMTurbo model. The deformation can maintain the axisymmetry and periodicity of the surface mesh.
  - Developed software that converts a blade geometry given in a B-Rep format, to a GMTurbo format.
- Developed a new parameterization method for aerodynamic shape optimization capable of morphing CAD-compatible B-Rep models by displacing user-defined handles, named B-Rep-Morpher. The method is associated with the:
  - Possibility to import and export shapes in IGES or STEP format.
  - Implementation of a method to compute geometric sensitivities.
  - Implementation of a strategy allowing for the deformation of surface meshes based on parameter changes of a reference B-Rep-Morpher model. The deformation can maintain the axisymmetry and periodicity of the surface mesh.
- Developed a new method to reduce the computational cost of RBF-based mesh displacement. The technique and the developed software relies on:
  - A predictor-corrector procedure; the predictor uses an RBF-centers coarsening method to reduce the computational cost.
  - Parallel implementation of SPAI preconditioners. No contributions were made to the theory of the SPAI. The number of decompositions needed by the SPAI was reduced, by means of an integer lattice-based approach.
  - Parallel implementation of the bbFMM. No contributions were made to the theory of the bbFMM.
  - Parallel implementation of the multi-r.h.s. BiCGStab method. No contributions were made to the theory of the BiCGStab method itself.
  - An integer lattice-based RBF interpolation method.
- Developed a method to quantify the violation or compliance with the constraint regarding the turbomachinery blades fixture holes.

8.2 Future Work Recommendations

This PhD thesis relied upon software and methods developed at the PCOpt/NTUA, which has been further enriched and integrated with new techniques and software.

Regarding the B-Rep-Morpher, a natural direction for future work is the integration of additional types of geometric constraints such as the adherence to maximal or minimal thicknesses, or relations between multiple parts, such as rigid movements of parts attached to deforming shapes.

Concerning the two-step RBF mesh displacement method, it can be extended to handle mesh displacement for fluid-structure interaction simulations. The software can be parallelized with CUDA and MPI to improve parallel performance and scalability further.

Regarding the further development of GMTurbo, the software can be extended to handle new features such as cooling holes, cooling slots, squealers, nonaxisymmetric hub and shroud, variable tip clearance, volute geometry generation, blade roots, interactive blade design, integrated mesh generation and uncertainty quantification techniques.

Concerning the TUB test case, it would be interesting to build and test in a measurement rig some of the shapes resulting from the optimizations presented in this thesis.

Finally, it is useful to investigate performances of the proposed methods in more complex design optimization scenarios, such as turbomachinery rows with tip clearance or wings with a nacelle attached.

Appendix A

Non-Uniform Rational B-Splines

Non-Uniform Rational B-Spline (NURBS) [27, 143] is a mathematical model that can be used to parameterize curves and surfaces. “Non-Uniform” refers to the parameterization of the shape, denoting the non-uniformity of the length of the curve segments constituting a whole curve. “Rational” refers to the underlying mathematical representation, which allows NURBS to represent exact conics (such as parabolic curves, circles, and ellipses). “B-splines” are parametric piecewise polynomial curves [144, §6]. NURBS curves and surfaces are useful for a number of reasons: *(a)* NURBS shapes are invariant under affine transformations [143, §7.1]; operations such as translations and rotations are applied to NURBS curves and surfaces by just applying them to their control points. *(b)* The NURBS model offers one common mathematical form for both analytical and free-form shapes. *(c)* NURBS provides the flexibility to design a large variety of shapes. *(d)* Algorithms involving NURBS are generally efficient and computationally accurate.

This appendix only gives some basics of NURBS. Computational operations on NURBS geometries is achieved through a series of algorithms that can be found in the literature. Typical examples are the interpolation or approximation of a set of points with NURBS curves, projection or inversion of a point on NURBS curve/surfaces, intersection and blending between two NURBS curves/surfaces, knot insertion and removal, degree elevation and construction of surfaces from curves. The latter includes coons, skinned, revolved and extended surfaces. These algorithms have been implemented, as found in the literature, for instance, to enable the NURBS representation of shapes into GMTurbo (Chapter 3). Apart from the NURBS construction techniques, more advanced geometry modeling algorithms are used, such as solid-solid intersection (Chapter 7). In this case, external libraries have been used, such as Open Cascade[145].

In Section 2.2.2, NURBS curves $\boldsymbol{C}(u)$ and surfaces $\boldsymbol{S}(u, v)$ have been introduced; their mathematical definitions are:

$$\boldsymbol{C}(u) = \sum_{i=0}^{I} R_i(u)\boldsymbol{P}_i \quad \text{and} \quad \boldsymbol{S}(u,v) = \sum_{i=0}^{I}\sum_{j=0}^{J} R_{i,j}(u,v)\boldsymbol{P}_{i,j}\ . \tag{A.1}$$

The rational basis functions $R_i(u)$ and $R_{i,j}(u,v)$ are defined as:

$$R_i(u) = \frac{N_{i,n}(u)w_i}{\sum_{j=0}^{I} N_{j,n}(u)w_j} \quad \text{and} \quad R_{i,j}(u,v) = \frac{N_{i,n}(u)N_{j,m}(v)w_{i,j}}{\sum_{p=0}^{I}\sum_{q=0}^{J} N_{p,n}(u)N_{q,m}(v)w_{p,q}}. \tag{A.2}$$

$N_{p,n}(u)$ are the i^{th} B-Spline basis functions of degree p and $w_i > 0$ ($w_{i,j}$) are the i^{th} ($(i,j)^{\text{th}}$) weights associated with the i^{th} ($(i,j)^{\text{th}}$) control points $\boldsymbol{P}_i$ ($\boldsymbol{P}_{i,j}$). The definition of the B-Spline basis functions requires to introduce the concept of knots.

Knot Vector The knots divide a NURBS curve into segments linked with a certain degree of continuity. Knots are organized in a knot vector that is a sequence of non-decreasing real values, the knots, $U = u_k : k = 0, \ldots, K$. NURBS surfaces require two knot vectors.

A knot vector can be clamped or unclamped. A clamped knot vector of degree p has the first and last knots with multiplicity equal to $p + 1$. If this does not hold, the knot vector is referred to as as unclamped. Unclamped knot vectors are valuable in the definition of periodic curves, namely closed curves (with coinciding first and last points) with "missing" knots, which help maintain a certain degree of continuity across the curve's begin and end. The knot span is the range of parameter values between two successive knots. Consecutive knots can have equal values (knot span of zero-length). Some coinciding knots are referred to as a knot with a certain multiplicity. The knot spans are also used to classify knot vectors. In detail, an unclamped knot vector is uniform if all knot spans have the same length. A clamped knot vector is uniform if all internal knot spans have identical lengths. In all other cases, the knot vector becomes nonuniform.

B-Spline Basis Functions The definition of the basis functions is recursive in degree p. The degree-0 basis functions $N_{i,p}(u)$ are piecewise constant: they are one over the corresponding knot span and zero everywhere else. For degrees higher than 0, the basis function $N_{i,p}$ is a linear combination of $N_{i,p-1}$ and $N_{i+1,p-1}$. The latter two functions are non-zero for p knot spans and overlapping for $p - 1$ knot spans. According to [146], the basis function $N_{i,p}(u)$ is defined as

$$N_{i,p}(u) = \begin{cases} \begin{cases} 1 & \text{if } u \in [u_i, u_{i+1}) \\ 0 & \text{elsewhere} \end{cases} & \text{if } p = 0 \\ f_{i,p}(u)N_{i,p-1}(u) + g_{i+1,p}(u)N_{i+1,p-1}(u) & \text{if } p > 0 \end{cases} ; \tag{A.3}$$

$f_{i,p}(u)$ increases linearly from zero to one on the span where $N_{i,p-1}(u)$ is non-zero, and $g_{i+1,p}(u)$ decreases linearly from one to zero on the span where $N_{i+1,p-1}(u)$ is non-zero so that $f_{i,p}(u)$ and $g_{i,p}(u)$ are positive when the corresponding lower order basis functions are non-zero. In detail, $f_{i,p}(u)$ and $g_{i,p}(u)$ are defined as

$$f_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} \quad \text{and} \quad g_{i,p}(u) = 1 - f_{i,p}(u) = \frac{u_{i+p} - u}{u_{i+p} - u_i} . \tag{A.4}$$

Whenever equation A.4 results to the quotient $0/0$ this is defined to be 0 [27, §2.2].

[27, §2] reports many important properties of basis functions. Some are as follows: *(a)* the sum of the basis functions for any value of the parameter u is equal to 1 (partition of unity); *(b)* the basis functions are non-negative in the whole real domain; *(c)* $p + 1$ basis function are non-zero in any knot-span with positive length; *(d)* all derivatives of a basis function exist in the interior of a knot span; *(e)* at a knot with multiplicity s, the basis function is $p - s$ times differentiable.

Figure A.1 shows the degree-0, -1 and -2 basis functions for a specific knot vector; a curve based on such a knot-vector is illustrated in Figure A.2. The knot span $[0.15, 0.25)$ is shorter than the others. On that knot span, the peak in the quadratic basis function is more distinct, reaching almost one. On the contrary, the adjacent basis functions decrease to zero more quickly. In the geometric interpretation, this means that a curve based on such a knot vector approaches the corresponding control point closely: see the 3rd control point in Figure A.2. In the case of a double knot, the length of the knot span becomes zero, and the peak in the quadratic basis function reaches one, as in the case of the double knot at 0.75, corresponding to the 6th control point of the curve. The quadratic basis function is no longer differentiable at that point, and a degree-2 curve will have a sharp corner if the neighbor control points are not collinear. Finally, periodic uniform knot vectors yield periodic uniform basis functions, namely each basis function is a translate of the others.

NURBS Curve and Surface Properties A NURBS curve is defined by its order, a set of weighted control points, and a knot vector. The order of a NURBS curve determines

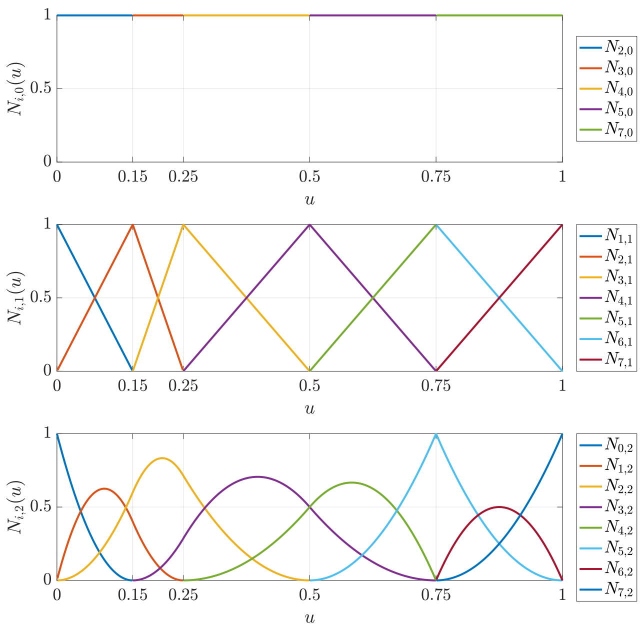


Figure A.1: Non-zero NURBS basis functions of 0th (top), 1st (middle) and 2nd (bottom) degree defined on the knot vector $[0, 0, 0, 0.150.25, 0.50, 0.75, 0.75, 1, 1, 1]$

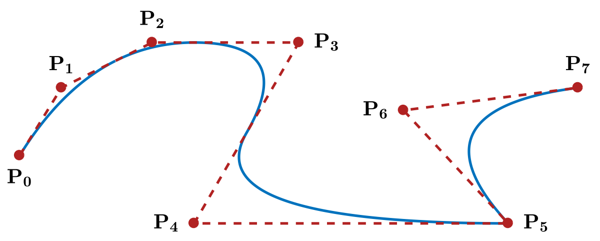


Figure A.2: A NURBS quadratic curve (blue) with its control polygon (red).

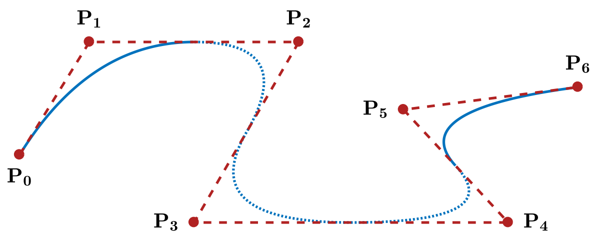


Figure A.3: A NURBS quadratic curve (blue) with its control polygon (red). The curve section affected by the displacement of the control point $\boldsymbol{P}_3$ is dotted.

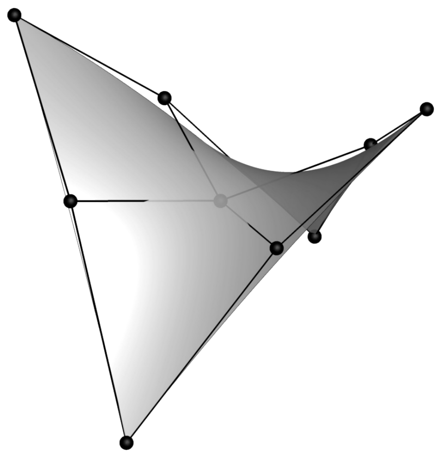


Figure A.4: NURBS quadratic surface. A NURBS surface (gray and translucent) with its control polygon (black).

the number of successive control points that influence each point on the curve. The curve is modeled mathematically by a piecewise polynomial of degree equal to the order of the curve minus one. In practice, cubic (fourth-order) curves are the ones most commonly used; curves of orders higher than 7 are numerically expensive to evaluate, due to the recursive procedure to compute the basis functions. Each control point is assigned a weight: the weight defines how much does a control point "attract" the curve and is essential to parameterize exact conics such as parabolic curves, circles, and ellipses.

