TECHNISCHE UNIVERSITÄT MÜNCHEN

Ingenieurfakultät Bau Geo Umwelt

Lehrstuhl für Statik

Node-based parametrization for shape optimal design

Majid Hojjat

Vollständiger Abdruck der von der Ingenieurfakultät Bau Geo Umwelt der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:

Univ.-Prof. Dr.-Ing. habil. Fabian Duddeck

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Kai-Uwe Bletzinger
2. Univ.-Prof. Dr. rer. nat. Ernst Rank
3. Univ.-Prof. Dr.-Ing. Thomas Rung, Technische Universität Hamburg-Harburg

Die Dissertation wurde am 30.09.2014 bei der Technischen Universität München eingereicht und durch die Ingenieurfakultät Bau Geo Umwelt am 22.01.2015 angenommen.

Abstract

Advances in adjoint sensitivity analysis and high performance computing have moved the tendency in the field of shape optimization towards node-based methods due to their extremely rich geometrical design space. Moreover, acting directly on the discretization, the node-based technique has a great advantage over classical CAD-based methods which require an extra parametrization tool. However, a robust methodology to transform the nodal sensitivity values to a meaningful shape deformation for general industrial applications is still missing. This work tries to take a step towards overcoming this deficiency.

In this work, *Vertex Morphing*, a novel node-based parameterization method for shape optimization is developed. This method introduces in addition to the geometry field, a *design control* field in which the mathematical optimization problem is defined. The linear mapping relating the geometry and the control field is a kernel-based explicit filtering operator. In this way, the regularization of the ill-posed shape optimization problem is included in the definition of the design variables. Furthermore, the problem of mesh regularization is tackled by including a projection step in the same mapping operator, which enforces the mesh quality criterion through a dimension reduction. This consistent formulation of the smoothing closes the gap between the sensitivity filtering techniques and the standard optimization theory.

Vertex Morphing is developed to be integrated in mechanical software frameworks in order to assist the engineering design process. As such, the method is successfully implemented in an optimization workflow for Computational Fluid Dynamics problems. The workflow is tested on several real size engineering applications in fields of automotive and wind engineering. The method shows to be very efficient and robust even for highly complex geometries and large problems with up to 3.3 million design variables.

Contents

Chapter 1

Introduction

In this thesis, a parametrization methodology for shape optimal design is proposed. Most of the topics covered in this text are one-to-one usable for discrete shape optimization in any numerical simulation. Here however, the target application is shape optimization of wall-bounded flows with additional focus on wind-exposed structures. As inherited from the nature of numerical design optimization, there are two major lines to be investigated: the numerical analysis and the design betterment. The numerical analysis includes modeling of the physics of the problem as well as evaluation of the design sensitivity. Design betterment involves the geometrical and algorithmic actions required for transforming the design sensitivity information into an improved geometry. Despite the tight relation of the mentioned elements, they can be studied as two more or less separate modules forming the design optimization process. Even though the text contains a chapter about the numerical simulation, the emphasis is on the shape related aspects. The design optimization workflow presented at the end binds those two modules in a CFD shape optimization software, which is applicable to real engineering applications.

1.1 Node-based parametrization

Talking about "shape improvement" of an existing geometry, one should first answer the following question: "how should be the shape variation described?". The most intuitive answer might be: "in the same way that the shape itself is described!". This answer is the basis of Computer Aided Geometrical Design (CAGD) optimization, in which the CAD parameters used for the construction of the initial shape are selected as the design variables of the optimization problem. CAD-based optimization is a robust method which has shown success, and is widely used in various fields, such as structural optimization [11, 28, 64] and CFD optimization [55, 106, 110, 113] . The resulting shape always sustains the geometrical quality of the initial one, as it is built based on the same set of geometrical objects.

In numerical mechanics, the geometry is approximated by a set of discretization elements, and from the numerical viewpoint, after meshing, there is no connection between the discrete geometry and the "real" shape. Having that in mind, an alternative answer to the question arisen in the previous paragraph would be: "through the nodes of the discretized geometry.". This is what we refer to as the "node-based" method. In some texts, the term "parameter-free" is chosen for this shape representation method which is an inaccurate abbreviation for "free of CAD parameters". In node-based shape optimization position of the discretization nodes are the design variables.

Advantages of node-based parametrization can be summarized in two main aspects, i.e. geometrical and algorithmic. The node-based parameters provide the largest possible design space for a discrete model. Unlike in CAD and "morphing box" techniques, the design improvement contains all the shape modes and the free evolution of the form can lead to any non-intuitive shape. From the algorithmic point of view, the node-based method brings shape optimization closer to the "standard simulation process" in industrial applications, by removing the parametrization module from the workflow, as the parametrization is defined directly on the mesh used for the simulation. The need to include an additional parametrization software, and more importantly the extremely time consuming process of manual geometry paramet-

rization, prevents the use of automatic shape optimization in many industrial designs.

There are however some challenges to be tackled before using node-based parametrization, i.e. shape irregularity, mesh dependency, mesh distortion and large number of design variables. The appearance of shape irregularities is due to the large number of unknowns in the inverse problem of shape optimization which makes it from a mathematical viewpoint an ill-posed problem. Smoothing of the sensitivities is an effective means to regularize the shape morphing problem and is applied in structural shape optimization [33, 70], CFD shape optimal design [51, 55, 85, 115, 130, 131], and topology optimization [10, 15, 143]. A major part of developments in smooth shape morphing is done in the field of computer graphics [7, 69, 87]. Moreover, filtering is used also as a design tool to cut off the higher shape modes, independent of the discretization [12]. There are different formulations for the filtering operator. Explicit filters use a distance-based function [32, 51, 131] similar to signal processing. Implicit filters define a PDE, whose inverse discrete operator acts as the smoother. This PDE can be based on geometrical definitions such as curvature [57, 60, 86], or defined by an imaginary solid domain [5, 7, 125]. The numerical solution of the PDE (or in simple words inverting the PDE operator) is sensitive to the mesh quality and can limit the robustness in complex geometries.

If the shape variation is not limited to a small amount, the problem of mesh quality arises quickly, which is even more pronounced for complex geometries. Some suggest including a mesh regularization step in the optimization process [70, 129]. Alternatively the mesh quality criteria can be added to the optimization problem as constraint [114].

1.2 Adjoint optimization

Although zero order optimization methods remain the most commonly applied tool in engineering design, they reach their limit of usability and efficiency for moderately complex 3D geometries, considering the required number of design variables. In contrary, first order algorithms show a great performance and speed, even for many design variables [2, 137]. However, the sensitivity information, i.e. the gradient of the

response surface is required, which is usually a challenging task.

Design sensitivity is the rate of dependency of each system response (objective or constraint) to all the design parameters, and can be constructed as a matrix that has as many columns as the system responses and as many rows as the design variables. In order to evaluate the sensitivity matrix, one can first study "how does each design variable influence the physical model" and then based on that to calculate the sensitivity for "every system response". As an alternative one can first find out "how does the physical model influence each system response" and then to evaluate the sensitivity for "every design variable". The first approach is the direct, and the second one is the adjoint sensitivity analysis [8, 18, 58, 73, 134]. The computational effort in direct sensitivity analysis scales with the number of design variables and in adjoint by the number of responses. Therefore when there are more responses than design variables (fat sensitivity matrix) the direct method is preferred, and in the opposite situation (tall matrix) the adjoint one. Thus for node-based optimization only adjoint can be sensibly used.

Adjoint methods provide the sensitivity value for every node almost by the same cost as the original problem [30, 98]. They are relatively easy to be implemented for structural optimization and therefore they have been for many years an established technique [19, 45, 135]. In CFD, in contrast to structural mechanics, the state equations are not self-adjoint which makes evaluation of the adjoint fields more demanding. However, in the past two decades, there has been a rapid development of CFD adjoint sensitivities as well [41, 53, 56, 103, 123]. Adjoint methods have been developed for both compressible [57] and incompressible [58, 90, 93] problems. The applications aimed in this work, such as automotive and wind engineering require the adjoint sensitivity of the incompressible Navier-Stokes equations, whose solution can be more involved, because unlike compressible equations, the pressure is not present in the continuity equation through the density term.

One of the main difficulties in adjoint sensitivity analysis is that the adjoint inertial momentum for time-dependent equations is defined backwards in time. Therefore, the solution must start from the end-

time, marching back towards the start-time, which causes extensive memory requirements. Some remedies to overcome this issue can be found in [43].

For shape optimization of flexible structures interacting with a fluid flow, e.g. light weight structures in wind, the fluid-structure coupling shall be considered both in state as well as the sensitivity equations [1, 23, 74, 78, 80, 122].

1.3 Goal of thesis

Out of the many publications about filtered node-based shape optimization, there are only few which arise discussions on consistent merging of the filtering operator into the standard optimization technology [15, 70, 120]. Moreover, being driven by aeronautics, turbomachinery and hydraulic applications, most of the works done for node-based adjoint optimization deal with smooth surfaces with relatively low curvatures and more importantly very limited design variation. This is not at all the case in applications aimed here, e.g. automotive industry.

This work tries to overcome those deficiency by introducing the *Vertex Morphing* method, which is an explicit node-based parametrization for shape optimization. The suggested notation combines both filtering and mesh improvement operators into a "parametrization piece" in the chain rule of differentiation. In addition to the geometry, a control field is introduced and linked to the geometry through a linear map. Similar to the geometry itself, the control field is discretized and the optimization problem is formulated in the discrete control space which is the set of all control parameters. Having a coarser discretization for the design control field appears to be identical to the procedure of the subdivision surface technology to generate geometry [16, 65, 72, 144]. This definition establishes also a link between node-based and spline-based parametrization since what is introduced here as the control field is equivalent to the control polygon of splines.