NURBS curves have the following useful properties: *(a)* For clamped knot vectors, $\boldsymbol{C}(u)$ interpolates $\boldsymbol{P}_0$ and $\boldsymbol{P}_I$, that is $\boldsymbol{C}(u_0) = \boldsymbol{P}_0$ and $\boldsymbol{C}(u_K) = \boldsymbol{P}_I$. *(b)* $\boldsymbol{C}(u)$ is a piecewise polynomial curve, formed by polynomials of degree p. *(c)* The number of knots K, number of control points N and the degree p are connected through K = N + p + 1;. *(d)* Any affine transformation can be applied to $\boldsymbol{C}(u)$ by applying it to its control points and weights. *(e)* Changes to the position of a control point $\boldsymbol{P}_i$ affect only the section of the curve on the interval $u \in [u_i, u_{i+p+1})$; see Figure A.3. *(f)* $\boldsymbol{C}(u)$ is infinitely differentiable on the interior of knot spans and is $p - s$ times differentiable at a knot of multiplicity s.

NURBS surfaces are tensor product surfaces. The latter are based on bidirectional curve scheme; this means that they employ bivariate basis functions which are the product of univariate basis functions specified on the two distinct parametric directions, as inferred from equation A.1. NURBS surfaces are defined through an $(I + 1) \times (J + 1)$ grid of control points $\boldsymbol{P}_{i,j}$ and two-knot vectors and orders, one for each parametric direction. The main properties of NURBS curves are also applicable to NURBS surfaces. Figures A.4 and 2.2 illustrate examples of NURBS surfaces.

Appendix B

Conformal Mapping

The GMTurbo parameterization described in Chapter 3 relies on conformal mapping to produce airfoils in 2D that correspond to airfoils lying on surfaces of revolution in 3D.

In mathematics, a conformal map is a function that preserves orientation and angles locally. Two metrics g and h are said to be conformally equivalent if there exists a strictly positive function c so that $g = ch$. Then, the function c is called the conformal factor. The conformal (or angle preserving) mapping, which is of interest in the context of the blade parameterization method, is that of a surface of revolution on the $m\theta$-plane. The mapping of the $m\theta$-plane in the 3D Cartesian space is illustrated in Figure B.1.

Let the generatrix of the surface of revolution be defined parametrically in the zr plane as $\mathbf{G}(u) = (z(u), r(u))$. Then, in the Cartesian space the surface of revolution is represented by $\mathbf{S}(u,\theta) = (r(u)\cos\theta, r(u)\sin\theta, z(u))$. A mapping Φ of $\mathbf{S}$ in the $m\theta$-space is defined as:

$$\begin{aligned} \Phi \; &: \; (r(u)\cos\theta, r(u)\sin\theta, z(u)) \mapsto (m(u), \theta) \\ \text{with} \quad m(u) &= \int_0^u \frac{\sqrt{\left(\frac{\mathrm{d}r(t)}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}z(t)}{\mathrm{d}t}\right)^2}}{r(t)} \mathrm{d}t \end{aligned} \tag{B.1}$$

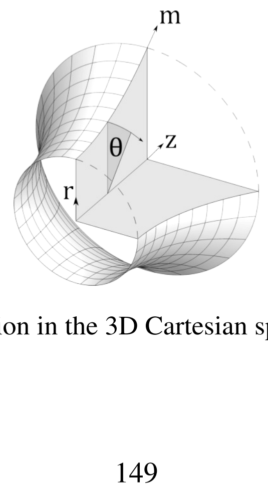


Figure B.1: A surface of revolution in the 3D Cartesian space and the representation of the conformal $m\theta$ 2D space.

Let $\boldsymbol{s}(u,\theta) = (r(u)\cos\theta, r(u)\sin\theta, z(u))$ denote a point on the surface and $\boldsymbol{p}(u,\theta) = (m(u),\theta)$ denote the same point mapped onto the (m,θ) plane. The metrics of the first fundamental form[1] of $\mathbf{s}$ are given as

$$\begin{aligned}
E_s &= \frac{\partial \mathbf{s}}{\partial u}\cdot\frac{\partial \mathbf{s}}{\partial u} = \left[\frac{\partial r(u)}{\partial u}\right]^2 + \left[\frac{\partial z(u)}{\partial u}\right]^2 \\
F_s &= \frac{\partial \mathbf{s}}{\partial u}\cdot\frac{\partial \mathbf{s}}{\partial \theta} = -r(u)\frac{\partial r(u)}{\partial u}\cos\theta\sin\theta + r(u)\frac{\partial r(u)}{\partial u}\cos\theta\sin\theta = 0 \\
G_s &= \frac{\partial \mathbf{s}}{\partial \theta}\cdot\frac{\partial \mathbf{s}}{\partial \theta} = r^2(u)\sin^2\theta + r^2(u)\cos^2\theta = r^2(u)
\end{aligned} \tag{B.2}$$

Likewise, the metrics of the first fundamental form of $\mathbf{p}$ are given as

$$\begin{aligned}
E_s &= \frac{\partial \mathbf{p}}{\partial u}\cdot\frac{\partial \mathbf{p}}{\partial u} = \frac{\left[\frac{\partial r(u)}{\partial u}\right]^2 + \left[\frac{\partial z(u)}{\partial u}\right]^2}{r^2(u)} \\
F_s &= \frac{\partial \mathbf{p}}{\partial u}\cdot\frac{\partial \mathbf{p}}{\partial \theta} = 0 \\
G_s &= \frac{\partial \mathbf{p}}{\partial \theta}\cdot\frac{\partial \mathbf{p}}{\partial \theta} = 1
\end{aligned} \tag{B.3}$$

From Equations B.2 and B.3, it can be shown that

$$E_s = \frac{1}{r^2(u)}E_p\,, \quad F_s = \frac{1}{r^2(u)}F_p\,, \quad G_s = \frac{1}{r^2(u)}G_p \tag{B.4}$$

Equation B.4 proves that the mapping defined by Equation B.1 is conformal with the conformal factor $c = \frac{1}{r^2(u)}$. The length is not preserved by the mapping. However, a relation exists [147], as follows. An infinitesimally small length $\mathrm{d}\ell_s$ on the 3D surface of revolution is expressed as

$$\mathrm{d}\ell_s = \sqrt{E_s\mathrm{d}u^2 + F_s\mathrm{d}u\mathrm{d}\theta + G_s\mathrm{d}\theta^2} \tag{B.5}$$

while the same infinitesimal length $\mathrm{d}\ell_p$ on the 2D conformal space of the surface is expressed as

$$\mathrm{d}\ell_p = \sqrt{E_p\mathrm{d}u^2 + F_p\mathrm{d}u\mathrm{d}\theta + G_p\mathrm{d}\theta^2} \tag{B.6}$$

Combining Equations B.5, B.6 and B.4, it is easily proven that

$$\frac{\mathrm{d}\ell_s}{\mathrm{d}\ell_p} = \frac{1}{r} \tag{B.7}$$

[1]In differential geometry, the first fundamental form of a surface is the inner product on the tangent vectors of the surface in three-dimensional Euclidean space [147, §16.6].

Appendix C

The Newton's and SQP Methods

In this appendix, some of the gradient-based numerical optimization methods for non-linear unconstrained and constrained problems that are used in the thesis are reviewed.

An optimization problem is formulated by first identifying one or more objectives, namely quantitative measures of the performance of the system under study, such as the pressure losses of a compressor row. The objective functions depend on the design variables or DoFs. The goal of numerical optimization is to find the values of the variables that optimize the objectives. Often the optimization problem is constrained, in some way: for instance, by limiting the values of the DoFs or by some other functions that must be in specific ranges. Formally, non-linear problems with one objective can be written as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_0(\boldsymbol{x}) \\ \text{subject to} \quad & f_i(\boldsymbol{x}) \leq 0, \quad i \in \mathcal{I} \\ & f_i(\boldsymbol{x}) = 0, \quad i \in \mathcal{E} \\ & \boldsymbol{x} \in \mathcal{X} \end{aligned} \tag{C.1}$$

where

- $\boldsymbol{x} = (x_1, \dots, x_N)^\mathsf{T}$ is the vector of DoFs,
- $f_0(\boldsymbol{x})$ is the scalar objective function (to be minimized),
- $f_i(\boldsymbol{x}), i \in \mathcal{E}$ are the equality constraints,
- $f_i(\boldsymbol{x}), i \in \mathcal{I}$ are the inequality constraints,
- $\mathcal{X} = \{x : x_j^{\min} < x_j < x_j^{\max},\ j \in [1, \dots, N]\}$ are the variable bounds,
- $\mathcal{I}$ and $\mathcal{E}$ are sets of indeces.

The class of gradient-based optimization methods includes, among a vast realm, steepest descent [148], Newton and quasi-Newton [149, 150] and SQP [151]. These methods are suitable for smooth (twice differentiable)-non-linear objective functions and constraints, and may find the minimum of the basin of attraction of the user-supplied starting point.

Line Search Line search are methods used as part of larger optimization algorithms, such as Newton and SQP. At each iteration k of the main algorithm, the line-search method look for for an improved point $\boldsymbol{x}_{k+1}$ along the line containing the current point $\boldsymbol{x}_k$ and parallel to the search direction $\boldsymbol{p}_k$, which is a vector determined by the main algorithm. Namely, the method finds the next iterate $\boldsymbol{x}_{k+1}$ by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \eta \boldsymbol{p}_k \quad . \tag{C.2}$$

The distance to move along $\boldsymbol{p}_k$ is determined by the step-length η that is found by approximately solving:

$$\min_{\eta>0} f_0\left(\boldsymbol{x}_k + \eta \boldsymbol{p}_k\right) \quad . \tag{C.3}$$

Line search algorithms generate a limited number of trial step-lengths until they establish one that approximates the minimum of equation C.3 so that $f_0\left(\boldsymbol{x}_k + \eta \boldsymbol{p}_k\right) < f_0\left(\boldsymbol{x}_k\right)$. At each new iterate of the outer optimization method, computing a new search direction, the line search method compute a new step-length. The Wolfe conditions are prevalently used inexact line search conditions [12, §3.1].

C.1 Newton's Method and Quasi-Newton Methods

Newton's Method and Quasi-Newton Methods [149, 150] are used for unconstrained Non-Linear Programming such as the geometric problem is Section 2.3.2.

Newton's Method Newton's methods use a second-order Taylor series expansion of the function about the point $f(\boldsymbol{x}_k + \boldsymbol{p})$:

$$f_0(\boldsymbol{x}_k + \boldsymbol{p}) \approx f_0(\boldsymbol{x}_k) + \boldsymbol{p}^{\mathsf{T}} \nabla f_0(\boldsymbol{x}_k) + \frac{1}{2} \boldsymbol{p}^{\mathsf{T}} \nabla^2 f_0(\boldsymbol{x}_k) \boldsymbol{p} \quad . \tag{C.4}$$

Assuming that $\nabla^2 f_0(\boldsymbol{x}_k)$ is positive definite, the Newton search direction is vector $\boldsymbol{p}$ that minimizes the right-hand-side of equation C.4, that is:

$$\boldsymbol{p}_k^{\mathsf{N}} = -\left(\nabla^2 f_0(\boldsymbol{x}_k)\right)^{-1} \nabla f_0(\boldsymbol{x}_k) \quad . \tag{C.5}$$

The Newton direction $\boldsymbol{p}_k^{\mathrm{N}}$ can be used when $\boldsymbol{\nabla}^2 f_0(\boldsymbol{x}_k)$ is positive definite; otherwise, $\left(\boldsymbol{\nabla}^2 f_0(\boldsymbol{x}_k)\right)^{-1}$ in equation C.5 would be undefined. In such a case, methods that modify the definition of $\boldsymbol{p}_k$ while retaining the second-order information, exist [12, §3.4]. Methods that use the Newton direction have a fast rate of convergence, typically quadratic. The main weakness of the Newton method is the need to compute the Hessian of the objective function, which usually is an expensive process.

Quasi-Newton Methods Quasi-Newton search directions provide an attractive alternative to Newton's method since they do not require computation of the Hessian and still attain a super-linear convergence rate. In place of the exact Hessian$\boldsymbol{\nabla}^2 f_0(\boldsymbol{x}_k)$, they use an approximation $\boldsymbol{B}_k$, which is updated in each iteration k to consider the additional information obtained: the updates rely on the fact that changes in the gradient $\boldsymbol{\nabla} f_0$ yield information about the second derivative of f_0 along the search direction. Two of the most widely used formulas for updating the Hessian approximation $\boldsymbol{B}_k$ are the Symmetric-Rank-one (SR1) formula [152] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula [153]. The latter is defined by

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{B}_k^T}{\boldsymbol{s}_k^T \boldsymbol{B}_k \boldsymbol{s}_k} \quad , \tag{C.6}$$

where $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y}_k = \boldsymbol{\nabla} f_{k+1} - \boldsymbol{\nabla} f_k$. The derivation of such equations is illustrated in [12, §6.1]. The quasi-Newton search direction is, then, defined as

$$\boldsymbol{p}_k^{QN} = -\left(\boldsymbol{B}_k\right)^{-1} \boldsymbol{\nabla} f_0(\boldsymbol{x}_k) \quad . \tag{C.7}$$

The BFGS formula generates positive-definite approximations whenever the initial approximation $\boldsymbol{B}_0$ is positive-definite and $\boldsymbol{s}_k^{\mathsf{T}} \boldsymbol{y}_k > 0$ [12, §6.1]. The initial Hessian approximation $\boldsymbol{B}_0$ can be chosen to be the identity matrix or a multiple of it, which reflects the DoFs scaling. A strategy for solving optimization problems with the quasi-Newton method is illustrated in Algorithm C.1.

C.2 Sequential Quadratic Programming

Optimization theory for smooth problems of the form of equation C.1 is based on the Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ function that combines the objective function $f_0(\boldsymbol{x})$ and the constraints

Algorithm C.1: BFGS-based quasi-Newton method

```
Given $\boldsymbol{x}_0$, $\boldsymbol{B}_0$ and $\epsilon_g > 0$
$k = 0$
repeat
    Compute the gradient $\nabla f(x_k)$
    Compute the search direction $\boldsymbol{p}_k$ by equation C.7
    Compute the step-length $\eta$ by performing a line search
    Compute $\boldsymbol{x}_{k+1}$ by equation C.2
    Compute $\boldsymbol{B}_{k+1}$ by equation C.6
    $k \leftarrow k + 1$
until $\|\nabla f(x_k)\| \geq \epsilon_g$
```

$f_i(\boldsymbol{x}),\ i \in \mathcal{I} \cup \mathcal{E}$. Its definition is:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f_0(\boldsymbol{x}) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i f_i(\boldsymbol{x}) \tag{C.8}$$

where $\boldsymbol{\lambda} = \{\lambda_i,\ i \in \mathcal{I} \cup \mathcal{E}\}$ is the vector of Lagrangian multipliers.Such a problem is solved by seeking for stationary points of the Lagrangian which respect certain optimality conditions, known as the Karush-Kuhn-Tucker (KKT) conditions [12, §12.4], and assumptions on the constraints [12, §12.6]. The solution of the KKT equations constitutes the basis of several non-linear programming algorithms, such as the SQP method.

SQP methods are state of the art in non-linear programming methods. For instance, [154] has implemented and tested a version that (according to the authors) outperforms every other tested method in terms of efficiency, accuracy, and percentage of successful solutions, over a considerable number of test problems. Based on [155–158], the SQP method allows mimicking Newton's method for constrained optimization. At each iteration, an approximation to the Hessian of the Lagrangian function is made using a quasi-Newton updating method; this is then used to generate a QP subproblem whose solution is used to form a search direction for a line search procedure.

Given the problem description in equation C.1, the main idea is the formulation of a QP subproblem based on a quadratic approximation of the Lagrangian function of equation C.8 with linearized constraints. Namely, the problem at an iterate $(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$ is modeled as:

$$\begin{aligned} \min_{\boldsymbol{p}} \quad & \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) + \boldsymbol{p}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) + \frac{1}{2} \boldsymbol{p}^{\mathsf{T}} \nabla^2_{\boldsymbol{xx}} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \boldsymbol{p} & \\ \text{subject to} \quad & f_i(\boldsymbol{x}_k) + \boldsymbol{p}^{\mathsf{T}} \nabla f_i(\boldsymbol{x}_k) \leq \boldsymbol{0}, & i \in \mathcal{I} \\ & f_i(\boldsymbol{x}_k) + \boldsymbol{p}^{\mathsf{T}} \nabla f_i(\boldsymbol{x}_k) = \boldsymbol{0}, & i \in \mathcal{E} \end{aligned} \tag{C.9}$$

where $\nabla^2_{xx}\mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$ is the Hessian of the Lagrangian function; bounds constraints, if any, are handled as inequality constraints.

Under certain conditions described in [12, §18.1], the quadratic problem in equation C.9 can be solved by QP programming [12, §16.1] leading to the solution $\boldsymbol{p}_k$ and updated Lagrangian multipliers. For each SQP iteration, $\boldsymbol{p}_k$ is used to obtain the new iterate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \eta \boldsymbol{p}_k$, where η is determined by an appropriate line search procedure. Many SQP methods approximate the Hessian of the Lagrangian through a quasi-Newton approach, such as the BFGS update formula (equation C.6) with $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y}_k = \nabla\mathcal{L}(\boldsymbol{x}_{k+1}, \lambda_{k+1}) - \nabla\mathcal{L}(\boldsymbol{x}_k, \lambda_{k+1})$. [157] recommended keeping the Hessian positive definite even though it might be indefinite at the solution point. To ensure the positive definitiveness of the approximated Hessian, the damped BFGS updating for SQP has been devised: in equation C.6, $\boldsymbol{y}_k$ is interchanged with $\boldsymbol{r}_k$ which is defined as

$$\boldsymbol{r}_k = \theta_k \boldsymbol{y}_k + (1 - \theta_k)\boldsymbol{B}_k \boldsymbol{s}_k \tag{C.10}$$

where

$$\theta_k = \begin{cases} 1 & \text{if } \boldsymbol{s}_k^\top \boldsymbol{y} \geq 0.2\boldsymbol{s}_k^\top \boldsymbol{B}_k \boldsymbol{s}_k \\ \left(0.8\boldsymbol{s}_k^\top \boldsymbol{B}_k \boldsymbol{s}_k\right) / \left(\boldsymbol{s}_k^\top \boldsymbol{B}_k \boldsymbol{s}_k - \boldsymbol{s}_k^\top \boldsymbol{y}\right) & \text{otherwise} \end{cases} . \tag{C.11}$$

This modified BFGS update method interpolates the current $\boldsymbol{B}_k$ and the one corresponding to the BFGS. The choice of θ_k ensures that the new approximation is close enough to the current approximation to guarantee positive definitiveness.

Apart from using a different quasi-Newton update, SQP algorithms also need adjustments to the line search strategy to ensure convergence from remote starting points. It is common to use a merit function, to control the step-size in the line search. [156] suggested the following possibility for such a function:

$$\phi(\boldsymbol{x}, \mu) = f_0(\boldsymbol{x}) + \sum_{i \in \mathcal{E}} \mu_i |f_i(\boldsymbol{x})| + \sum_{i \in \mathcal{I}} \mu_i \max(0, f_i(\boldsymbol{x})) \quad . \tag{C.12}$$

μ_i are penalty parameters which [157] updated, at a iteration k, with the following strategy:

$$\mu_i = (\mu_{k+1})_i = \max\left(\lambda_i, \frac{(\mu_k)_i + \lambda_i}{2}\right), \ (\mu_0)_i = \frac{\|\nabla f_0(\boldsymbol{x}_0)\|}{\|\nabla f_i(\boldsymbol{x}_0)\|}, \quad i \in \mathcal{I} \cup \mathcal{E}. \tag{C.13}$$

This allows positive contribution from inactive constraints in the QP solution at an iteration k but were recently active. An optimization procedure based on SQP in sketched in Algorithm C.2.

Algorithm C.2: SQP Method

```
Given $(\boldsymbol{x}_0, \boldsymbol{\lambda}_0)$ and $\boldsymbol{B}_0$
$k = 0$
**repeat**
    Compute $\boldsymbol{p}_k$ and $\boldsymbol{\lambda}_{k+1}$ by solving the QP problem in equation C.9 with $\boldsymbol{B}_k$ in place of
      $\nabla^2_{\boldsymbol{xx}} \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$
    Compute $(\mu_k)_i,\ i \in \mathcal{I} \cup \mathcal{E}$ by equation C.12 such that $\boldsymbol{p}_k$ is a descent direction for $\phi$ at
      $\boldsymbol{x}_k$
    Compute the step-length $\eta$ by performing a line search on the merit function $\phi$
    Compute $\boldsymbol{x}_{k+1}$ by equation C.2
    Compute $\boldsymbol{B}_{k+1}$ by equation C.6 with $\boldsymbol{r}_k$ in place of $\boldsymbol{y}_k$
    $k \leftarrow k + 1$
**until** *optimality conditions satisfied.*
```

Appendix D

Evolutionary Algorithms and the EASY Software

Gradient-free optimization methods include, among many others, simulated annealing [159], Particle Swarm Optimization (PSO) [160], surrogate optimization [161], Generalized Pattern Search (GPS) [162] and Evolutionary Algorithms (EAs) [163]. This appendix focuses on the latter and the Evolutionary Algorithm System (EASY) software [164–167] developed by the PCOpt/NTUA. EASY supports MAEAs with several metamodel types and other methods to enhance EAs, such as distributed, asynchronous and hierarchical search.

D.1 Evolutionary Algorithms

Extensive descriptions of EAs can be found in [163, 168, 169]. The theory of evolution of species inspired EAs; the algorithms mimick natural processes of parent selection, crossover (the combination of parents) and mutation (random changes) among a population of "individuals". Each individual is a candidate solution of the optimization problem, whose genotype is the set of DoFs or design variables, and whose phenotype is the objective function and constraint values. Such evolution operators have a certain degree of randomness so that EAs are classified as stochastic methods. Over successive generations, the population "evolves" toward one with expectedly better-performing individuals. EAs methods address optimization problems where the objective or constraint functions are smooth, non-smooth and non-differentiable, and the DoFs are discrete or continuous; the user does not need to provide any initial point. The objective and constraints are treated as black-box functions. Extension to multi-objective optimization computing Pareto fronts of non-dominated solutions is also in use [170].

However, EAs have a significant disadvantage, which is the large number of candidate solutions to be evaluated in order to reach the optimal solution. Such a disadvantage is detrimental in case the candidate solution evaluation is time-consuming, as in the case of CFD-based simulations: surrogate optimization is best suited to time-consuming objective functions. A surrogate or metamodel is a function that approximates the objective function. The metamodel is useful because it takes practically zero time to evaluate a new candidate solution. So, for instance, to search for a minimum of the objective function, an optimization can be run on the metamodel to find an approximation to the minimizer of the objective function. Response surfaces, Radial Basis Function [171] or other models are used as metamodels, trained on a set of evaluated candidate solutions. EAs assisted with surrogate evaluation models are referred to as Metamodel-Assisted EAs (MAEAs). In MAEAs, surrogate metamodels management is important [172]. Surrogate models can be trained either before the EA optimization starts, which is referred to as an MAEA with off-line trained metamodels or during the evolution by exploiting newly computed information, which is referred to as an MAEA with on-line trained metamodels [173]. The former are usually trained by a Design of Experiments (DoE) [174]. Moreover, surrogate optimization must consider the range of validity of the underlying metamodel: based on this criterion, metamodels are distinct between global and local ones [172].

D.2 Evolutionary Algorithms in EASY

The PCOpt/LTT has been developing EA strategies for the last two decades, and it has implemented them in software named EASY [175]. The algorithm used by the EASY software is the (μ, λ)-EA: it handles three different populations in each generation, namely the offspring population with λ individuals, the parent population with μ individuals and the elite population with ϵ individuals. Constrained optimization is handled using a penalty function [164, §2.3.4]. The outline of the algorithm is as follows.

Initialization Initialize the offspring population at random, by considering the user-defined lower and upper bounds for each design variable. Optionally, the user can inject individuals in the initial population.

Evaluation Compute the objectives and constraints values of each individual of the current offspring population. The evaluated individuals are stored in a database to avoid repeated evaluations.

Fitness Assignment Compute the fitness values of each individual of the current three populations. For single-objective problems, the fitness coincides with the objective. For multi-objective problems, the fitness is computed with Pareto dominance techniques [164, §2.3.3].

Elite Selection Chose the elite population among the non-dominated individuals of the current offspring population and the previous elite population. If the number exceeds the user-defined max. number ϵ, trimming is necessary. This operator supports the monotonic convergence of the algorithm over the generations by retaining the best solutions.

Elitism Operator The members of the elite population substitute some members in the current offspring population.

Parent Selection Form the parent population by selecting μ members, based on their fitness, from the union of the previous parent population and the current offspring population. Lower fitness leads to greater selection probability and an individual can be selected more than once as a parent, in which case it contributes to the "genes" of more than one child. Various strategies exist to select the parents, see [164, §2.3.2].

Crossover Operator Create a new offspring population applying the crossover operator to the parent population. Parents are selected with a certain probability, again based on their fitness, and combined in different ways to produce a new offspring. This operator enables the algorithm to extract the best "genes" from various parents and recombine them into potentially fitter children. Many strategies exist, see [164, §2.3.2].

Mutation Operator Members of the offspring population undergo mutations with a small user-defined probability. This operator helps to maintain the diversity in the population and explore the design space by overcoming the stagnation of the evolution.

Termination Criteria The algorithm stops when one of the user-supplied stopping criteria is met, such as the number of evaluations. Otherwise, the algorithm returns to the evaluation step.

In EASY, the process can be assisted by metamodels as follows. In [176], the PCOpt/LTT proposed a MAEA strategy based on the low-cost pre-evaluation method: metamodels are trained and used in each generation, and only a few best candidates, according to the metamodel-based evaluations, are re-evaluated with CFD simulations during a pre-evaluation phase. This low-cost pre-evaluation phase starts once a predefined number of individuals is available in the database, to train the metamodel; up to that point, the standard EA is used. Information computed with the problem-specific model is continuously added to the on-line metamodels. In multi-objective optimization, different metamodels are trained to predict different objective functions. Compared to the standard algorithm above, MAEAs substitute the evaluation phase with the following one.

Evaluation Once the database contains a sufficient number of entries, the evaluation starts according to the following steps. Otherwise, the standard EA evaluation phase is used.

Inexact Pre-Evaluation Train the local metamodel(s) using the database entries to compute the objective and constraint function values of each individual of the current offspring population.

Selection for Individuals to be Re-Evaluated Select the most promising individuals, λ_e, based on the objective function values approximated by the metamodels. Usually, 10% of the current offspring population is selected.

Exact Evaluation Re-evaluate exactly and store in the database the λ_e individuals.