In *Vertex Morphing*, normal and tangential surface directions are treated simultaneously and therefore large design variations can be

performed without any extra in-plane regularization. The method is designed to be applied on engineering problems and therefore is robust w.r.t. the shape complexity and can be used for non-smooth surfaces with kinks, mesh irregularities, etc. Combination of this explicit shape control method and adjoint sensitivity analysis forms an efficient and strong optimization tool which can be used for shape optimization of very large problems with millions of design variables.

1.4 Outline

The remaining of the text is organized as follows:

Chapter 2 introduces first the mathematical optimization problem and reviews different classes of design optimization w.r.t the choice of the solution strategy and the shape parametrization. Shape and mesh regularization as the key aspect of node-based shape optimization is also discussed.

In chapter 3, first the continuous 1D formulation of the *Vertex Morphing* method is shown, and then its discretized form is derived. At the end the 3D version of the method is presented which combines shape and mesh regularization operators. The method is positioned among the others and some properties are compared and discussed.

Chapter 4 is about the governing state equations and their physical interpretation. Since the sensitivity analysis for the shape optimization is closely related to the state equations, it is explained in the same chapter.

Chapter 5 includes algorithmic and implementation aspects of the optimization workflow and wraps up the single features into a software framework.

In chapter 6 the method is applied on several test cases ranging from small 2D examples to complex industrial ones. The selection of the

test cases is such that each of them rises a certain characteristics of such a type of optimization.

At the end, concluding remarks as well as suggestions for potential future work are mentioned in chapter 7.

Chapter 2

Shape optimization

Design optimization is a very common practice almost in every engineering process. The decision about “what is the best design within the feasible ones” can be taken in various ways. e.g. trial and error, parameter study, etc. "Design" itself can also have different meanings, e.g. material. Here we interpret "design" as the "shape" of the object to be improved. Shape optimal design is often classified into the following groups, even though their definitions can have overlaps in some cases: sizing, shape and topology optimization. In sizing optimization, the design parameters do not change the geometry, but design properties such as thickness of different elements. Shape optimization, which is the topic of discussion in this work, tries to find the optimal shape of an object while preserving the topology (or the main geometrical characteristics) of the initial design. Topology optimization goes one step further and aims in proposing the best topology and shape among all possible geometries.

This work is about shape optimization of structures and industrial parts that are either exposed to fluid flow, or are supposed to guide the fluid flow. The shape optimization problem is seen as the evolution of the shape in several improvement steps towards the optimum. Therefore,

the presented workflow can be put under the category of "nested analysis and design", NAND, in contrast to "simultaneous analysis and design", SAND [3]. This choice fulfills the modularity condition of the presented process in section 5.

In this chapter, first the mathematical optimization problem is stated. Different solution strategies are briefly named and the method of choice is presented. Next, the shape representation as the key feature in shape optimization is explained. The common shape parametrization techniques are reviewed in order to prepare the basis for introducing the proposed shape representation method, presented in chapter 3.

2.1 Optimization problem

The shape optimization problem has to be formulated in terms of a standard mathematical optimization problem in order to be solved. This step seems to be straightforward. However, in many engineering optimization applications a clear goal and definition of the problem is missing. Basically, the first step is to decide about an optimization scenario. Then based on a solution strategy, the already defined optimization problem will be solved. The target of the optimization is to improve one or some properties of a system, e.g. efficiency, weight, cost, etc. by suitable choice of one or some characteristics of the system, such as the shape, material, etc. The objective function represents the system property to be improved, while the design variables are the characteristics to be modified.

2.1.1 Formulation

The goal is to find the suitable value of the optimization variable $s \in \mathbb{V}$, for which the objective functional $J(s) \in \mathbb{R}$ has its minimum value. $\mathbb{V}$ is the design space and can have different definitions depending on the problem. There can be also some constraints limiting the choice of s directly or indirectly.

$$J(s) \rightarrow min \tag{2.1}$$

$$h_i(s) \leq 0, h_i \in \mathbb{R} \tag{2.2}$$

For an unconstrained optimization, the optimality condition follows

$$\nabla J(s^o) = 0, \tag{2.3}$$
$$|\nabla^2 J(s^o)| > 0. \tag{2.4}$$

The first condition indicates that s^o is a local extremum of the objective function, and the second one shows that this extremum is a minimum. If J is a convex function, there exist only one s^o, which is the solution of the optimization problem. Otherwise, the solution is the smallest local minimum.

In this work, the focus is on unconstrained optimization. However, since the state mechanical equations are treated as a constraint to the problem in adjoint sensitivity analysis, there are few sentences spent on explanation of constrained problems. In constrained optimization, the objective function and constraints are combined in a Lagrangian function $L \in \mathbb{R}$, so that the solution of the original constrained problem can be found by solving an unconstrained optimization over L. In order to do so, a suitable Lagrangian multiplier λ_i for each constraint has to be found.

$$L(s, \lambda_i) = J(s) + \lambda_i h_i(s) \tag{2.5}$$
$$\lambda_i > 0 \tag{2.6}$$

Note that at the optimum, independent of the values of λ_i, $L(s^o) = J(s^o)$, because active constraints are zero: $h_i(s^o) = 0$. L has saddle point properties and the optimization problem is reformulated as the saddle point search on L. The Dual function is defined as the minimum of the Lagrangian function w.r.t s, for a given λ_i:

$$D(\lambda_i) = \min_s L(s, \lambda_i) \tag{2.7}$$

To find the optimum, the Lagrangian multiplier λ^o which maximizes the Dual function has to be found. There exist various ways to find

the optimal solution (s^o, λ_i^o), most of which are based on staggered consequent minimization and maximization of the Lagrangian and the dual function respectively. In each step, either λ_i or s is kept fixed and the extremum of the other variable is found. Furthermore, the solution of the constrained optimization problem (s^o, λ_i^o) satisfies the Karush-Kuhn-Tucker (KKT) necessary conditions:

$$\frac{\partial L}{\partial s}\Big|_{s^o,\lambda^o} = \frac{\partial J}{\partial s}\Big|_{s^o} + \lambda_i^o \frac{\partial h_i}{\partial s}\Big|_{s^o} = 0 \tag{2.8}$$

$$\frac{\partial L}{\partial \lambda_i}\Big|_{s^o} = h_i(s^o) = 0 \tag{2.9}$$

Constrained solution strategies are not further discussed in this text.

2.1.2 Solution algorithm

Once the optimization problem is defined, the strategy to find the optimum (improved) s^o should be selected. Decision of the suitable strategy depends on several aspects such as the problem size, complexity of the governing equations, software availability, computation time, convexity of the response surface, etc. Solution methods can be classified based on the order of derivatives of the objective function required. In this work, a first order optimization is presented. However, for the sake of comparison, there is a short explanation of the other algorithms in the following paragraphs.

Zero order methods Zero order methods have perhaps the widest range of usage, particularly in engineering applications, due to their robustness and generality. The optimizer performs multiple evaluations of the objective function and no further information is required. Even a simple trial and error or parameter study (design of experiments, as the most common industrial optimization) can be called a zero order optimization. Though, there has been enormous amount of theoretical and algorithmic work on zero order methods [4, 42].

A group of zero order methods approximate the response surface by a function, and find the solution which minimizes this substitute function. This is usually done in multiple levels, in order to efficiently spend

the function evaluations close to the minimum while examining the full design space for potential regions where the minimum can lay. Thus, in each level, a better approximation of the response surface is available. Stochastic methods can help finding the best distribution of the evaluation points within the design space.

The other zero order solution strategy is based on the evolution theory of biological system, which states that the chance of survival for species depends on their fitness compared to the rest of the population. Engineering optimization has made use of this theory and different design alternatives are treated as a biological population in which some individuals are fitter than the others. In every evolution generation, the individual designs with higher fitness get the chance to have a larger contribution to the next generation population [4]. In such a way, those characteristics which lead to a higher fitness are preserved and those which exist in less fit individuals are gradually vanished.

Zero order methods can be applied in optimization of problems of any degree of complexity. They perform their search over the entire design space and aim in finding the global optimum without getting trapped in local optima. From the algorithmic view point, they are easily applicable to every engineering process. The main drawback is that usually many objective evaluations are required. Even though there are various improvements and enhancements proposed in order to increase efficiency of these methods, generally speaking, they are all computationally costly. Moreover, their usability is usually limited to a relatively small number of design parameters. As the number of optimization parameters increases, they get more and more computationally demanding and the convergence rate gets more poor. Therefore, in the context of the current work, in which a shape optimization with many design geometrical parameters is desired, use of zero order methods is not a choice.