A comprehensive description of the methods implemented in EASY is available in the D. Kapsoulis' PhD Thesis [164] (same group).

Appendix E

The PUMA CFD Flow and Adjoint Solver

For CFD computations and their adjoint counterparts, this thesis relies upon CFD software developed by the PCOpt/NTUA which is named PUMA (Parallel-Unstructured-Multi-Row-Adjoint) [8, 164, 177–181]. The flow models used in this thesis consist of the Reynolds-Averaged Navier-Stokes (RANS) equations for either compressible or incompressible flows with the Spalart-Allmaras turbulence model [182]. The PUMA solver is implemented on NVIDIA Graphics Processing Units (GPUs) using CUDA [183], taking advantage of the parallel speed-up they offer: up to 45x compared to the CPU implementation of the same code [184]. Moreover, PUMA is capable of running on many GPUs on the same or different computational nodes, using MPI. This appendix illustrates the basics of the governing equations for compressible flows.

E.1 RANS Equations for Compressible Flows

The equations are expressed w.r.t. a (relative) non–inertial frame of reference rotating at a constant speed (occasionally zero). By defining a coordinate system $O(x_1\ x_2\ x_3)$ which rotates at a constant rotation speed ω_m ($m = 1, 2, 3$), then the RANS equations for compressible flows are expressed as

$$R_n = \frac{\partial f_{nk}^{\text{inv}}}{\partial x_k} - \frac{\partial f_{nk}^{\text{vis}}}{\partial x_k} + S_n = 0\ . \tag{E.1}$$

R_n is the residual of the mean-flow Navier-Stokes (NS) equations. Inviscid fluxes f_{nk}^{inv}, viscous fluxes f_{nk}^{vis} and source terms S_n (corresponding to the Coriolis force) are defined as:

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

v_m^A $(m = 1, 2, 3)$ are the velocity components w.r.t. the absolute/inertial frame of reference, ρ the fluid density, p the static pressure, v_m^R the relative velocity components, q_k the heat flux and h_t the total enthalpy. τ_{mk} are the components of the viscous stress tensor for Newtonian fluids that are function of the bulk viscosity μ and turbulent viscosity μ_t. The relative velocity components v_m^R are linked to the absolute ones v_m^A by the relation $v_m^A = v_m^R + v_m^F$, 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. δ_{mk} and $\varepsilon_{m\ell k}$ are the Kronecker and Levi-Civita symbols, respectively. The turbulent viscosity μ_t is computed by means of the one-equation Spalart-Allmaras turbulence model [182]. Moreover, $U_n = \left[\rho \;\; \rho v_1^A \;\; \rho v_2^A \;\; \rho v_3^A \;\; \rho E\right]$ are the conservative mean-flow variables.

To fully define the flow problem, the mean-flow and Spalart-Allmaras equations must be solved for a particular set of boundary conditions. Some of the conditions used in this thesis are as follows.

Symmetry Normal gradients of flow variables and the normal velocity across the boundary are set to 0.

Periodic Flow variables are equal along periodically paired points; in the case of peripheral periodicity, vector and tensor quantities are properly rotated.

Wall For no-slip walls, the relative velocity is set equal to the wall boundary velocity and the turbulence variable to 0, for the Spalart-Allmaras model.

Inlet For subsonic inlet boundaries, the total pressure, total temperature, inlet absolute velocity direction and inlet turbulence level are specified. The static pressure, absolute velocity magnitude or the local Mach number are extrapolated from the flow domain.

Outlet At subsonic outlets, the outlet static pressure distribution, the outlet mean static pressure or the outlet mass flow rate is specified.

Far-Field Depending on the local velocity, if the flow enters the domain, the boundary is locally considered as an inlet; otherwise, it is locally considered as an outlet.

The inviscid fluxes are computed using the Roe's approximate Riemann solver [185]. The spatial discretization of the inviscid fluxes is second-order accurate, with appropriate flux limiters [186]. Viscous fluxes are computed using an edge-based central difference scheme. The discretized equations are solved using a time-marching algorithm and a point-implicit Jacobi scheme. More details on the RANS equations and their discretization are provided in [8, §2].

E.2 The Continuous Adjoint Method for Aerodynamic Shape Optimization

As illustrated in Section 4.2, in gradient-based shape optimization, it is necessary to compute the gradient of an objective function F w.r.t. a set of design variables $b_i,\ i = 1, \ldots, N$. This section briefly illustrates how to obtain this gradient using the continuous adjoint method for compressible flows. Herein, the partial derivative $\frac{\partial \Phi}{\partial b_i}$ is associated with the change in a field quantity Φ induced solely by changes in the flow variables caused by changes in b_i, while the total derivative $\frac{\delta \Phi}{\delta b_i}$ takes into account also the change in Φ due to the change in position x_k of mesh nodes.

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

$$F = \int_{S_{\mathrm{Obj}}} F_S \mathrm{d}S \ . \tag{E.3}$$

The adjoint formulation starts by defining the augmented function

$$F_{\mathrm{aug}} = F + \int_{\Omega} \Psi_n R_n \mathrm{d}\Omega + \int_{\Omega} \tilde{\nu}_a R_{\tilde{\mu}} \mathrm{d}\Omega, \quad n = 1, \ldots, 5 \ , \tag{E.4}$$

where Ψ_n are the mean-flow adjoint variables, $\tilde{\nu}_a$ the adjoint turbulence model variable and $R_{\tilde{\mu}}$ the residual of the turbulence model equation. Upon convergence of the flow equations, the residuals tend to zero and, consequently, F_{aug} tends to F, so that the required sensitivities can be computed from $\frac{\delta F_{\mathrm{aug}}}{\delta b_i}$. By differentiating equation E.4 and applying the Leibniz theorem:

$$\frac{\delta F_{\mathrm{aug}}}{\delta b_i} = \frac{\delta F}{\delta b_i} + \int_{\Omega} \Psi_n \frac{\partial R_n}{\partial b_i} \mathrm{d}\Omega + \int_{\partial\Omega} \Psi_n R_n \frac{\delta x_k}{\delta b_i} \mathsf{n}_k \mathrm{d}S + \int_{\Omega} \tilde{\nu}_a \frac{\partial R_{\tilde{\mu}}}{\partial b_i} \mathrm{d}\Omega + \int_{\partial\Omega} \tilde{\nu}_a R_{\tilde{\mu}} \frac{\delta x_k}{\delta b_i} \mathsf{n}_k \mathrm{d}S \ . \tag{E.5}$$

The term $\frac{\delta F}{\delta b_i}$ is developed based on the definition of the specific objective function F. During the mathematical development of $\frac{\delta F_{\text{aug}}}{\delta b_i}$, volume integrals containing variations of flow quantities w.r.t. the design variables arise. Because variations of flow quantities w.r.t. the design variables are associated with high computational cost, the integrals containing them are set equal to zero and by doing so, a new set of Partial Differential Equations (PDEs), the so-called field adjoint equations, arise:

$$-A_{nmk}\frac{\partial \Psi_n}{\partial x_k} - \mathcal{K}_m + \mathcal{K}_m^{\text{SA}} + \mathsf{S}_m^{\text{adj}} = 0\,. \tag{E.6}$$

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, $\mathsf{S}_m^{\text{adj}}$ is the adjoint to the Coriolis force and $A_{nmk} = \frac{\partial f_{nk}^{inv}}{\partial U_m}$ is the inviscid flux Jacobian of the flow equations. An equation is similarly developed for the Spalart-Allmaras turbulence model [8, §3.1.3].

After having treated all volume integrals arising from the differentiation of F_{aug}, the surface integrals remain. The dependency of the sensitivity derivatives computation formula on the variation of the flow variables along the boundaries is eliminated by setting any integral containing these variations to zero. At the same time, the flow boundary conditions must also be considered: variations in the imposed quantities are zero and integrals must be developed so that variations in imposed quantities are eliminated. The factors associated with the remaining flow quantity variations (like variations in pressure for the wall boundaries) are set to zero, which lead to the boundary conditions of adjoint RANS and SA PDEs. Each type of boundary condition gives rise to specific adjoint boundary conditions and sensitivity derivatives equations as exposed in detail in [8, §3.1.4]. Upon convergence of the adjoint PDEs, the expression for computing the sensitivity derivatives includes only surface integrals containing variations in geometric quantities.

The adjoint PDEs equations are solved in PUMA similarly to the primal PDEs. The PhD thesis of K. Tsiakas [8] contains an extensive review of the RANS equations, the continuous adjoint method for compressible and incompressible flows, accompanied with formulations for the boundary conditions and the strategies for their numerical solution in PUMA.

Appendix F

Bi-Conjugate Gradient Stabilized Method

The Bi-Conjugate Gradient Stabilized (BiCGStab) method is used in Chapter 5 for solving the RBF interpolation and is herein analyzed; this requires to briefly illustrate also the Conjugate Gradient (CG) and Bi-Conjugate Gradient (BiCG) method.

Conjugate Gradient Method The CG method is suitable for symmetric positive definite linear systems. The technique generates successive approximations to the solution and corresponding residuals and search directions. Only a few vectors need to be stored in memory, and a few scalars are computed to keep track of orthogonality conditions that indicate that the distance to the exact solution is minimized according to a metric. The metric and method come from the fact that the solution of the linear system is also the zero of the quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^{\mathsf{T}}\boldsymbol{b}. \tag{F.1}$$

The second derivative of $f(\boldsymbol{x})$, $\nabla^2 f(\boldsymbol{x}) = \boldsymbol{A}$ guarantees that a solution exists and is unique since $\boldsymbol{A}$ is positive definite. The first derivative $\nabla f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$ intuitively gives a search direction. However, such a quantity is only used as a metric to check the convergence, namely the residual. The search direction, instead, is computed so that it is orthogonal to previous search directions.

In detail, at an iteration k, the approximated solution $\boldsymbol{x}_k$ is updated by a multiple α_k of the search direction vector $\boldsymbol{p}_k$:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k \boldsymbol{p}_k. \tag{F.2}$$

Equivalently, the residuals $\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$ are updated as

$$\boldsymbol{r}_k = \boldsymbol{r}_{k-1} - \alpha_k \boldsymbol{A}\boldsymbol{p}_k. \tag{F.3}$$

Using

$$\alpha_k = (\boldsymbol{r}_{k-1}^\mathsf{T}\boldsymbol{r}_{k-1})(\boldsymbol{p}_k^\mathsf{T}\boldsymbol{A}\boldsymbol{p}_k) \tag{F.4}$$

minimizes $\boldsymbol{r}_k^\mathsf{T}\boldsymbol{A}\boldsymbol{r}_k$. The search direction is updated by

$$\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{p}_{k-1}. \tag{F.5}$$

Using

$$\beta_k = (\boldsymbol{r}_k^\mathsf{T}\boldsymbol{r}_k)(\boldsymbol{r}_{k-1}^\mathsf{T}\boldsymbol{r}_{k-1}) \tag{F.6}$$

guarantees that $\boldsymbol{r}_{k-1}$ and $\boldsymbol{r}_k$ be orthogonal.

The CG method was originally proposed in [187]. A more formal explanation of the CG method with reference to many theoretical properties and convergence analyses can be found in [188, §7.6] and [189, §2.3.1].

Bi-Conjugate Gradient Method The CG method is not proper for non-symmetric systems because the residual vectors cannot be made orthogonal with short recurrences [190]. The BiCG method replaces the orthogonal sequence of residuals by two mutually orthogonal sequences. In detail, the BiCG method augments the updating formulas of the residuals involving both the coefficient matrix and its transpose. Thus, the technique requires to update two sequences of residuals

$$\boldsymbol{r}_k = \boldsymbol{r}_{k-1} - \alpha_k \boldsymbol{A}\boldsymbol{p}_k \quad \text{and} \quad \hat{\boldsymbol{r}}_k = \hat{\boldsymbol{r}}_{k-1} - \alpha_k \boldsymbol{A}^\mathsf{T}\hat{\boldsymbol{p}}_k \tag{F.7}$$

and two sequences of directions

$$\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{p}_{k-1} \quad \text{and} \quad \hat{\boldsymbol{p}}_k = \hat{\boldsymbol{r}}_k + \beta_{k-1}\hat{\boldsymbol{p}}_{k-1} \tag{F.8}$$

Using

$$\alpha_k = \frac{\hat{\boldsymbol{r}}_{k-1}^\mathsf{T}\boldsymbol{r}_{k-1}}{\hat{\boldsymbol{p}}_k^\mathsf{T}\boldsymbol{A}\boldsymbol{p}_k} \quad \text{and} \quad \beta_k = \frac{\hat{\boldsymbol{r}}_k^\mathsf{T}\boldsymbol{r}_k}{\hat{\boldsymbol{r}}_{k-1}^\mathsf{T}\boldsymbol{r}_{k-1}} \tag{F.9}$$

ensures bi-orthogonality relations $\hat{\boldsymbol{r}}_i^\mathsf{T}\boldsymbol{r}_j = \hat{\boldsymbol{p}}_i\boldsymbol{A}\boldsymbol{p}_j = 0$ with $i \neq j$ to be fulfilled. The BiCG method is unstable and may break down, for instance, when $z_0^\mathsf{T}\hat{\boldsymbol{r}}_0$ approaches 0 [189, §2.3.5]; the BiCGStab method improves upon this strategy.