Gradient based methods Gradient based methods use the derivative information of the objective function w.r.t. the optimization parameters ($\frac{\partial J}{\partial s}$ and $\frac{\partial^2 J}{\partial s^2}$ and ...) in order to find the optimum. The order of the gradient based method depends on the order of the derivative information used. So, a first order method uses only the the gradient $\frac{\partial J}{\partial s}$. The optimization is usually done in several consecutive iterations. In each iteration, an improvement update step is added to

the design variable. So the design variable in the $(n+1)$'th iteration s^{n+1}, is calculated based on the value of the previous step s^n and the update value δs^{n+1}:

$$s^{n+1} = s^n + \delta s^{n+1} \tag{2.10}$$

The update step can be decomposed to two components, the search direction $b^{n+1} \in \mathbb{V}$ and the step length $\alpha^{n+1} \in \mathbb{R}$.

$$\delta s^{n+1} = \alpha^{n+1} b^{n+1} \tag{2.11}$$

Once the search direction is found, the optimization problem is reduced to a one dimensional one, in which the optimal value of the step length has to be found. This one dimensional problem is called the "line search". In this work, the simplest first order algorithm, "steepest descent" is used. In steepest descent the search direction is the negative gradient vector:

$$b^{n+1} = -\nabla_s J(s^n) \Rightarrow \delta s^{n+1} = -\alpha^{n+1} \nabla_s J(s^n) \tag{2.12}$$

The Newton's method, as a second order optimization algorithm includes the second order derivative in calculation of the update step:

$$\delta s = -(\nabla_s^2 J(s^n))^{-1} \nabla_s J(s^n) \tag{2.13}$$

In order to demonstrate the performance of this optimization method, let us consider a 2D quadratic optimization problem.

$$\mathbb{V} = \mathbb{R}^2 \tag{2.14}$$

$$J = J(s_1, s_2) = a_1 s_1^2 + a_2 s_2^2 + a_3 s_1 s_2 \tag{2.15}$$

$$a_1, a_2, a_3 \geq 0 \tag{2.16}$$

if $a_3 = 0$ and $a_1 = a_2$, the response surface is a circular paraboloid with its minimum at the origin $(s_1, s_2) = (0, 0)$. For such a case the negative of the gradient vector always points toward the minimum:

$$-\nabla_s J = -(2a_1 s_1, 2a_2 s_2) = -2a_1(s1, s2) \tag{2.17}$$

Therefore, updating the design with a single steepest descent iteration with a step length of $\alpha = 1/(2a_1)$ brings us to the optimum in a single step (figure 2.1 top). Now, we assume that $a_1 < a_2$, and still $a_3 = 0$. This would stretch the paraboloid in the direction of s_1, and as plotted in the middle graph of figure 2.1, the steepest descent search direction of equation (2.17) does not point to the optimum anymore. Therefore one should perform multiple iterations in order to decrease the objective value. The Newton's methods however offers a better search direction:

$$-\nabla_s^2 J = -\begin{pmatrix} 2a_1 & a_3 \\ a_3 & 2a_2 \end{pmatrix} \tag{2.18}$$

$$a_3 = 0 \Rightarrow b = -\begin{pmatrix} 2a_1 & 0 \\ 0 & 2a_2 \end{pmatrix}^{-1} \begin{pmatrix} 2a_1 s_1 \\ 2a_2 s_2 \end{pmatrix} = -\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \tag{2.19}$$

This search direction brings us to the optimum with a single step. This is no surprise as the optimization problem is quadratic itself. One can still bring the good performance of the steepest descent method back to the optimization problem, by a suitable variable transformation. If instead of (s_1, s_2), we solve the problem for $(\tilde{s}_1, \tilde{s}_2)$ where $\tilde{s}_1 = (1/\sqrt{a_1})s_1$ and $\tilde{s}_2 = (1/\sqrt{a_2})s_2$. The objective function in terms of the transformed variables is

$$J(\tilde{s}) = \tilde{s}_1^2 + \tilde{s}_2^2 \tag{2.20}$$

Similar to the case in which $a_1 = a_2$, the transformed optimization can be solved by steepest descent in a single step. This transformation is called “scaling of the design variables”. Note that the adjusted search direction in the "conjugate gradient" method can also be seen as a re-parametrization which is done in an iterative manner.

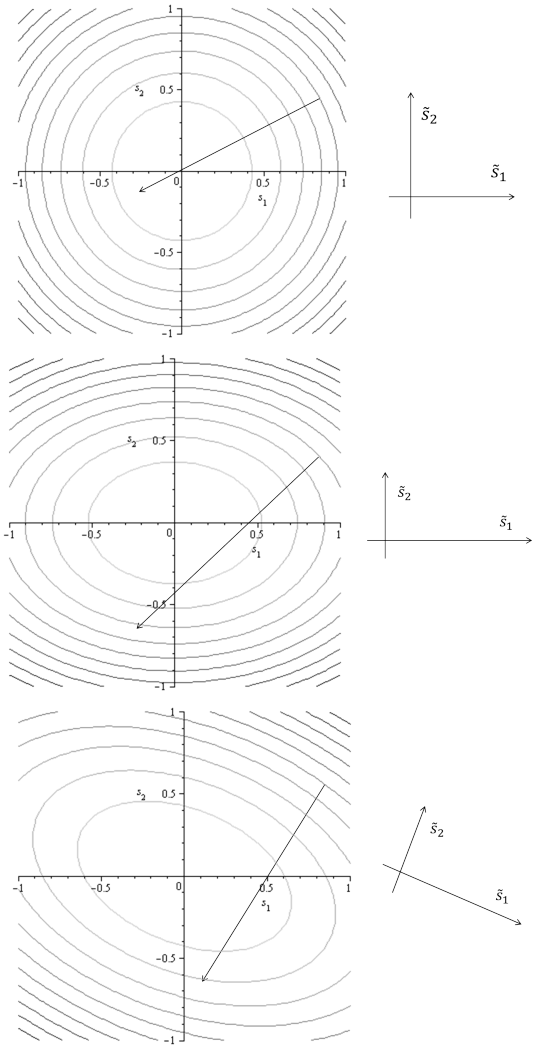


Figure 2.1: Gradient direction for a quadratic function with three sets of coefficients (left). Schematic sketch of the ideally "regularized" coordinate system for each case (right)

If $a_3 \neq 0$ only scaling the variable would not be enough, as the cross term $s_1 s_2$ would add skewness (rotation) to the response surface. In such a case the variable transformation should include cross terms as well. This transformation tries to bring the second order information to the problem by establishing a link between the variables. This is equivalent to using the Newton's method, which includes $\nabla^2_{\mathbf{s}} J$ in the update step. Once the update vector for the new variables $\delta\tilde{\mathbf{s}}$ is calculated, one should calculate the update vector of the original problem $\delta\mathbf{s}$ based on that:

$$\tilde{\mathbf{s}}_{(1\times 2)} = \mathbf{A}_{(2\times 2)}\mathbf{s}_{(1\times 2)} \Rightarrow \delta\mathbf{s} = \mathbf{A}^{-1}\delta\tilde{\mathbf{s}} \tag{2.21}$$

$\mathbf{A}$ is the transformation (scaling) matrix, and (2×2) indicates the dimensions of the matrix. In general, the gradient vector and the Hessian matrix of the objective function J w.r.t. $\tilde{\mathbf{s}}$ are the following:

$$\nabla_{\tilde{\mathbf{s}}} J = \mathbf{A}^{-T}\nabla_{\mathbf{s}} J \tag{2.22}$$

$$\nabla^2_{\tilde{\mathbf{s}}} J = \mathbf{A}^{-T}\nabla^2_{\mathbf{s}} J\mathbf{A}^{-1} \tag{2.23}$$

To find the best variable transformation for the steepest descent, one would need to have the Hessian $\nabla^2_{\mathbf{s}} J$. Therefore, the problem has to be reformulated such that by applying the variable transformation of equation (2.21), the steepest descent update step $\delta\mathbf{s}_{SD}$ with $\alpha = 1$ is the same as the Newton's step $\delta\mathbf{s}_N$:

$$\begin{aligned} \delta\mathbf{s}_{SD} &= \delta\mathbf{s}_N \\ \Rightarrow \mathbf{A}^{-1}\delta\tilde{\mathbf{s}} &= \delta\mathbf{s}_N \\ \Rightarrow -\mathbf{A}^{-1}\nabla_{\tilde{\mathbf{s}}} J &= -(\nabla^2_{\mathbf{s}} J)^{-1}\nabla_{\mathbf{s}} J \\ \Rightarrow -\mathbf{A}^{-1}\mathbf{A}^{-T}\nabla_{\mathbf{s}} J &= -(\nabla^2_{\mathbf{s}} J)^{-1}\nabla_{\mathbf{s}} J \\ \Rightarrow \mathbf{A}^T\mathbf{A} &= \nabla^2_{\mathbf{s}} J \end{aligned} \tag{2.24}$$

The transformation matrix $\mathbf{A}$ will be further related to the regularization and parametrization operator in sections 2.3 and 3, respectively.

As it will be discussed in section 3.1.2, when the Newton's method is applied, this variable transformation does not affect the search direction:

$$\begin{aligned}\mathbf{b}_N =& \mathbf{A}^{-1}\tilde{\mathbf{b}}_N \\ =& -\mathbf{A}^{-1}(\nabla^2_{\tilde{\mathbf{s}}}J)^{-1}\nabla_{\tilde{\mathbf{s}}}J \\ =& -\mathbf{A}^{-1}(\mathbf{A}^{-T}\nabla^2_{\mathbf{s}}J\mathbf{A}^{-1})^{-1}\mathbf{A}^{-T}\nabla_{\mathbf{s}}J\end{aligned} \tag{2.25}$$

The inverse of the Hessian matrix can be calculated based on equation (2.23) as:

$$(\nabla^2_{\tilde{\mathbf{s}}}J)^{-1} = \mathbf{A}(\nabla^2_{\mathbf{s}}J)^{-1}\mathbf{A}^T \tag{2.26}$$

Substituting (2.26) in equation (2.25) gives:

$$\begin{aligned}\mathbf{b}_N =& -\mathbf{A}^{-1}\mathbf{A}(\nabla^2_{\mathbf{s}}J)^{-1}\mathbf{A}^T\mathbf{A}^{-T}\nabla_{\mathbf{s}}J \\ =& -(\nabla^2_{\mathbf{s}}J)^{-1}\nabla_{\mathbf{s}}J,\end{aligned} \tag{2.27}$$

which is the original Newton's search direction.

2.2 Geometry parametrization

In order to perform shape optimization, we should describe the geometry of the object in terms of the optimization variables. This is because the mathematical problem is defined with abstract and dimensionless numbers without any geometrical interpretation. The transformation from the design variables s to the geometry x is called "shape parametrization" or "geometry representation". Note that the word "parametrization" does not necessarily mean that there is always a finite set of parameters describing the shape and as it will be seen in the following sections, s can be a continuous field. Here, we use a general notation for the shape parametrization operator:

$$x = \mathcal{A}(s), \mathcal{A} : \mathbb{V} \rightarrow \mathbb{W} \tag{2.28}$$

$x \in \mathbb{W}$ describes the geometry of the object and as before, $s \in \mathbb{V}$ is the design variable of the optimization problem. For example, a meaningful way to control (describe) the shape of a circle in a 2D plane is to parametrize its geometry by its shape and its center coordinates:

$$\begin{aligned} s = (s_1, s_2, s_3) = (c_x, c_y, r) \in \mathbb{V} = (\mathbb{R}, \mathbb{R}, \mathbb{R}^+) \\ x = \{(x_x, x_y, x_z) \in \mathbb{W} = \mathbb{R}^2 | (x_x - c_x)^2 + (x_y - c_y)^2 - r_2 = 0\} \end{aligned} \tag{2.29}$$

where c_x, c_y and r represent the x and y coordinates of the center and the radius of the circle respectively. Another example would be the description of a 2D curve by use of quadratic Bézier interpolation with three "control points" (P_1, P_2, P_3). Coordinates of the control points can be the design variables of the optimization problem:

$$\begin{aligned} s &= (s_1, s_2, ..., s_6) \\ &= (P_{1,x}, P_{1,y}, P_{2,x}, P_{2,y}, P_{3,x}, P_{3,y}), \quad s \in \mathbb{R}^6 \\ x &= x(\xi) = (x_x(\xi), x_y(\xi)), \end{aligned} \tag{2.30}$$

where

$$\begin{aligned} x_x(\xi) &= (1-\xi)^2 s_1 + 2(1-\xi)\xi s_3 + \xi^2 s_5 \\ x_y(\xi) &= (1-\xi)^2 s_2 + 2(1-\xi)\xi s_4 + \xi^2 s_6, \quad \xi \in \{0, 1\}. \end{aligned} \tag{2.31}$$

In this case the geometry is parametrized by 6 design variables as coordinates of the control points. There exists a "tracing parameter" or "local coordinate" ξ as a helping variable. If one wants to march on the entire shape (the 2D curve in this case), it is enough to vary ξ from 0 to 1. Moreover, all the coefficients multiplied by the design variables of equation (2.30) can be collected in a matrix operator $\mathbf{A}$:

$$x(\xi) = \mathbf{A}(\xi) s \tag{2.32}$$

where

$$\mathbf{A} = \begin{pmatrix} (1-\xi)^2 & 0 & 2(1-\xi)\xi & 0 & \xi^2 & 0 \\ 0 & (1-\xi)^2 & 0 & 2(1-\xi)\xi & 0 & \xi^2 \end{pmatrix} \tag{2.33}$$

In the sequence, the geometry representation methods are classified into two general categories, implicit and explicit. In the implicit representation, the geometry is related to the design variables through an implicit equation, with the following general form:

$$x = \{x | A(x, s) = 0\} \tag{2.34}$$

In explicit representation, the geometry is explicitly expressed as a function of the design variables, with the general form:

$$x = A(s) \tag{2.35}$$

Note that this classification is rather a notation-dependent definition, as basically the implicit functions can be written in explicit form and the other way around. Based on this definition, the example of the circle (equation (2.29)) is an implicit shape representation and the example of the 2D Bézier curve (equation (2.30)) is an explicit one. It is often the case that the A function of the equation (2.35) depends on the geometry x itself, and a linearization of the geometry representation function is required. In the following sections some key methods of shape parametrization are reviewed. More information about different parametrization techniques can be found in [36, 112].

2.2.1 Implicit representation

As any shape representation method, the goal is to describe and control the shape of an object in the 3D space with a set of variables. Usually it is enough if the boundary surface of the object as the interface between the region filled with the material and the void region is known. Many implicit representation methods introduce a scalar field in the 3D space, which has the value zero at the interface, and non-zero in the rest of the space. There are plenty of suggestions how to construct and vary this field. The "level set" method [91] and the "phase-field" method [132] are examples of implicit shape representation. In topology optimization the geometry is described implicitly, for instance by use of a material density field which indicates the material-filled or void regions in the

space.

Implicit shape representation is used successfully in numerical computation with complex interfaces and varying geometries. For example, in the “Volume of Fluid” method (VOF) [49] which deals with multiphase fluid simulations, the interface of different fluids is defined through a phase variable. If a free surface water calculation is performed, the phase variable has the value 1 in the regions filled with water and the value 0 for the air. The interface is where the inter-phase forces interact is the surface on which the phase variable switches from 1 to 0. The "immersed boundary" method is an other example of usage of implicit shape definition in numerical simulations [83, 100].

2.2.2 CAD

The word CAD stands for Computer Aided Design, which includes any computer-related technique for geometrical design. However, in the current document this term refers only to the surface (shape) representation methods which describe the shape (continuous or discrete) as a function of coordinates of a finite set of points in the space, called control points. Since every point on the surface inherits its location from some control points, changing the position of each control point would change the shape. CAD functions are defined often piecewise, which means that the “influence radius” of every control point is limited to a certain part of the surface.

In industrial computer aided design, Beziér and Spline curves are standard means and are very well established. NURBS (Non-uniform rational B-spline) as a generalization of this class of curves is used in many CAD software as well. In the parametric space, a NURBS (or spline or B-spline) patch is a rectangular surface with k rows and l columns of control points. The position of every surface point with surface coordinates $\bar{\xi} = (\xi_1, \xi_2)$ is generated as a weighted sum of the coordinates of the control points:

$$x(\xi_1, \xi_2) = A_{i,j}(\xi_1, \xi_2) P_{i,j} \tag{2.36}$$

where $P_{i,j}$ is the coordinate vector of the control point in the ith row and jth column. Similar to the shape functions in the Finite Element Method, the rational basis functions $A_{i,j}$ are the influence weights of the control points at every (ξ_1, ξ_2). These functions are calculated recursively and as a tensor product of the one dimensional base functions for each parametric coordinate. Depending on the type of curve (NURBS, etc.) the construction of the basis functions varies. Here, no deeper formulation of these patches is shown since it is not directly relevant to this work.

A widely used way to control and change the shape of objects is to use "morphing boxes". This technique is also referred to as "free-form deformation" [75, 117] and is been explored intensively, mainly due to its applicability in computer graphics [62]. The space around the optimization surface is spanned by some boxes, e.g. hexahedra. Again there are some basis functions defined for each box which interpolate the coordinates of the box corners inside the box (or in the neighboring boxes in case of higher order shape functions). For shape optimization, one can morph different parts of the geometry by deforming the morphing boxes built around it. Unlike NURBS (or in general CAD) functions which are used for both defining and morphing the geometry, morphing boxes are constructed around an already existing geometry in order to modify its shape. Construction of these boxes, as well as enforcing a desired level of geometrical continuity at the interface of two boxes are challenging and there exist morphing packages for shape optimization of CFD problems e.g. ANSA [94], Fluent, Hyperworks. It should be noted that construction of the morphing boxes for complex geometries requires a lot of manual work. Alternatively morphing boxes can be built as a Cartesian grid over a large domain containing the whole geometry. Cartesian morphing boxes provide a very poor parametrization and limited design freedom.

The limited number of CAD (or morphing box) design parameters allows for the use of zero order optimization algorithms, as well as direct sensitivity evaluation for gradient based algorithms e.g. finite differences [50]. The generated geometries inherit the properties of the initial design, with respect to the order of continuity, etc. as they are built by the same set of piecewise defined functions. From the

mathematical point of view, the optimization problem with CAD parametrization does not need regularization since the number of design parameters is usually relatively small.

In order to do CAD-based shape optimization, a CAD system (or software, or toolbox) should be included into the optimization process. The role of this system is to transform the movement of control points to the surface of the shape at every optimization step. Additionally, and in the case of gradient based algorithms, the control point sensitivities need to be calculated based on the surface sensitivity. Small numbers of design parameters lead to a restricted design space and the resulting shape has always the same geometrical characteristics as of the initial design. Furthermore, the CAD parametrization used for the geometrical design does not necessarily contain the shape modes which are influential to the optimization objective. For instance, in order to design a rectangular surface, it is enough to build a first order (linear) B-spline surface with four control points. Let us assume that while optimizing the shape, this surface needs to undergo some curvature in order to improve the objective function. This curvature cannot be made by changing the position of the four control points. In such a case one needs either to deal with the shape modes offered by the initial CAD model which can restrict the amount of improvement, or to modify and enrich the CAD functions e.g. by knot insertion, which requires a manual and probably non-trivial process. The same holds for the influence of the morphing boxes (their positions, their number, etc.) in the resulting shapes.