Algorithm F.1: Bi-conjugate gradient stabilized method [189, §2.3.8]

```
Given $\boldsymbol{x}_0$, $\boldsymbol{A}$, $\boldsymbol{M}$, $\boldsymbol{b}$
$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$
$\hat{\boldsymbol{r}} = \boldsymbol{r}_0$
for $k = 1, 2, \dots$ do
    $\rho_{k-1} = \hat{\boldsymbol{r}}^\mathsf{T}\boldsymbol{r}_{k-1}$
    if $\rho_{k-1} = 0$ then method fails
    if $k = 1$ then
        $\boldsymbol{p}_1 = \boldsymbol{r}_0$
    else
        $\beta_{k-1} = (\rho_{k-1}/\rho_{k-2})(\alpha_{k-1}/\omega_{k-1})$
        $\boldsymbol{p}_k = \boldsymbol{r}_{k-1} + \beta_{k-1}(\boldsymbol{p}_{k-1} - \omega_{k-1}v_{k-1})$
    $\hat{\boldsymbol{p}} = \boldsymbol{M}^{-1}\boldsymbol{p}_k$
    $\boldsymbol{v}_k = \boldsymbol{A}\hat{\boldsymbol{p}}_k$
    $\alpha_k = \rho_{k-1}/(\hat{\boldsymbol{r}}_k^\mathsf{T}\boldsymbol{v}_k)$
    $\boldsymbol{s} = \boldsymbol{r}_{k-1} - \alpha_k\boldsymbol{v}_k$
    if $||\boldsymbol{s}||$ small enough then
        $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k\hat{\boldsymbol{p}}$
        exit the loop
    $\hat{\boldsymbol{s}} = \boldsymbol{M}^{-1}\boldsymbol{s}_k$
    $\boldsymbol{t} = \boldsymbol{A}\hat{\boldsymbol{s}}$
    $w_k = (\boldsymbol{t}^\mathsf{T}\boldsymbol{s})(\boldsymbol{t}^\mathsf{T}\boldsymbol{t})$
    $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k\hat{\boldsymbol{p}} + \omega_k\hat{\boldsymbol{s}}$
    $\boldsymbol{r}_k = \boldsymbol{s} - \omega_k\boldsymbol{t}$
    if Convergence criteria fulfilled then exit the loop
```

Bi-Conjugate Gradient Stabilized Method In the BiCG method, the recurrence operations in equations F.7 and F.8 are used to update the residuals and search directions, respectively. These can also be written as (see [191, §7.4.1]):

$$\begin{aligned} \boldsymbol{r}_k &= P_k(\boldsymbol{A})\boldsymbol{r}_0, \quad \hat{\boldsymbol{r}}_k = P_k(\boldsymbol{A}^\mathsf{T})\hat{\boldsymbol{r}}_0 \\ \boldsymbol{p}_k &= T_k(\boldsymbol{A})\boldsymbol{r}_0, \quad \hat{\boldsymbol{p}}_k = T_k(\boldsymbol{A}^\mathsf{T})\hat{\boldsymbol{r}}_0 \end{aligned} \tag{F.10}$$

where $P_k(\boldsymbol{A})$ and $T_k(\boldsymbol{A})$ are the k^{th}-degree polynomial in $\boldsymbol{A}$. Their recurrence relations are then defined as:

$$P_k(\boldsymbol{A}) = P_{k-1}(\boldsymbol{A}) - \alpha_k\boldsymbol{A}T_{k-1}(\boldsymbol{A}) \quad \text{and} \quad T_k(\boldsymbol{A}) = P_k(\boldsymbol{A}) - \beta_{k+1}T_{k-1}(\boldsymbol{A}) \tag{F.11}$$

This view suggests that, for instance, if $P_k(\boldsymbol{A})$ reduces $\boldsymbol{r}_0$ to a smaller vector $\boldsymbol{r}_k$, then it might be advantageous to apply this "contraction" operator twice, by computing $\boldsymbol{r}_k = P_k^2(\boldsymbol{A})\boldsymbol{r}_0$. Such a strategy is adopted by the Conjugate Gradient Squared (CGS) method.

The BiCGStab method was proposed in [192], and it is based on the same concept, but instead of applying the operators $P_k(\boldsymbol{A})$ and $T_k(\boldsymbol{A})$ twice, it makes use of a new polynomial which is defined recursively for stability purposes. Namely:

$$\begin{aligned} \boldsymbol{r}_k &= Q_k(\boldsymbol{A})P_k(\boldsymbol{A})\boldsymbol{r}_0, \quad \hat{\boldsymbol{r}}_k = Q_k(\boldsymbol{A})P_k(\boldsymbol{A}^\mathsf{T})\hat{\boldsymbol{r}}_0 \\ \boldsymbol{p}_k &= Q_k(\boldsymbol{A})T_k(\boldsymbol{A})\boldsymbol{r}_0, \quad \hat{\boldsymbol{p}}_k = Q_k(\boldsymbol{A})T_k(\boldsymbol{A}^\mathsf{T})\hat{\boldsymbol{r}}_0 \end{aligned} \tag{F.12}$$

The recurrence relation of $Q_k(t)$ is defined by $Q_{k+1}(t) = (1 - \omega_k t)Q_k(t)$; the derivation of the scalar ω_k is described in [191, §7.4.2]. Basically, $Q_k(\boldsymbol{A})$ stands for a steepest descent update. The computation of some parameters appearing in the BiCG method, such as α_k and β_k, must be rearranged based on the new information, as explained in [192]. Moreover, in the BiCG, vectors generated using $\boldsymbol{A}^\mathsf{T}$ and $\boldsymbol{M}^\mathsf{T}$ are only used to compute scalar quantities; additional considerations allow to derive these scalar quantities avoiding the use of the transpose matrices, which, in some applications, are difficult to compute. A resulting strategy based on these modifications is illustrated in Algorithm F.1.

Appendix G

Parametric Effectiveness

Parametric Effectiveness (PE) was introduced in [33] and [35]; it measures the ability of the DoFs defining a CAD shape to be used for optimization. PE has been used in Chapter 2 to assess the B-Rep-Morpher method. It compares the change in performance achievable by a specific parameterization to the performance improvement attainable if the surface of the model was free to move, surface node by surface node (in the discrete sense).

PE is defined as the ratio of the change in performance achieved by perturbing all the DoFs and the shape surface node-by-node in the steepest descent sense subject to the constraint of equal root-mean-squared boundary displacement. In detail, the linearized variation in the objective function F due to a variation in the DoFs $\boldsymbol{b}$, in the steepest descent direction ($\Delta\boldsymbol{b} = -\eta_{\text{CAD}}\,(\mathrm{d}F/\mathrm{d}\boldsymbol{b})$), is defined as

$$\Delta F^{\text{CAD}} = \frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{b}}\Delta\boldsymbol{b} = -\eta_{\text{CAD}}\left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{b}}\right)^2 . \tag{G.1}$$

Equivalently, the linearized variation in F due to a variation in the nodal coordinates $\boldsymbol{x}$ of the shape surface is

$$\Delta F^{\text{Nodal}} = \frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}\Delta\boldsymbol{x} = -\eta_{\text{Nodal}}\left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}\right)^2 . \tag{G.2}$$

These lead to the definition of the PE as

$$\text{PE} = \frac{\Delta F^{\text{CAD}}}{\Delta F^{\text{Nodal}}} = \frac{\eta_{\text{CAD}}}{\eta_{\text{Nodal}}}\frac{\left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{b}}\right)^2}{\left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}\right)^2} . \tag{G.3}$$

The ratio $\eta_{\text{CAD}}/\eta_{\text{Nodal}}$ is computed from the constraint of equal root-mean-squared boundary (S) displacement, by assuming that [33]

$$\int_S \left(\Delta \boldsymbol{x}^{\mathrm{Nodal}} \cdot \boldsymbol{n}\right)^2 \mathrm{d}S = \int_S \left(\Delta \boldsymbol{x}^{\mathrm{CAD}} \cdot \boldsymbol{n}\right)^2 \mathrm{d}S \ . \tag{G.4}$$

By taking into account that

$$\Delta \boldsymbol{x}^{\mathrm{Nodal}} = -\eta_{\mathrm{Nodal}} \left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}\right)^{\mathsf{T}} \quad \text{and} \quad \Delta \boldsymbol{x}^{\mathrm{CAD}} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{b}} \Delta \boldsymbol{b} = -\eta_{\mathrm{CAD}} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{b}} \left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{b}}\right)^{\mathsf{T}} \tag{G.5}$$

we get

$$\frac{\eta_{\mathrm{CAD}}}{\eta_{\mathrm{Nodal}}} = \sqrt{\frac{\int_S \left(\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{b}} \left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{b}}\right)^{\mathsf{T}} \cdot \boldsymbol{n}\right)^2 \mathrm{d}S}{\int_S \left(\left(\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}\right)^{\mathsf{T}} \cdot \boldsymbol{n}\right)^2 \mathrm{d}S}} \ , \tag{G.6}$$

where $\boldsymbol{n}$ is the surface normal.

The quantity $\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{x}}$ is computed by the adjoint method, $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{b}}$ by the differentiation of the shape parameterization and $\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{b}}$ by the chain rule.

Higher PE values indicate that the DoFs of the model can optimally change the shape. Low PE values indicate that the parameterization of the model is less adequate to improve the specific objective function F.

Bibliography

[1] D. Thévenin and G. Janiga. *Optimization and Computational Fluid Dynamics*. Springer Science & Business Media, 2008 (cit. on p. 1).

[2] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, 2007 (cit. on pp. 3, 4).

[3] Wikipedia Contributors. *Computational Fluid Dynamics — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Computational_fluid_dynamics&oldid=915311506`. [Online; accessed 25-September-2019]. 2019 (cit. on p. 4).

[4] Wikipedia Contributors. *Mathematical Optimization — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Mathematical_optimization&oldid=918504161`. [Online; accessed 29-September-2019]. 2019 (cit. on p. 4).

[5] L. S. Lasdon. *Optimization Theory for Large Systems*. Courier Corporation, 2002 (cit. on p. 5).

[6] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998 (cit. on p. 5).

[7] G. R. Anderson and M. J. Aftosmis. 'Adaptive Shape Parameterization for Aerodynamic Design'. In: *NASA - Technical Report NAS-2015-02* (2015) (cit. on p. 5).

[8] K. T. Tsiakas. 'Development of Shape Parameterization Techniques, a Flow Solver and its Adjoint, for Optimization on GPUs. Turbomachinery and External Aerodynamics Applications'. PhD Thesis. National Technical University of Athens (NTUA), 2019 (cit. on pp. 6, 39, 42, 138, 161, 163, 164).

[9] *Pointwise™Mesh Generation Software*. `https://www.pointwise.com/`. [Online; accessed 4-October-2019] (cit. on p. 7).

[10] *Autogrid5™Mesh Generation Software*. `https://www.numeca.com/product/autogrid5`. [Online; accessed 4-October-2019] (cit. on p. 7).

[11] P. N. Koch, T. W. Simpson, J. K. Allen and F. Mistree. 'Statistical Approximations for Multidisciplinary Design Optimization: The Problem of Size'. In: *Journal of Aircraft* 36.1 (1999), pp. 275–286 (cit. on p. 10).

[12] S. Wright and J. Nocedal. *Numerical Optimization*. 2nd ed. Springer-Verlag, 2006 (cit. on pp. 10, 152–155).

[13] K. T. Tsiakas, F. Gagliardi, X. S. Trompoukis and K. C. Giannakoglou. 'Shape Optimization of Turbomachinery Rows using a Parametric Blade Modeller and the Continuous Adjoint Method Running on GPUs'. In: *7th ECCOMAS Conference Proceedings*. 2016 (cit. on pp. 10, 61).

[14] J. A. Samareh. 'Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization'. In: *AIAA Journal* 39.5 (2001), pp. 877–884 (cit. on p. 10).

[15] A. G. Liatsikouras and G. Pierrot. ‘Soft Handle Triggering: A CAD-Free Parameterization Tool for Adjoint-Based Optimization Methods’. In: *6th ECCM and 7th ECFD Conference Proceedings*. 2018 (cit. on p. 10).

[16] D. Sieger, S. Menzel and M. Botsch. ‘On Shape Deformation Techniques for Simulation-Based Design Optimization’. In: *New Challenges in Grid Generation and Adaptivity for Scientific Computing*. Springer, 2015, pp. 281–303 (cit. on pp. 10, 11).

[17] A. M. Morris, C. B. Allen and T. C. S. Rendall. ‘CFD-Based Optimization of Aerofoils using Radial Basis Functions for Domain Element Parameterization and Mesh Deformation’. In: *International Journal for Numerical Methods in Fluids* 58.8 (2008), pp. 827–860 (cit. on p. 11).

[18] M. E. Biancolini, I. M. Viola and M. Riotte. ‘Sails Trim Optimisation using CFD and RBF Mesh Morphing’. In: *Computers & Fluids* 93 (2014), pp. 46–60 (cit. on pp. 11, 12, 69).

[19] T. T. Robinson, C. G. Armstrong, H.-S. Chua, C. Othmer and T. Grahs. ‘Optimizing Parameterized CAD Geometries using Sensitivities Based on Adjoint Functions’. In: *Computer-Aided Design and Applications* 9.3 (2012), pp. 253–268 (cit. on pp. 12, 62).

[20] M. J. Martin, E. Andres, M. Widhalm, P. Bitrian and C. Lozano. ‘Non-Uniform Rational B-Splines-Based Aerodynamic Shape Design Optimization with the DLR TAU Code’. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 226.10 (2012), pp. 1225–1242 (cit. on p. 12).

[21] S. Xu, W. Jahn and J.-D. Muller. ‘CAD-Based Shape Optimisation with CFD using a Discrete Adjoint’. In: *International Journal for Numerical Methods in Fluids* 74.3 (2014), pp. 153–168 (cit. on p. 12).