2.2.3 Node-based

An alternative approach for shape definition is to see the geometry as a continuous surface in the space which has to deform through a continuous deformation field in order to get to its optimal shape. Here, compared to previously introduced methods, the shape deformation is not described by means of a finite set of parameters, but by a continuous spatial field, the "shape deformation" field. The same idea applies for discrete geometries, which represent the shape by a finite set of points in the space. In that case the deformation field is defined on the same discrete space as of the geometry. In other words, the coordinates

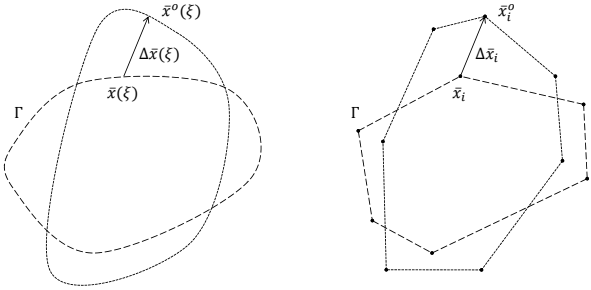


Figure 2.2: Node-based shape definition for continuous and discrete geometries

of each surface point in the discrete geometry are the design parameters (figure 2.2). The present work can be categorized as a node-based method, and therefore the formulation and the properties of node-based shape optimization are presented in the following paragraphs. Note that we use the term node-based even for the continuous case (as in figure 2.2), because every arbitrary point in the continuous surface can be seen as a "node" as well.

We introduce the following surface curvilinear coordinates system with $\bar{g}_1$ and $\bar{g}_2$ being its base vectors in the 3D case(figure 2.3).

$$\bar{g}_1 = \frac{d\bar{x}}{d\xi_1} \ , \ \bar{g}_2 = \frac{d\bar{x}}{d\xi_2} \tag{2.37}$$

Here, vectors defined in 2D or 3D space are noted with an upper bar. In every optimization iteration, the design surface undergoes a deformation towards its optimal shape. In such a way, the optimization iterations can be seen as a pseudo time dimension through which the design evolves. In other words, every surface particle travels in the optimization pseudo time on its $\eta(\xi_1, \xi_2)$ trajectory curve. The normal to surface direction $\bar{g}_3 = \bar{g}_1 \times \bar{g}_2$ is the “shape relevant” direction because moving the surface in that direction changes the shape. Neglecting

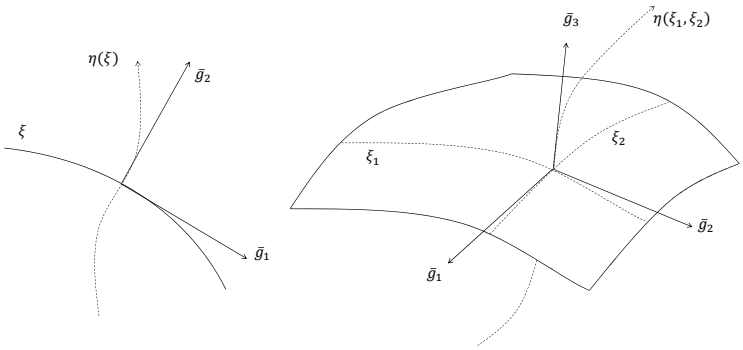


Figure 2.3: 2D and 3D surface curvilinear coordinates

the discretization (linearization) error, the tangential directions, ξ_1, ξ_2 have no influence on the shape, but they affect the surface mesh. Using the surface curvilinear base vectors instead of the Cartesian coordinate system, the geometry at every surface point can be decomposed into a normal and two tangential components:

$$\bar{x} = x_x \bar{i} + x_y \bar{j} + x_z \bar{k} = x_{t_1} \frac{\bar{g}_1}{|\bar{g}_1|} + x_{t_2} \frac{\bar{g}_2}{|\bar{g}_2|} + x_n \frac{\bar{g}_3}{|\bar{g}_3|} \tag{2.38}$$

$\bar{i}$, $\bar{j}$ and $\bar{k}$ are the unit base vectors of the 3D Cartesian coordinate system in x, y and z direction, respectively. In this work, the calculation of the surface coordinate system for discrete surfaces is done through a piece-wise linear interpolation of the nodal coordinates.

Node-based methods offer the largest design space possible for the optimization of a discrete geometry, since the design space is as large as the geometry space. However, the very large number of design variables makes the solution of the optimization problem more demanding. For instant, zero order methods are almost never applicable, as their computational cost (number of function evaluations) increases exponentially by the number of design variables. This motivates the use of gradient-based optimization. Evaluation of the full gradient vector for a problem with many variables can be challenging as well. The required

number of function evaluations for calculation of the gradient vector by use of a direct method such as finite difference is proportional to the number of design parameters . As the result, sensitivity calculation in node-based optimization is almost always done by adjoint methods.

Employing the position of discretization points as optimization design variables not only broadens the space of possible designs, but also brings practical and procedural advantage to the optimization process. This is because the discretized problem which is used for numerical calculation can be directly used for shape optimization as well. Therefore, there is no need for a CAD tool to be integrated in the optimization loop. Similar to topology optimization, the optimal design is not influenced by the way the initial design is produced, and hence the designer does not need to predict the final shape while setting up the optimization.

Despite all mentioned advantages, node-based methods suffer from mesh dependency, mesh irregularity and non-smooth shape derivatives [17, 70, 86]. The reason is that the shape optimization problem is not well-posed (section 2.3) and therefore the problem needs to be "regularized". The next section discusses the regularization more in depth. Another shortcoming of node-based shape representation is that the optimal shape, as the solution of the problem, is available only as a discrete surface. This causes difficulties for inclusion of such methods in industrial design and production chains, because there is usually a CAD definition required. Building a CAD geometry based on the optimal discrete surface should be done either manually, or by some curve fitting techniques. Note that this problem exists in geometries produced by topology optimization, morphing boxes or any other technique with discrete shape functions as well.

2.3 Regularization

It is known from statistics and mathematics that "ill-posed" problems need to be regularized. Inverse problems are typically ill-posed, particularly when they have many variables. In mechanics, an inverse problem is the process of finding the system characteristics, for which a given response is to be expected. Optimization problems, including shape optimization are examples of inverse problems in engineering.

There exists various articles and books on the mathematical analysis of regularization of ill-posed problems w.r.t. to uniqueness, condition of stability, etc. Here we present the regularization process from a more practical and engineering point of view with the focus on node-based shape optimization problems. Usually CAD parametrized shapes do not require regularization, because of their limited number of parameters. In a node-based optimization problem which deals with many design parameters an additional regularization treatment is needed [85].

Let us assume the following abstract inverse problem. The operator G maps $x \in \mathbb{V}$ onto $y \in \mathbb{W}$, where both $\mathbb{V}$ and $\mathbb{W}$ are Hilbert spaces.

$$Gx = y \tag{2.39}$$

According to Jacques Hadamard [44], equation (2.39) is well-posed if:

- there exists a solution x for every y
- the solution is unique
- the solution is stable (G^{-1} is continuous)

The equation is ill-posed (unstable) if it is not well-posed. Regularization methods try to stabilize the solution of an ill-posed equation system by approximating the inverse operator G^{-1} by a regularized operator R_ϵ. ϵ is called the regularization parameter and in this work is referred to as regularization intensity. The response y is decomposed to the response of the regularized solution $G(x^r)$ and a noise y^n.

$$y = Gx^r + y^n \tag{2.40}$$

The regularization operator should be defined such that for $y \to 0$, one can find a regularization intensity ϵ for which

$$x^r = R_\epsilon y \to x \tag{2.41}$$

A natural way to solve an inverse problem is to formulate it as a minimization problem. An example is the least square fit for a series of given statistical data which uses the minimization of the Euclidean distance. The solution of equation (2.39) by minimization of an arbitrary norm of the residual would be:

$$\underset{x}{argmin}\|Gx - y\| \tag{2.42}$$

Note that transforming the inverse problem to an optimization problem does not make it well-posed. It is also known from literature that instabilities exist very often in least square fitting. In general, there are two possibilities for regularizing equation (2.42): reducing the dimensions of the problem to a "small" finite number of variables (section 2.2.2), or adding a penalizing term to the equation. Having in mind that here the goal is to regularize the node-based shape optimization problem (without applying extra parametrization), we naturally choose the second option.

Regularization methods are categorized by the type of the norm they use in order to penalize the minimization problem of equation (2.42). A well-known regularization method is Tikhonov regularization based on the L^2 norm. This method is known as ridge regression in statistics and is commonly used in fields such as image filtering, etc. (e.g. [25]). In Tikhonov method, the second norm of an operator Λ applied on the variable x is added to the minimization problem (equation (2.42)). The solution of the Tikhonov regularized problem in its general form follows this formula:

$$\underset{x}{argmin}(\|Gx - y\|^2 + \|\Lambda x\|^2) \tag{2.43}$$

Construction of a suitable Tikhonov operator Λ can be complicated, and therefore in a simplified notation, it is replaced by the identity operator I, and the effect of Λ is compressed in a single scalar value, the regularization parameter ϵ.

$$\underset{x}{argmin}(\|Gx - y\|^2 + \epsilon\|x\|^2) \tag{2.44}$$

This notation is in fact a constrained minimization problem, which is formulated by use of the Lagrangian multiplier ϵ. ϵ controls how intensively the solution should be "regularized". So, it cannot be chosen very small, because then the regularizing effect is very small and the problem remains ill-posed. On the other hand, when ϵ is selected to be very large, the objective function to be minimized is dominated by the regularization term. In the limit $\epsilon \to \infty$, the problem reduces to minimization of $\|x\|^2$ with the trivial solution $x = 0$. It can be shown that when G is compact 2.44 has a unique solution [63]:

$$x = R_\epsilon y = (G^T G + \epsilon I)^{-1} G^T y \tag{2.45}$$

where R_ϵ is the regularized inverse operator with the regularization intensity ϵ. The T superscript indicates the adjoint (transpose) operator. R_ϵ smooths the solution by penalizing large values of the norm $\epsilon\|x\|$. The role of the regularization term can be understood more clearly by Singular Value Decomposition SVD of the Tikhonov operator. Here the finite dimensional inverse problem is assumed:

$$\mathbf{Gx} = \mathbf{y}, \mathbf{G} = \mathbf{U\Sigma V}^T \tag{2.46}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices ($\mathbf{UU}^T = \mathbf{I}$) and $\mathbf{\Sigma}$ is a diagonal matrix. Known from SVD, for a square matrix, the influence of operator $\mathbf{G}$ is decomposed into three sequential operations: a rotation by $\mathbf{U}$, a scaling by $\mathbf{\Sigma}$ and a second rotation by $\mathbf{V}$. Another geometrical interpretation of this decomposition is the coordinate transformation from a basis with base vectors defined by the rows of $\mathbf{U}$ (reference basis) to a basis whose base vectors are defined by the rows of $\mathbf{V}$ (target basis). The entities of the diagonal matrix $\mathbf{\Sigma}$ are responsible for scaling between the two spaces. Note that the basis are orthogonal since $\mathbf{U}$ and $\mathbf{V}$ are unitary. Substituting the decomposition in the Tikhonov regularized operator leads to:

$$\begin{aligned} \mathbf{R}_\epsilon &= (\mathbf{G}^T\mathbf{G} + \epsilon\mathbf{I})^{-1}\mathbf{G}^T \\ &= (\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T + \epsilon\mathbf{V}\mathbf{I}\mathbf{V}^T)^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T \\ &= \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma} + \epsilon\mathbf{I})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{V}\mathbf{\Sigma}_\epsilon^{-1}\mathbf{U}^T \end{aligned} \tag{2.47}$$

The non-regularized inverse operator would have the following single value decomposition:

$$\mathbf{R} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T \tag{2.48}$$

Comparing equations (2.47) and (2.48), the regularized inverse operator $\mathbf{R}_\epsilon$ has identical rotation operators $\mathbf{U}$ and $\mathbf{V}$ as the original problem. The only difference is that the entities of $\mathbf{\Sigma}^{-1}$ are scaled by a factor as the following:

$$\mathbf{\Sigma}_{\epsilon ij}^{-1} = \left(\frac{\mathbf{\Sigma}_{ij}^2}{\mathbf{\Sigma}_{ij}^2 + \epsilon}\right)\mathbf{\Sigma}_{ij}^{-1} \tag{2.49}$$

Scaling the entities of the diagonal matrix $\mathbf{\Sigma}$ is equivalent to the variable transformation discussed in section 2.1.2. There, the optimization problem was regularized by assuming non-uniform weighting factors for different dimensions of the domain (design space) $\mathbb{V}$. This process formulates the inverse problem on a new domain (say $\mathbb{V}_{reg}$ instead of the original domain, $\mathbb{V}$) so that it is well-posed. Note that a main source of instability in the solution of an ill-posed problem is that some coefficients of the operator $\mathbf{\Sigma}$ have extremely small magnitudes (one would say because $\mathbf{G}$ is badly conditioned). This disturbs convergence of the numerical scheme in prediction of the unknown vector $\mathbf{x} = \mathbf{G}^{-1}\mathbf{y}$ as $\mathbf{\Sigma}$ would contain entities which are orders of magnitudes greater than the others. The scaling factor of equation (2.49) which results from the penalty term $\epsilon\|\mathbf{x}\|^2$ in equation (2.44) has a value close to one when the diagonal term has a large absolute value compared to the regularization constant $\mathbf{\Sigma}_{ij}^2 \gg \epsilon$. But when $\mathbf{\Sigma}_{ij}^2$ is very small, adding the non-zero ϵ to the denominator prevents $\mathbf{\Sigma}_\epsilon$ to have very large entities. One can clearly see in equation (2.49) how the regularization intensity ϵ modifies the original problem to assure stability. The question of

what is the ideal ϵ depends on the specific problem. In most of the engineering applications of regularization it is not possible to calculate the optimal ϵ and therefore it is treated as a solution parameter (input value) evaluated by trial and error and the experience from similar problems.

2.3.1 Shape regularization

Similar to many other inverse problems, Tikhonov type regularization based on equation (2.43) is used for optimization problems as well. The cost function of the problem is penalized by the regularization term which is usually presented as a norm:

$$J^r(x) = J(x) + \epsilon\|\Lambda x\|^2 \tag{2.50}$$

In shape optimization the unknown variable is the geometry x. The gradient is calculated by differentiation of the original cost function together with the regularization term. In [85] it is stated that for a second order problem anything related to the second derivative would likely work. The Tikhonov shape regularization (penalty) term $\|\Lambda x\|^2$ can be constructed based on geometrical meanings, e.g. curvature. In gradient-based shape optimization, an alternative way is to regularize the problem by filtering directly the shape derivatives (spatial gradient vector) instead of adding the penalty term into the objective functional. An example of this type, is the regularization method proposed by Jameson [57] and later on studies by many others, e.g. [86] and [116]. This method suggests penalizing high curvatures (large second spatial derivatives) of the shape derivative which can be directly included into the gradient calculation as a preconditioning operator. This operator enforces the regularity (smoothness) by letting the shape derivative be the solution of an elliptic equation, with a given coefficient ϵ. This type of filters are "implicit" regularization operators as they act directly on the gradient operator, and not on its inverse. This method performs very well and is widely used, particularly in aerodynamic shape optimal design. This regularization is further discussed in section 3.1.5. In an other implicit formulation, [125] and [5], assume a pseudo elasticity for the optimization surface in order to regularize the optimization problem. In this case, the regularization operator would be the stiffness matrix of

the 2D structure (i.e. shell) describing the design surface. There is no principal difference between this method and the later, as the stiffness of the structure is nothing but a geometrical derivative (curvature in case of shells in bending) multiplied by the material tensor. Since the structure is fictional, the material quantities (elasticity modulus, etc.) are to be set as regularization parameters, equivalent to ϵ. In [114] the regularity is imposed as a surface smoothness constraint to the optimization problem. Despite the good control on the mesh quality, the resulting constrained problem has a higher complexity compared to the original unconstrained problem, as it is the case for any optimization problem with inequality constraints.

Alternatively, one can apply the regularization operator directly on the inverse operator G^{-1} of equation (2.39) or equivalently on the right hand side, y. This type of operator is referred to as "explicit" in contrast to the described "implicit" operators. Clearly the inverse of an explicit regularization operator can be used as an implicit one. Explicit filters are used in statistics e.g. for estimation of probability density function. As one of the most famous techniques, Parzen window [27, 37, 99] uses a kernel function in order to estimate the unknown "smooth" distribution, based on a given "non-smooth" distribution (or data set, or function). Parzen window is a non-parametric technique and therefore there is no assumption about the unknown variable following a particular distribution. In [142] it is proven that Parzen window (explicit smoother) is an approximation of the Least Squares regularization (implicit smoother). Similar to implicit filtering, one can use explicit filters for smoothing the sensitivity field. The suggested method of this work is a kernel-based regularization method based on Parzen window, and therefore more discussion is followed in chapter 3.

2.3.2 Mesh regularization

As mentioned in 2.2.3, neglecting the discretization error, only the normal-to-surface movement of the nodes can change the geometry. Therefore usually the normal-to-surface coordinate of each point $\bar{g}_3$ is considered as the design variable. The regularization approaches mentioned in the previous section were also meant to provide a stable shape optimization, independent of the surface discretization. However, modifying the coordinates of the nodes in the normal direction affects

the surface mesh. This can be a limiting factor in varying the design, as the discretization quality is necessary for a stable, and accurate numerical scheme. Therefore, the surface discretization shall be regularized within the optimization as well. This section first explores the problem of mesh distortion in shape evolution and then reviews the common treatments. The proposed remedy is presented in chapter 3.

In this section, for the sake of simplicity, the shape optimization of a curve in the 2D space is studied (figure 2.3). As the surface deforms within the optimization loop, its discretization elements deform as well. This can easily lead to badly distributed mesh density, distorted and even overlapping elements. The attempt to bring the surface discretization back to its initial situation is often called *in-plane* regularization [57, 128].

The major source of mesh distortion in shape optimization is the change of (determinant of) the Jacobian $\frac{d\bar{x}}{d\xi}$. A non-uniform change of the Jacobian leads to a non-uniform change in the mesh density. For the case of a surface in the 3D space, in addition to the mesh density the element shapes vary as well, since the variation of Jacobian takes place in both directions.