[22] A. Brune, T. Weber-Martins and R. Anderl. ‘Morphing Boxes for the Integration of Shape Optimization in the Product Design Process’. In: *Computer-Aided Design and Applications* 15.2 (2018), pp. 219–226 (cit. on p. 12).

[23] R. McNeel et al. *Rhinoceros Version 6*. 2018. URL: `https://www.rhino3d.com/` (cit. on p. 12).

[24] Q. Zou and H.-Y. Feng. ‘Push-Pull Direct Modeling of Solid CAD Models’. In: *Advances in Engineering Software* 127 (2019), pp. 59–69 (cit. on p. 13).

[25] Wikipedia Contributors. *ISO 10303 — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=ISO_10303&oldid=927302944`. [Online; accessed 10-January-2020]. 2019 (cit. on p. 13).

[26] *Product Data Representation and Exchange – Part 203: Application Protocol: Configuration Controlled 3D Design of Mechanical Parts and Assemblies*. ISO 10303-242, 2014 (cit. on p. 13).

[27] L. Piegl and W. Tiller. *The NURBS Book*. Springer Science & Business Media, 1997 (cit. on pp. 16, 48, 58, 143, 145).

[28] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009 (cit. on p. 17).

[29] H. Wendland. ‘Piecewise Polynomial, Positive Definite and Compactly Supported Radial Functions of Minimal Degree’. In: *Advances in Computational Mathematics* 4.1 (1995), pp. 389–396 (cit. on p. 17).

[30] Wikipedia Contributors. *Thin Plate Spline — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Thin_plate_spline&oldid=889887924`. [Online; accessed 3-October-2019]. 2019 (cit. on p. 17).

[31] A. de Boer, M. S. Van der Schoot and H. Bijl. 'Mesh Deformation Based on Radial Basis Function Interpolation'. In: *Computers and Structures* 85.1114 (2007), pp. 784–795 (cit. on pp. 17, 70, 72, 87).

[32] G. E. Fasshauer. *Meshfree Approximation Methods with Matlab*. Vol. 6. World Scientific Publishing Company, 2007 (cit. on pp. 18, 67, 72).

[33] T. T. Robinson, C. G. Armstrong and H.-S. Chua. 'Determining the Parametric Effectiveness of a CAD Model'. In: *Engineering with Computers* 29.1 (2013), pp. 111–126 (cit. on pp. 21, 169, 170).

[34] T. T. Robinson, C. G. Armstrong and H.-S. Chua. 'Strategies for Adding Features to cad Models in Order to Optimize Performance'. In: *Structural and Multidisciplinary Optimization* 46.3 (2012), pp. 415–424 (cit. on p. 21).

[35] D. Agarwal, C. Kapellos, T. T. Robinson and C. G. Armstrong. 'Using Parametric Effectiveness for Efficient CAD-Based Adjoint Optimization'. In: *Computer-Aided Design and Applications* 16.4 (2019), pp. 703–719 (cit. on pp. 21, 169).

[36] *6th AIAA CFD Drag Prediction Workshop*. URL: `https://aiaa-dpw.larc.nasa.gov/` (visited on 01/04/2018) (cit. on p. 22).

[37] J. W. Demmel, J. R. Gilbert and X. S. Li. 'An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination'. In: *SIAM Journal on Matrix Analysis and Applications* 20.4 (1999), pp. 915–952 (cit. on p. 27).

[38] H. Wendland. 'On the Smoothness of Positive Definite and Radial Functions'. In: *Journal of Computational and Applied Mathematics* 101.1-2 (1999), pp. 177–188 (cit. on p. 28).

[39] M. Drela. 'Pros and Cons of Airfoil Optimization'. In: *Frontiers of Computational Fluid Dynamics* 1998 (1998), pp. 363–381 (cit. on p. 28).

[40] J. J. Thibert, M. Granjacques and L. H. Ohman. *AGARD Advisory Report No. 138 - Experimental Data Base for Computer Program Assessment*. Advisory Group for Aerospace Research and Development, 1979 (cit. on p. 28).

[41] *2nd AIAA CFD Drag Prediction Workshop*. URL: `https://aiaa-dpw.larc.nasa.gov/Workshop2/` (visited on 01/04/2018) (cit. on p. 34).

[42] J. Brezillon and R. P. Dwight. 'Aerodynamic Shape Optimization using the Discrete Adjoint of the Navier-Stokes Equations: Applications Towards Complex 3D Configurations'. In: *KATnet II Conference on Key Aerodynamic Technologies*. 36-1. 2009 (cit. on p. 36).

[43] H. N. Cantrell and J. E. Fowler. 'The Aerodynamic Design of Two-Dimensional Turbine Cascades for Incompressible Flow with a High-Speed Computer'. In: *Journal of Basic Engineering* 81.3 (1959), pp. 349–359 (cit. on p. 40).

[44] L. J. Pritchard. 'An Eleven Parameter Axial Turbine Airfoil Geometry Model'. In: *ASME 1985 International Gas Turbine Conference and Exhibit*. Vol. 1. 3. American Society of Mechanical Engineers. 1985, pp. 1–12 (cit. on p. 40).

[45] M. A. Trigg, G. R. Tubby and A. G. Sheard. 'Automatic Genetic Optimization Approach to Two-Dimensional Blade Profile Design for Steam Turbines'. In: *Journal of Turbomachinery* 121.1 (1999), pp. 11–17 (cit. on p. 40).

[46] T. Korakianitis. 'Hierarchical Development of Three Direct-Design Methods for Two-Dimensional Axial-Turbomachinery Cascades'. In: *Journal of Turbomachinery* 115.2 (1993), pp. 314–324 (cit. on p. 40).

[47] T. Korakianitis. 'Prescribed-Curvature-Distribution Airfoils for the Preliminary Geometric Design of Axial-Turbomachinery Cascades'. In: *Journal of Turbomachinery* 115.2 (1993), pp. 325–333 (cit. on p. 40).

[48] J. Hoschek and R. Muller. ‘Turbine Blade Design by Lofted B-Spline Surfaces’. In: *Journal of Computational and Applied Mathematics* 119.1-2 (2000), pp. 235–248 (cit. on p. 40).

[49] S. Goel, J. I. Cofer and H. Singh. ‘Turbine Airfoil Design Optimization’. In: *ASME 1996 International Gas Turbine and Aeroengine Congress and Exhibition*. American Society of Mechanical Engineers. 1996, V001T01A055–V001T01A055 (cit. on p. 40).

[50] J. Li, Z. Feng, J. Chang and Z. Shen. ‘Aerodynamic Optimum Design of Transonic Turbine Cascades using Genetic Algorithms’. In: *Journal of Thermal Science* 6.2 (1997), pp. 111–116 (cit. on p. 40).

[51] K. Yamamoto and O. Inoue. ‘Applications of Genetic Algorithm to Aerodynamic Shape Optimization’. In: *12th Computational Fluid Dynamics Conference*. 1995, p. 1650 (cit. on p. 40).

[52] M. Rossgatterer, B. Juttler, M. Kapl and G. Della Vecchia. ‘Medial Design of Blades for Hydroelectric Turbines and Ship Propellers’. In: *Computers & Graphics* 36.5 (2012), pp. 434–444 (cit. on p. 41).

[53] *TURBODesign Suite*. [Online; accessed 14-May-2019]. 2019. URL: `https://www.adtechnology.com/` (cit. on p. 41).

[54] *PropCad*. [Online; accessed 14-May-2019]. 2019. URL: `https://hydrocompinc.com/software/propcad` (cit. on p. 41).

[55] *FINE/Design3D*. [Online; accessed 14-May-2019]. 2019. URL: `https://www.numeca.com/product/finedesign3d` (cit. on p. 41).

[56] *BladeModeler*. [Online; accessed 14-May-2019]. 2019. URL: `https://www.ansys.com/products/fluids/ansys-blademodeler` (cit. on p. 41).

[57] P. L. Miller, J. H. Oliver, D. P. Miller and D. L. Tweedt. ‘BladeCAD: An Interactive Geometric Design Tool for Turbomachinery Blades’. In: (1996) (cit. on p. 42).

[58] G. N. Koini, S. S. Sarakinos and I. K. Nikolos. ‘A Software Tool for Parametric Design of Turbomachinery Blades’. In: *Advances in Engineering Software* 40.1 (2009), pp. 41–51 (cit. on p. 42).

[59] *Solidworks*. [Online; accessed 25-July-2019]. 2019. URL: `https://www.solidworks.com/it` (cit. on p. 42).

[60] Wikipedia Contributors. *Hash table — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Hash_table&oldid=904649285`. [Online; accessed 8-July-2019]. 2019 (cit. on p. 47).

[61] S.-M. Hu, C.-L. Tai and S.-H. Zhang. ‘An Extension Algorithm for B-Splines by Curve Unclamping’. In: *Computer-Aided Design* 34.5 (2002), pp. 415–419 (cit. on p. 48).

[62] D. G. Kirkpatrick. ‘Efficient Computation of Continuous Skeletons’. In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE. 1979, pp. 18–27 (cit. on p. 50).

[63] Wikipedia Contributors. *Voronoi diagram — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-July-2019]. 2019. URL: `https://en.wikipedia.org/w/index.php?title=Voronoi_diagram&oldid=903684632` (cit. on p. 50).

[64] M. Schmitt. ‘Some Examples of Algorithms Analysis in Computational Geometry by Means of Mathematical Morphological Techniques’. In: *French Workshop on Geometry and Robotics*. Springer. 1988, pp. 225–246 (cit. on p. 50).

[65] Wikipedia Contributors. *Point in Polygon — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Point_in_polygon&oldid=906498053`. [Online; accessed 17-July-2019]. 2019 (cit. on p. 50).

[66] Wikipedia Contributors. *k-means Clustering — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=905043786. [Online; accessed 17-July-2019]. 2019 (cit. on p. 52).

[67] L. Fottner. *Test Cases for Computation of Internal Flows in Aero Engine Components*. AGARD-AR-275. 1990 (cit. on p. 53).

[68] Wikipedia Contributors. *Automatic Differentiation — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Automatic_differentiation&oldid=944490943. [Online; accessed 14-April-2020]. 2020 (cit. on p. 61).

[69] Wikipedia Contributors. *Finite Difference — Wikipedia, Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Finite_difference&oldid=948576715. [Online; accessed 14-April-2020]. 2020 (cit. on p. 61).

[70] S. Chen and D. Torterelli. 'Three-Dimensional Shape Optimization with Variational Geometry'. In: *Structural Optimization* 13.2-3 (1997), pp. 81–94 (cit. on p. 61).

[71] E. Hardee, K.-H. Chang, J. Tu, K. K. Choi, I. Grindeanu and X. Yu. 'A CAD-Based Design Parameterization for Shape Optimization of Elastic Solids'. In: *Advances in Engineering Software* 30.3 (1999), pp. 185–199 (cit. on p. 61).

[72] M. Nemec and M. J. Aftosmis. 'Adjoint Sensitivity Computations for an Embedded-Boundary Cartesian Mesh Method'. In: *Journal of Computational Physics* 227.4 (2008), pp. 2724–2742 (cit. on p. 61).

[73] D. Agarwal, T. T. Robinson, C. G. Armstrong, S. Marques, I. Vasilopoulos and M. Meyer. 'Parametric Design Velocity Computation for CAD-Based Design Optimization using Adjoint Methods'. In: *Engineering with Computers* 34.2 (2018), pp. 225–239 (cit. on p. 62).

[74] M. Banović, O. Mykhaskiv, S. Auriemma, A. Walther, H. Legrand and J.-D. Müller. 'Algorithmic Differentiation of the Open CASCADE Technology CAD Kernel and its Coupling with an Adjoint CFD Solver'. In: *Optimization Methods and Software* 33.4-6 (2018), pp. 813–828 (cit. on p. 62).

[75] M. M. Selim and R. P. Koomullil. 'Mesh Deformation Approaches – A Survey'. In: *Journal of Physical Mathematics* 7.181 (2016), pp. 2090–0902 (cit. on pp. 68, 69).

[76] J. Niu, J. Lei and J. He. 'Radial Basis Function Mesh Deformation Based on Dynamic Control Points'. In: *Aerospace Science and Technology* 64 (2017), pp. 122–132 (cit. on p. 68).

[77] G. A. Strofylas, G. N. Lygidakis and I. K. Nikolos. 'An Agglomeration Strategy for Accelerating RBF-Based Mesh Deformation'. In: *Advances in Engineering Software* 107 (2017), pp. 13–37 (cit. on pp. 68, 70).

[78] T. C. S. Rendall and C. B. Allen. 'Parallel Efficient Mesh Motion using Radial Basis Functions With Application to Multi-Bladed Rotors'. In: *International Journal for Numerical Methods in Engineering* 81.1 (2010), pp. 89–105 (cit. on pp. 68, 70, 71, 74).

[79] C. B. Allen. 'Parallel Universal Approach to Mesh Motion and Application to Rotors in Forward Flight'. In: *International Journal for Numerical Methods in Engineering* 69.10 (2007), pp. 2126–2149 (cit. on p. 68).

[80] M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger and T. S. Coffey. 'A Comparison of Mesh Morphing Methods for 3D Shape Optimization'. In: *Proceedings of the 20th International Meshing Roundtable*. Springer, 2011, pp. 293–311 (cit. on pp. 68, 69).

[81] J. Batina. 'Unsteady Euler Airfoil Solutions using Unstructured Dynamic Meshes'. In: *AIAA Journal* 28.8 (1990), pp. 1381–1388 (cit. on p. 68).

[82] C. Degand and C. Farhat. 'A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes'. In: *Computers & structures* 80.3 (2002), pp. 305–316 (cit. on p. 68).