Generally speaking, moving the curve towards its positive (negative) curvature direction makes $\frac{d\bar{x}}{d\xi}$ smaller (larger). This phenomenon is observed in shape derivative calculation of surface defined objectives as well, where the normal vector ($\bar{g}_2$ in 2D and $\bar{g}_3$ for 3D) has to be differentiated [115]. Figure 2.4 shows shrinking of the shape, as it is moved uniformly in the direction of positive curvature. This shrinkage depends on the curvature κ and the step length $|\Delta\bar{x}|$. The Jacobian in this case is the following:

$$\frac{d\bar{x}^{k+1}}{d\xi} = \frac{d\bar{x}^{k}}{d\xi}\left(1 - \frac{|\Delta x_n|}{R^k}\right) \ , \ R^k = \frac{1}{|\bar{\kappa}|} = \left|\frac{d\bar{g}^k}{d\xi}\right|^{-1} \tag{2.51}$$

where $\bar{x}^k$ and $\bar{x}^{k+1}$ correspond to the geometry at pseudo time instances

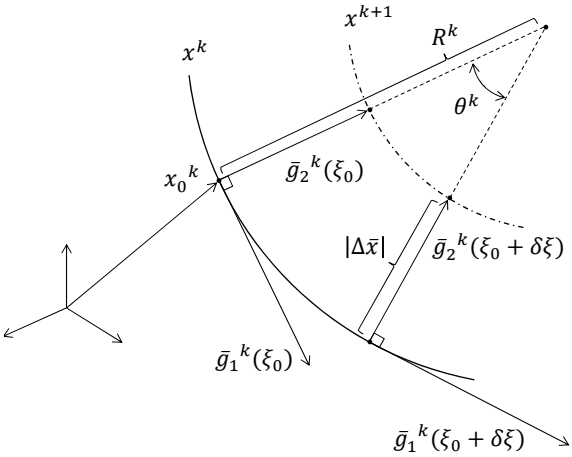


Figure 2.4: Demonstration of surface mesh density variation within an update step

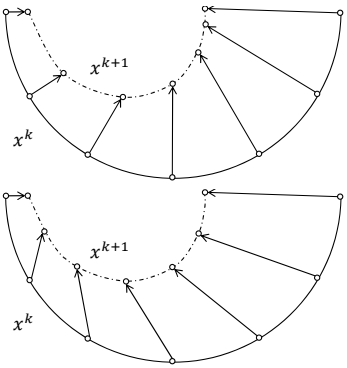


Figure 2.5: Top: change in the mesh density by a non-uniform update length, bottom: considering the in-plane direction can maintain the mesh quality

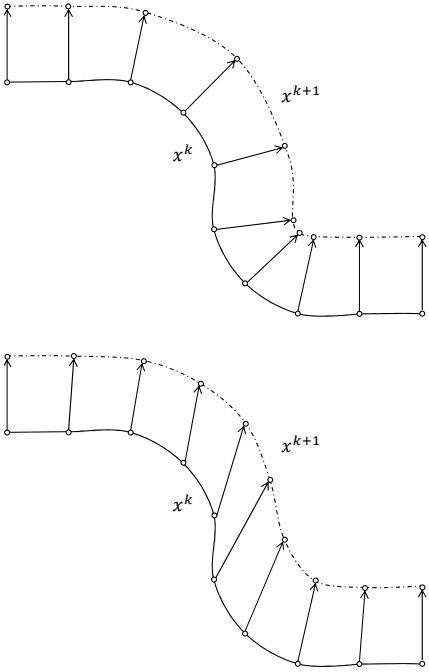


Figure 2.6: Top: change in the mesh density in regions of varying curvature, bottom: considering the in-plane direction can maintain the mesh quality

k and $k+1$. As curvature and the step length are the parameters changing the determinant, there exist two scenarios in which the mesh density (and the element shape) is deteriorated. First, when the shape update takes place at regions with varying curvature, and second, when non-uniform shape updates are applied on curved surfaces. Simple sketches of figures 2.5 and 2.6 visualize the mentioned phenomena. The issue of surface mesh quality urges a targeted motion of the surface mesh in the tangential directions, so that the mesh quality is preserved, but the original shape optimization problem is not affected. Preserving the mesh quality after shape variation can be regarded as an additional optimization problem to be solved in parallel to the main shape optimization problem. Therefore, one can enhance the

optimization problem by an augmented cost function J^a which includes the mesh quality term J^m as a function of the tangential coordinates. As mentioned before, the original optimization functional depends only on the normal component. Therefore:

$$J^a\left(x_{t_1}, x_{t_2}, x_n\right) = J\left(x_n\right) + J^m\left(x_{t_1}, x_{t_2}\right), \tag{2.52}$$

$$\frac{\partial J^m}{\partial x_n} = 0 \Rightarrow \frac{\partial J^a}{\partial x_n} = \frac{\partial J}{\partial x_n}. \tag{2.53}$$

This is a penalty notation similar to the one used in Tikhonov regularized cost function (equation (2.50)). However, the big difference here is that the original cost function and the mesh cost function are defined on different domains. As the result, two different problems for the mesh and the optimality need to be solved. These solutions are done either simultaneously or in a staggered manner. Note that for large shape variations the linearity assumption of the surface coordinate system is not accurate anymore and the coupling between x_n and x_t shall be considered. A meaningful example for the regularity indicator J^m can be the relative mesh density.

In a staggered approach, after each update step (or few steps) an in-plane mesh regularization step is performed [33, 70, 127, 128, 143]. The aim of mesh regularization is to modify the surface coordinates of the points $\bar{\xi} = (\xi_1, \xi_2)$, such that a mesh quality functional formulated in the physical space is minimized. Even though sequential use of in- and out-plane regularization forms a strong shape update strategy, separate treatment of normal and tangential directions requires an accurate definition of the surface normal, which is generally not available in discrete free-form geometries. This is an important drawback of this method, as the approximated tangential coordinates are highly sensitive to noisiness or curvature of the discrete surface. This limits the robustness of this staggered approach, particularly in complex geometries with sharp corners, etc. Moreover, the in-plane regularization adds to the computational cost, particularly if a strong regularization with implicit mesh correction procedure is applied [46, 127, 128].

As an alternative, one can define both functions J and J^m on the complete set of variables (normal and tangential directions), and to apply the J^m as an inequality constraint to the minimization of the original cost function J, similar to the mentioned process for shape regularization in section 2.3.1. This formulation triples the size of the optimization problem and makes it more complex due to the additional constraint. However the method has shown success in 3D structural optimization [114].

The proposed method in chapter 3 performs the mesh regularization step simultaneously with the shape optimal design, since by construction the design variables take into account the mesh regularity condition. The update direction includes a tangential direction which provides an almost constant mesh density.

Chapter 3

Vertex Morphing

In this chapter, a novel node-based formulation for shape optimization, *The Vertex Morphing Method*, is presented. The essence of the method is the filtering of the sensitivity field as well as the shape update vector by help of a suitable parametrization. Unlike most of the previous works on node-based shape optimization, the filtering (regularization) operations are derived consistently from the chain rule of differentiation. This formulation is nothing but an elaborate variable transformation (explained in section 2.1.2) enhanced with a suitable dimensional reduction for mesh quality regularization. In terms of the methods described in section 2.3, the regularization approach is an explicit kernel-based filtering (Parzen) which aims in suitable scaling of the parameters in each optimization iteration in order to avoid numerical instability.

The method is explained first in a one dimensional formulation, so that the complexity of the expressions does not distract the attention from the main idea. Moreover, the mesh quality problem does not appear in one dimensional shape optimization which allows for separate analysis of the shape regularization. After that, some characteristics of the method are presented and the proposed method is compared to other

alternatives, w.r.t. the filtering, etc. At the end, the three dimensional formulation is presented, in which the mesh regularization operator is merged into the *Vertex Morphing* parametrization.

3.1 1D Vertex Morphing

The goal is to find the optimal design surface field $x(\xi) \in \mathbb{V}$ and $V : (-1, 1) \to \mathbb{R}$, for which the state equation $F(x, w) = 0$ is satisfied and the objective function $J(x, w)$ is minimized. ξ is the surface coordinate and w is the state variable. The surface is controlled by a field $s(\xi), s \in \mathbb{V}$ (figure 3.1) through a functional, $\mathcal{A}(s, \xi) = x(\xi), \mathcal{A} : \mathbb{V} \to \mathbb{V}$. So the surface geometry at $\xi = \xi_0$ is found as,

$$x(\xi_0) = \mathcal{A}(s, \xi_0). \tag{3.1}$$

Here, we assume that $\mathcal{A}$ is linear and formulate it as the inner product (kernel):

$$\mathcal{A}(s, \xi_0) = \int_{-1}^{1} A(\xi_0, \xi) s(\xi) d\xi, A : \bar{\mathbb{R}}^2 \to \mathbb{R}. \tag{3.2}$$