[83] D. Zeng and C. R. Ethier. 'A Semi-Torsional Spring Analogy Model for Updating Unstructured Meshes in 3D Moving Domains'. In: *Finite Elements in Analysis and Design* 41.11 (2005), pp. 1118–1139 (cit. on p. 68).

[84] C. L. Bottasso, D. Detomi and R. Serra. 'The Ball-Vertex Method: A New Simple Spring Analogy Method for Unstructured Dynamic Meshes'. In: *Computer Methods in Applied Mechanics and Engineering* 194.39 (2005), pp. 4244–4264 (cit. on p. 68).

[85] S. H. Huo, F. S. Wang, W. Z. Yan and Z. F. Yue. 'Layered Elastic Solid Method for the Generation of Unstructured Dynamic Mesh'. In: *Finite Elements in Analysis and Design* 46.10 (2010), pp. 949–955 (cit. on p. 68).

[86] S.-Y. Hsu and C.-L. Chang. 'Mesh Deformation Based on Fully Stressed Design: The Method and 2-D Examples'. In: *International Journal for Numerical Methods in Engineering* 72.5 (2007), pp. 606–629 (cit. on p. 68).

[87] K. Stein, T. E. Tezduyar and R. Benney. 'Automatic Mesh Update with the Solid-Extension Mesh Moving Technique'. In: *Computer Methods in Applied Mechanics and Engineering* 193.21 (2004), pp. 2019–2032 (cit. on p. 68).

[88] C. Burg. 'Analytic Study of 2D and 3D Grid Motion using Modified Laplacian'. In: *International Journal for Numerical Methods in Fluids* 52.2 (2006), pp. 163–197 (cit. on p. 69).

[89] B. T. Helenbrook. 'Mesh Deformation using the Biharmonic Operator'. In: *International Journal for Numerical Methods in Engineering* 56.7 (2003), pp. 1007–1021 (cit. on p. 69).

[90] Y. Zhao and A. Forhad. 'A General Method for Simulation of Fluid Flows with Moving and Compliant Boundaries on Unstructured Grids'. In: *Computer Methods in Applied Mechanics and Engineering* 192.39 (2003), pp. 4439–4466 (cit. on p. 69).

[91] X. Liu, N. Qin and H. Xia. 'Fast Dynamic Grid Deformation Based on Delaunay Graph Mapping'. In: *Journal of Computational Physics* 211.2 (2006), pp. 405–423 (cit. on p. 69).

[92] J. A. S. Witteveen. 'Explicit and Robust Inverse Distance Weighting Mesh Deformation for CFD'. In: *48th AIAA Aerospace Sciences Meeting*. Vol. 165. 2010 (cit. on p. 69).

[93] J. F. Thompson, Z. U. Warsi and W. C. Mastin. *Numerical Grid Generation: Foundations and Applications*. Vol. 45. North-Holland Amsterdam, 1985 (cit. on p. 69).

[94] H. M. Tsai, A. S. F. Wong, J. Cai, Y. Zhu and F. Liu. 'Unsteady Flow Calculations with a Parallel Multiblock Moving Mesh Algorithm'. In: *AIAA Journal* 39.6 (2001), pp. 1021–1205 (cit. on p. 69).

[95] D. Sieger, S. Menzel and M. Botsch. 'RBF Morphing Techniques for Simulation-Based Design Optimization'. In: *Engineering with Computers* 30.2 (2014), pp. 161–174 (cit. on p. 69).

[96] M. S. Floater and A. Iske. 'Multistep Scattered Data Interpolation using Compactly Supported Radial Basis Functions'. In: *Journal of Computational and Applied Mathematics* 73.1 (1996), pp. 65–78 (cit. on p. 70).

[97] F. J. Narcowich, R. Schaback and J. D. Ward. 'Multilevel Interpolation and Approximation'. In: *Applied and Computational Harmonic Analysis* 7.3 (1999), pp. 243–261 (cit. on p. 70).

[98] D. Lazzaro and L. B. Montefusco. 'Radial Basis Functions for the Multivariate Interpolation of Large Scattered Data Sets'. In: *Journal of Computational and Applied Mathematics* 140.1 (2002), pp. 521–536 (cit. on p. 70).

[99] M. Cordero-Gracia, M. S. Gomez, J. Ponsin and E. Valero. ‘An Interpolation Tool for Aerodynamic Mesh Deformation Problems Based on Octree Decomposition’. In: *Aerospace Science and Technology* 23.1 (2012), pp. 93–107 (cit. on p. 70).

[100] T. C. S. Rendall and C. B. Allen. ‘Reduced Surface Point Selection Options for Efficient Mesh Deformation using Radial Basis Functions’. In: *Journal of Computational Physics* 229.8 (2010), pp. 2810–2820 (cit. on p. 70).

[101] M. Botsch and L. Kobbelt. ‘Real-Time Shape Editing using Radial Basis Functions’. In: *Computer Graphics Forum*. Vol. 24. 3. Wiley Online Library. 2005, pp. 611–621 (cit. on p. 70).

[102] L. Kedward, C. B. Allen and T. C. S. Rendall. ‘Efficient and Exact Mesh Deformation using Multiscale RBF Interpolation’. In: *Journal of Computational Physics* 345 (2017), pp. 732–751 (cit. on p. 70).

[103] T. Gillebaart, D. S. Blom, A. H. van Zuijlen and H. Bijl. ‘Adaptive Radial Basis Function Mesh Deformation using Data Reduction’. In: *Journal of Computational Physics* 321 (2016), pp. 997–1025 (cit. on p. 71).

[104] G. F. Fasshauer. *Meshfree Approximation Methods with MATLAB*. Illinois Institute of Technology: World Scientific Pub. Co. Inc., Apr. 2007 (cit. on p. 72).

[105] A. Kallischko. ‘Modified Sparse Approximate Inverses (MSPAI) for Parallel Preconditioning’. PhD Thesis. Technische Universitat Munchen, 2007 (cit. on pp. 76, 77, 79, 82).

[106] W. Fong and E. Darve. ‘The Black-Box Fast Multipole Method’. In: *Journal of Computational Physics* 228.23 (2009), pp. 8712–8725 (cit. on pp. 76, 84, 101, 110).

[107] M. Benzi and M. Tuma. ‘A Comparative Study of Sparse Approximate Inverse Preconditioners’. In: *Applied Numerical Mathematics* 30.2-3 (1999), pp. 305–340 (cit. on p. 77).

[108] J. Rahola. *Experiments on Iterative Methods and the Fast Multipole Method in Electromagnetic Scattering Calculations. Tech. Rep. TR/PA/98/49*. Tech. rep. CERFACS, Toulouse, France, 1998 (cit. on p. 77).

[109] M. Benzi. ‘Preconditioning Techniques for Large Linear Systems: A Survey’. In: *Journal of Computational Physics* 182.2 (2002), pp. 418–477 (cit. on pp. 77, 79).

[110] B. Carpentieri. ‘Algebraic Preconditioners for the Fast Multipole Method in Electromagnetic Scattering Analysis from Large Structures: Trends and Problems’. In: *Electronic Journal of Boundary Elements* 7.1 (2009), pp. 13–49 (cit. on p. 77).

[111] S. Demko, W. F. Moss and P. W. Smith. ‘Decay Rates for Inverses of Band Matrices’. In: *Mathematics of Computation* 43.168 (1984), pp. 491–499 (cit. on pp. 77, 79).

[112] G. H. Golub and C. F. Van Loan. *Matrix Computations*. 4th ed. Vol. 3. Johns Hopkins University Press, 2012 (cit. on p. 77).

[113] B. Carpentieri, I. S. Duff and L. Giraud. ‘Sparse Pattern Selection Strategies for Robust Frobenius-Norm Minimization Preconditioners in Electromagnetism’. In: *Numerical Linear Algebra with Applications* 7.7-8 (2000), pp. 667–685 (cit. on p. 79).

[114] G. Alleon, M. Benzi and L. Giraud. ‘Sparse Approximate Inverse Preconditioning for Dense Linear Systems Arising in Computational Electromagnetics’. In: *Numerical Algorithms* 16.1 (1997), pp. 1–15 (cit. on p. 79).

[115] B. Carpentieri, I. S. Duff, L. Giraud and M. Magolu Monga Made. ‘Sparse Symmetric Preconditioners for Dense Linear Systems in Electromagnetism’. In: *Numerical Linear Algebra with Applications* 11.8-9 (2004), pp. 753–771 (cit. on p. 83).

[116] L. Greengard and V. Rokhlin. ‘A Fast Algorithm for Particle Simulations’. In: *Journal of Computational Physics* 73.2 (1987), pp. 325–348 (cit. on pp. 83, 95, 101).

[117] N. A. Gumerov and R. Duraiswami. 'Fast Radial Basis Function Interpolation via Preconditioned Krylov Iteration'. In: *SIAM Journal on Scientific Computing* 29.5 (2007), pp. 1876–1899 (cit. on p. 84).

[118] B. Carpentieri, I. S. Duff, L. Giraud and G. Sylvand. 'Combining Fast Multipole Techniques and an Approximate Inverse Preconditioner for Large Electromagnetism Calculations'. In: *SIAM Journal on Scientific Computing* 27.3 (2005), pp. 774–792 (cit. on p. 84).

[119] A. I. Heft, T. Indinger and N. A. Adams. *Introduction of a New Realistic Generic Car Model for Aerodynamic Investigations*. Tech. rep. SAE Technical Paper, 2012 (cit. on p. 88).

[120] E. M. Papoutsis-Kiachagias, S. Porziani, C. Groth, M. E. Biancolini, E. Costa and K. C. Giannakoglou. 'Aerodynamic Optimization of Car Shapes using the Continuous Adjoint Method and an RBF Morpher'. In: *11 th International Conference EUROGEN*. 2015, pp. 14–16 (cit. on p. 88).

[121] J. Barnes and P. Hut. 'A Hierarchical O(N Log N) Force-Calculation Algorithm'. In: *Nature* 324.6096 (1986), p. 446 (cit. on p. 95).

[122] V. Rokhlin. 'Diagonal Forms of Translation Operators for the Helmholtz Equation in Three Dimensions'. In: *Applied and Computational Harmonic Analysis* 1.1 (1993), pp. 82–93 (cit. on p. 101).

[123] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin and J. Zhao. 'A Wideband Fast Multipole Method for the Helmholtz Equation in Three Dimensions'. In: *Journal of Computational Physics* 216.1 (2006), pp. 300–325 (cit. on p. 101).

[124] L. Greengard and V. Rokhlin. 'A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions'. In: *Acta Numerica* 6 (1997), pp. 229–269 (cit. on p. 101).

[125] E. Darve and P. Have. 'Efficient Fast Multipole Method for Low-Frequency Scattering'. In: *Journal of Computational Physics* 197.1 (2004), pp. 341–363 (cit. on p. 101).

[126] Z. Gimbutas and V. Rokhlin. 'A Generalized Fast Multipole Method for Nonoscillatory Kernels'. In: *SIAM Journal on Scientific Computing* 24.3 (2003), pp. 796–817 (cit. on p. 101).

[127] M. Messner, B. Bramas, O. Coulaud and E. Darve. *Optimized M2L Kernels for the Chebyshev Interpolation Based Fast Multipole Method*. Research Report. Oct. 2012, p. 22. URL: `https://hal.inria.fr/hal-00746089` (cit. on pp. 101, 110).

[128] L. N. Trefethen. *Approximation Theory and Approximation Practice*. Vol. 128. SIAM, 2013 (cit. on p. 102).

[129] J. C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman and Hall/CRC, 2002 (cit. on p. 103).

[130] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner and T. Takahashi. 'Task-Based FMM for Multicore Architectures'. In: *SIAM Journal on Scientific Computing* 36.1 (2014), pp. C66–C93 (cit. on pp. 109, 111, 115).

[131] O. Coulaud, P. Fortin and J. Roman. 'Hybrid MPI-Thread Parallelization of the Fast Multipole Method'. In: *Parallel and Distributed Computing, 2007. ISPDC'07*. IEEE. 2007, p. 52 (cit. on p. 111).

[132] A. Chandramowlishwaran, S. Williams, L. Oliker, I. Lashuk, G. Biros and R. Vuduc. 'Optimizing and Tuning the Fast Multipole Method for State-of-the-Art Multicore Architectures'. In: *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. IEEE. 2010, pp. 1–12 (cit. on p. 115).

[133] R. Yokota and L. A. Barba. 'A Tuned and Scalable Fast Multipole Method as a Preeminent Algorithm for Exascale Systems'. In: *The International Journal of High Performance Computing Applications* 26.4 (2012), pp. 337–346 (cit. on p. 115).

[134] Wikipedia Contributors. *SIMD — Wikipedia, The Free Encyclopedia.* `https://en.wikipedia.org/w/index.php?title=SIMD&oldid=875674721`. [Online; accessed 5-February-2019]. 2018 (cit. on p. 115).

[135] Wikipedia Contributors. *OpenMP — Wikipedia, The Free Encyclopedia.* `https://en.wikipedia.org/w/index.php?title=OpenMP&oldid=868943699`. [Online; accessed 5-February-2019]. 2018 (cit. on p. 115).

[136] I. Vasilopoulos, P. Flassig, M. Meyer and K. Reiche. 'Aerodynamic Optimization of the Turbolab Stator: A Comparative Study between Conventional and Adjoint-Based Approaches'. In: *International Conference on Numerical Optimization Methods for Engineering Design NOED*. 2016 (cit. on p. 117).

[137] J.-D. Mueller, J. Hueckelheim and O. Mykhaskiv. 'STAMPS: A Finite-Volume Solver Framework for Adjoint Codes Derived with Source-Transformation AD'. In: *2018 Multidisciplinary Analysis and Optimization Conference*. 2018, p. 2928 (cit. on p. 117).

[138] O. Mykhaskiv, M. Banovic, S. Auriemma, P. Mohanamuraly, A. Walther, H. Legrand and J.-D. Muller. 'NURBS-Based and Parametric-Based Shape Optimization with Differentiated CAD Kernel'. In: *Computer-Aided Design and Applications* 15.6 (2018), pp. 916–926 (cit. on p. 117).

[139] M. Banović, I. Vasilopoulos, A. Walther and M. Meyer. 'Algorithmic Differentiation of an Industrial Airfoil Design Tool Coupled with the Adjoint CFD Method'. In: *Optimization and Engineering* (2019), pp. 1–22 (cit. on pp. 117, 126).

[140] I. Vasilopoulos. 'CAD-Based and CAD-Free Aerodynamic Shape Optimization of Turbomachinery Blade Rows using the Adjoint Method'. PhD Thesis. National Technical University of Athens (NTUA), 2020 (cit. on p. 117).

[141] J. Mihalyovics, C. Bruck, D. Peitsch, I. Vasilopoulos and M. Meyer. 'Numerical and Experimental Investigations on Optimized 3D Compressor Airfoils'. In: *ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition*. 2018, AT39A038 (cit. on pp. 117, 126).

[142] AboutFlow Project Website: TU Berlin TurboLab Stator Case. 2016. URL: `http://aboutflow.sems.qmul.ac.uk/events/munich2016/benchmark/testcase3/` (cit. on p. 117).

[143] D. F. Rogers. *An Introduction to NURBS: With Historical Perspective*. Elsevier, 2000 (cit. on p. 143).

[144] J. Gallier and J. H. Gallier. *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*. Morgan Kaufmann, 2000 (cit. on p. 143).

[145] *Open CASCADE Technology (OCCT) C++ Libraries*. `https://www.opencascade.com/`. [Online; accessed 5-October-2019] (cit. on p. 143).

[146] C. de Boor. *Subroutine Package for Calculating with B-Splines*. Tech. rep. Los Alamos Scientific Lab., N. Mex., 1971 (cit. on p. 144).

[147] E. Abbena, S. Salamon and A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. Chapman and Hall/CRC, 2017 (cit. on p. 150).

[148] A.-L. Cauchy. 'Méthode Générale pour la Résolution des Systemes d'Equations Simultanées'. In: *Comptes Rendus sci.* 25.1847 (1847), pp. 536–538 (cit. on p. 151).

[149] Wikipedia contributors. *Newton's Method in Optimization — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Newton%27s_method_in_optimization&oldid=924735934`. [Online; accessed 23-November-2019]. 2019 (cit. on pp. 151, 152).

[150] F. E. Curtis and X. Que. 'A Quasi-Newton Algorithm for Nonconvex, Nonsmooth Optimization with Global Convergence Guarantees'. In: *Mathematical Programming Computation* 7.4 (2015), pp. 399–428 (cit. on pp. 151, 152).

[151] R. H. Byrd, R. B. Schnabel and G. A. Shultz. 'A Trust Region Algorithm for Nonlinearly Constrained Optimization'. In: *SIAM Journal on Numerical Analysis* 24.5 (1987), pp. 1152–1170 (cit. on p. 151).

[152] F. H. Khalfan, R. H. Byrd and R. B. Schnabel. 'A Theoretical and Experimental Study of the Symmetric Rank-One Update'. In: *SIAM Journal on Optimization* 3.1 (1993), pp. 1–24 (cit. on p. 153).

[153] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2013 (cit. on p. 153).

[154] K. Schittkowski. 'NLPQL: a FORTRAN Subroutine Solving Constrained Nonlinear Programming Problems'. In: *Annals of Operations Research* 5.2 (1986), pp. 485–500 (cit. on p. 154).

[155] M. C. Biggs. 'Constrained Minimization using Recursive Quadratic Programming'. In: *Towards Global Optimization* (1975) (cit. on p. 154).

[156] S.-P. Han. 'A Globally Convergent Method for Nonlinear Programming'. In: *Journal of Optimization Theory and Applications* 22.3 (1977), pp. 297–309 (cit. on pp. 154, 155).

[157] M. J. D. Powell. 'A Fast Algorithm for Nonlinearly Constrained Optimization Calculations'. In: *Numerical Analysis* (1978), pp. 144–157 (cit. on pp. 154, 155).

[158] M. J. D. Powell. 'The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimization Calculations'. In: *Nonlinear Programming 3*. Elsevier, 1978, pp. 27–63 (cit. on p. 154).

[159] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. 'Optimization by Simulated Annealing'. In: *Science* 220.4598 (1983), pp. 671–680 (cit. on p. 157).

[160] Y. Zhang, S. Wang and G. Ji. 'A Comprehensive Survey on Particle Swarm Optimization Algorithm and its Applications'. In: *Mathematical Problems in Engineering* 2015 (2015) (cit. on p. 157).

[161] Y. Wang and C. A. Shoemaker. 'A General Stochastic Algorithmic Framework for Minimizing Expensive Black Box Objective Functions Based on Surrogate Models and Sensitivity Analysis'. In: *ArXiv* (2014) (cit. on p. 157).

[162] M. A. Abramson. 'Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems'. PhD Thesis. Department of Computational and Applied Mathematics, Rice University, 2002 (cit. on p. 157).

[163] D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer Science & Business Media, 2006 (cit. on p. 157).

[164] D. H. Kapsoulis. 'Low-Cost Metamodel-Assisted Evolutionary Algorithms with Applications in Shape Optimization in Fluid Dynamics'. PhD Thesis. National Technical University of Athens (NTUA), 2019 (cit. on pp. 157–161).

[165] M. K. Karakasis. 'Hierarchical, Distributed Evolutionary Algorithms and Computational Intelligence in Aerodynamic Shape Optimization, on Multiprocessing Systems'. PhD Thesis. National Technical University of Athens (NTUA), 2006 (cit. on p. 157).

[166] I. C. Kampolis. 'Parallel, Multilevel Algorithms for the Aerodynamic Optimization in Turbomachines'. PhD Thesis. National Technical University of Athens (NTUA), 2009 (cit. on p. 157).

[167] A. P. Giotis. 'Application of Evolutionary Algorithms, Computational Intelligence and Advanced Computational Fluid Dynamics Techniques to the Optimization-Inverse Design of Turbomachinery Cascades, using Parallel Processing'. PhD Thesis. National Technical University of Athens (NTUA), 2003 (cit. on p. 157).

[168] T. Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996 (cit. on p. 157).

[169] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Vol. 16. John Wiley & Sons, 2001 (cit. on p. 157).

[170] C. M. Fonseca and P. J. Fleming. 'Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms. I. A Unified Formulation'. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 28.1 (1998), pp. 26–37 (cit. on p. 158).

[171] H.-M. Gutmann. 'A Radial Basis Function Method for Global Optimization'. In: *Journal of Global Optimization* 19.3 (2001), pp. 201–227 (cit. on p. 158).

[172] Y. Jin. 'Surrogate-Assisted Evolutionary Computation: Recent Advances and Future Challenges'. In: *Swarm and Evolutionary Computation* 1.2 (2011), pp. 61–70 (cit. on p. 158).

[173] M. K. Karakasis and K. C. Giannakoglou. 'On the Use of Metamodel-Assisted, Multi-Objective Evolutionary Algorithms'. In: *Engineering Optimization* 38.8 (2006), pp. 941–957 (cit. on p. 158).

[174] D. Lim, Y.-S. Ong, Y. Jin and B. Sendhoff. 'A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation'. In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. ACM. 2007, pp. 1288–1295 (cit. on p. 158).

[175] The EASY (Evolutionary Algorithms SYstem) Software. 2008. URL: `http://velos0.ltt.mech.ntua.gr/EASY` (cit. on p. 158).

[176] M. K. Karakasis, A. P. Giotis and K. C. Giannakoglou. 'Inexact Information Aided, Low-Cost, Distributed Genetic Algorithms for Aerodynamic Shape Optimization'. In: *International Journal for Numerical Methods in Fluids* 43.10-11 (2003), pp. 1149–1166 (cit. on p. 159).

[177] D. G. Koubogiannis. 'Numerical Solution of the Navier-Stokes Equations on Unstructured Grids in a Parallel Processing Environment'. PhD Thesis. National Technical University of Athens (NTUA), 1998 (cit. on p. 161).

[178] 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 (NTUA), 2005 (cit. on p. 161).

[179] V. G. Asouti. 'Aerodynamic Analysis and Design Methods at High and Low Speed Flows, on Multiprocessor Platforms'. PhD Thesis. National Technical University of Athens (NTUA), 2009 (cit. on p. 161).

[180] T. D. 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 (NTUA), 2011 (cit. on p. 161).

[181] X. S. Trompoukis. 'Solving Aerodynamic-Aeroelastic Problems on Graphics Processing Units'. PhD Thesis. National Technical University of Athens (NTUA), 2012 (cit. on p. 161).

[182] P. Spalart and S. Allmaras. ‘A One-Equation Turbulence Model for Aaerodynamic Flows’. In: *30th Aerospace Sciences Meeting and Exhibit*. 1992, p. 439 (cit. on pp. 161, 162).

[183] *NVIDIA Corporation. NVIDIA CUDA Homepage*. `https://developer.nvidia.com/cuda-zone`. [Online; accessed 11-April-2020] (cit. on p. 161).

[184] 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’. In: *International Journal for Numerical Methods in Fluids* 67.2 (2011), pp. 232–246 (cit. on p. 161).

[185] P. L. Roe. ‘Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes’. In: *Journal of Computational Physics* 43.2 (1981), pp. 357–372 (cit. on p. 163).

[186] V. Venkatakrishnan. ‘On the Accuracy of Limiters and Convergence to Steady State Solutions’. In: *31st Aerospace Sciences Meeting*. 1993, p. 880 (cit. on p. 163).

[187] M. R. Hestenes and E. Stiefel. *Methods of Conjugate Gradients for Solving Linear Systems*. Vol. 49. 1. NBS Washington, DC, 1952 (cit. on p. 166).

[188] R. L. Burden, J. D. Faires and A. C. Reynolds. *Numerical Analysis*. PWS Publishing Company, 1978 (cit. on p. 166).

[189] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Vol. 43. SIAM, 1994 (cit. on pp. 166, 167).

[190] V. V. Voevodin. ‘The Problem of Non-Self-Adjoint Generalization of the Conjugate Gradient Method is Closed’. In: *USSR Computational Mathematics and Mathematical Physics* 23 (1983), pp. 143–144 (cit. on p. 166).

[191] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Vol. 82. SIAM, 2003 (cit. on pp. 167, 168).

[192] H. Van der Vorst. ‘Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems’. In: *SIAM Journal on Scientific and Statistical Computing* 13.2 (1992), pp. 631–644 (cit. on pp. 167, 168).

Acronyms

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
a.k.a.	also known as
B-Rep	Boundary-Representation
bbFMM	black-box Fast Multipole Method
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BiCG	Bi-Conjugate Gradient
BiCGStab	Bi-Conjugate Gradient Stabilized
BLAS	Basic Linear Algebra Subprograms
CAD	Computer-Aided-Design
CFD	Computational Fluid Dynamics
CG	Conjugate Gradient
CGS	Conjugate Gradient Squared
CUDA	Compute Unified Device Architecture
DoE	Design of Experiments
DoF	Degree of Freedom
EA	Evolutionary Algorithm
EASY	Evolutionary Algorithm System
EU	European Union
FD	Finite Differences
FEM	Finite Element Method
FFD	Free Form Deformation
FMM	Fast Multipole Method
FSPAI	Factorized Sparse Approximate Inverse
GMRES	Generalized Minimal Residual
GMTurbo	Geometric Modeler for Turbomachinery
GPS	Generalized Pattern Search
GPU	Graphics Processing Unit
HPC	High-Performance Computing
IODA	Industrial Optimal Design using Adjoint CFD
ITN	Initial Training Network
KKT	Karush-Kuhn-Tucker
L2L	Local to Local
L2P	Local to Particles
LE	Leading Edge
LP	Linear Programming
M2L	Multipole to Local
M2M	Multipole to Multipole

M2P	Multipole to Particle
MAEA	Metamodel-Assisted EA
MPI	Message Passing Interface
NP	Non-Linear Programming
NS	Navier-Stokes
NTUA	National Technical University of Athens
NURBS	Non-Uniform Rational B-Spline
OpenMP	Open Multiprocessing
P2L	Particles to Local
P2M	Particles to Multipole
P2P	Particles to Particles
PCOpt	Parallel CFD & Optimization Unit
PDE	Partial Differential Equation
PE	Parametric Effectiveness
POSIX	Portable Operating System Interface for Unix
PSO	Particle Swarm Optimization
PUMA	Parallel-Unstructured-Multi-Row-Adjoint
QP	Quadratic Programming
r.h.s.	right-hand side
RANS	Reynolds-Averaged Navier-Stokes
RBF	Radial Basis Function
SA	Spalart-Allmaras
SIMD	Single Instruction stream, Multiple Data stream
SPAI	Sparse Approximate Inverse
SQMR	Symmetric Quasi-Minimal Residual
SQP	Sequential Quadratic Programming
SR1	Symmetric-Rank-one
STEP	STandard for the Exchange of Product model data
STL	Stereo Lithography interface format
SVD	Singular Values Decomposition
TE	Trailing Edge
TUB	Technical University of Berlin
w.r.t.	with respect to