Having in mind this relation, the geometry and its variation at $\xi = \xi_0$ are:

$$x(\xi_0) = \int_{-1}^{1} A(\xi_0, \xi) s(\xi) d\xi \tag{3.3}$$

$$\delta x(\xi_0) = \int_{-1}^{1} A(\xi_0, \xi) \delta s(\xi) d\xi \tag{3.4}$$

In fact $A(\xi_0, \xi)$ is the derivative of $x(\xi_0)$ w.r.t. $s(\xi)$, because,

$$\delta x(\xi_0) = \int_{-1}^{1} \frac{dx(\xi_0)}{ds(\xi)} \delta s(\xi) d\xi \tag{3.5}$$

and comparing equation (3.4) and (3.6), it is clear that

$$A(\xi_0, \xi) = \frac{dx(\xi_0)}{ds(\xi)} \tag{3.6}$$

For gradient based shape optimization, one needs to calculate the sensitivity of the objective w.r.t. the design:

$$\frac{dJ}{ds} = \frac{\partial J}{\partial s} + \int_{-1}^{1} \frac{\partial J}{\partial x(\xi)} \frac{dx(\xi)}{ds} d\xi. \tag{3.7}$$

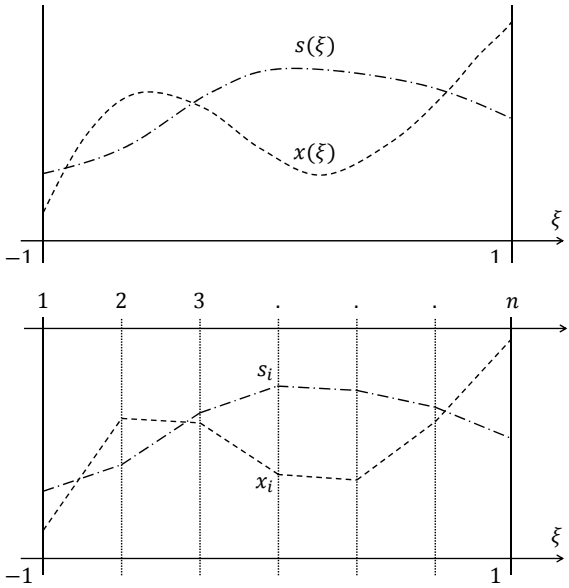


Figure 3.1: Top: the geometry x and design control field s as functions of surface coordinates ξ, bottom: the discrete geometry vector $\mathbf{x}$ and the design vector $\mathbf{s}$

As s is the shape control field, $\dfrac{\partial J}{\partial s} = 0$. $\dfrac{\partial J}{\partial x(\xi)}$ is the sensitivity of the objective to the shape variation. Considering equation (3.7), the sensitivity of the objective w.r.t. the design at $\xi = \xi_0$ can be written as,

$$\frac{dJ}{ds(\xi_0)} = \int_{-1}^{1} \frac{\partial J}{\partial x(\xi)} \frac{dx(\xi)}{ds(\xi_0)} d\xi. \tag{3.8}$$

Replacing equation (3.6) in (3.8) the sensitivity is calculated as

$$\frac{dJ}{ds(\xi_0)} = \int_{-1}^{1} \frac{\partial J}{\partial x(\xi)} A(\xi, \xi_0) d\xi. \tag{3.9}$$

There are several possibilities to interpret the procedures of equation (3.4) and (3.8) together with the kernel function A. The most obvious is to see it as a smoothing operation for stabilizing the nu-

merical optimization (section 2.3). Also, one can see it as a filtering procedure in order to suppress higher oscillating geometrical modes when generating the geometry from the control field. That is the point of view which is favored here by *Vertex Morphing* and also by others as e.g. [10] following the idea of dealing with non-convex optimization problems. Then the filtering operation is used to steer the optimizer to local minima which are characterized by the selected modes of the applied filter. Finally, when the control field s and the geometry x are of the same dimension, equations (3.4) and (3.8) represent nothing more than a re-parametrization of the original problem by a variable transformation from x to s, in which the inverse optimization problem (more precisely the gradient operator) is better conditioned. That view tells that by the filtering operations presented in equations (3.4) and (3.8) all optimal solutions (local and global) will not be modified and, indeed, filtering (or smoothing) is a procedural mean as indicated just before.

3.1.1 Discretization

In numerical shape optimization, usually one deals with discretized geometries. Therefore, we seek the vector of nodal coordinates $\mathbf{x} = [x_1, x_2, ..., x_n], \mathbf{x} \in \mathbb{R}^n$ which defines the discretized design surface (figure 3.1). Here, the control field has the same discretization as the geometry. So there exist n design parameters $\mathbf{s}$ as well. Also, one can choose different discretizations for x and s. For instance, using a coarse grid for s is equivalent to having a CAD control polygon defined by some few control nodes. This will be later presented in this chapter through an example. Again, $J(\mathbf{x}, \mathbf{w}, \mathbf{s})$ has to be minimized with the discrete state equation as equality constraint $F(\mathbf{x}, \mathbf{w}, \mathbf{s}) = 0$, where $\mathbf{w}$ is the vector of state variables. Equivalent to equation (3.2), applying the A operator function to a discretized control field, the operator matrix $\mathbf{A}$ is defined which linearly maps $\mathbf{s}$ onto $\mathbf{x}$:

$$x(\xi_i) = x_i = A_{ij} s_j \tag{3.10}$$

$$\delta x_i = A_{ij} \delta s_j \tag{3.11}$$

$$\frac{dJ}{ds_i} = \frac{dJ}{dx_j}\frac{dx_j}{ds_i} = A_{ji}\frac{dJ}{dx_j} = \mathbf{A}^T\mathbf{b}. \tag{3.12}$$

The transpose of the operator matrix $\mathbf{A}^T$, maps the nodal sensitivities $b_j = \frac{dJ}{dx_j}$ back onto the control parameters. To conclude, $\mathbf{A}$ is used twice. In the context of the filtering idea, first, for forward filtering of the control parameters $\mathbf{s}$ or their variations $\delta\mathbf{s}$ to generate the geometry $\mathbf{x}$ or its variation $\delta\mathbf{x}$

$$\delta\mathbf{x} = \mathbf{A}\delta\mathbf{s} \tag{3.13}$$

and, second, for backward filtering of the nodal sensitivities from the geometry to the design control using the transpose of $\mathbf{A}$:

$$\frac{dJ}{d\mathbf{s}} = \mathbf{A}^\mathbf{T}\mathbf{b} \tag{3.14}$$

Some details about the construction of $\mathbf{A}$ can be found in [12]. It appears that the operator matrix $\mathbf{A}$ is symmetric for most of the reasonable kernel functions which explains the forward filtering of sensitivities as an alternative. Also, for those cases the matrix $\mathbf{A}$ can be proven to be of full rank and invertible.

As a final observation one can see from the combination of equation (3.13) and (3.14) that the design control field must not be resolved because it cancels out, obviously. As a matter of fact, the design control field s and its discrete parameters $\mathbf{s}$ deal as a mean to establish the theory of *Vertex Morphing*. This is a big advantage regarding the implementation compared with standard CAD and other shape morphing methods because the explicit treatment of control parameters and the related data structures can be omitted by applying the *Vertex Morphing* method.

3.1.2 Optimal shape

In order to study the effect of the geometry description in terms of $\mathbf{s}$ on the solution of the shape optimization problem, let us look at the second order Taylor series expansion of the objective function,

$$\tilde{J} = J + (\nabla_\mathbf{s} J)^T \delta\mathbf{s} + \frac{1}{2}\delta\mathbf{s}^T \mathbf{H}_\mathbf{s} \delta\mathbf{s} \tag{3.15}$$

$\nabla_\mathbf{s} J$ and $\mathbf{H}_\mathbf{s}$ are the gradient vector and the Hessian matrix respectively. The $\mathbf{s}$ subscripts show that the operators act w.r.t the control

parameters $\mathbf{s}$. Solving the stationary condition $\nabla_{\mathbf{s}}\tilde{J} = 0$ for $\delta\mathbf{s}$ gives,

$$\delta\mathbf{s} = -\mathbf{H}_{\mathbf{s}}^{-1}\nabla_{\mathbf{s}}J \tag{3.16}$$

Now we reformulate equation (3.16) in terms of $\mathbf{x}$. Assuming $\mathbf{A}$ to be invertible and using $\delta\mathbf{s} = \mathbf{A}^{-1}\delta\mathbf{x}$,

$$\begin{aligned} \delta\mathbf{s} &= \left(-\mathbf{A}^T\mathbf{H}_{\mathbf{x}}\mathbf{A}\right)^{-1}\mathbf{A}^T\nabla_{\mathbf{x}}J \\ \Rightarrow \delta\mathbf{x} &= \mathbf{A}\left(-\mathbf{A}^{-1}\mathbf{H}_{\mathbf{x}}^{-1}\mathbf{A}^{-T}\right)\mathbf{A}^T\nabla_{\mathbf{x}}J = -\mathbf{H}_{\mathbf{x}}^{-1}\nabla_{\mathbf{x}}J. \end{aligned} \tag{3.17}$$

As it can be seen, solving the shape optimization for the design parameters $\mathbf{s}$ results exactly to the solution of the problem with $\mathbf{x}$ as design variable. Assuming the problem to be convex, there is no influence from the choice of $\mathbf{A}$ on the optimal design, and the design space is as broad as the discrete geometry allows for. In other words and as mentioned before, the role of the full rank matrix $\mathbf{A}$ is just to re-parametrize the geometry by a variable transformation (scaling). It is clear that this effect can only be exploited by first order or quasi-Newton optimization algorithms, as it cancels out if the full second order information is used. As a matter of fact for the applications reported in chapter 6, very good experience had been made with simple steepest descent techniques using the filter to converge to intentional selected local minima.

3.1.3 Path to optimum

Independence of the optimal solution of the filtering can be seen in the simple shape optimization example of figure 3.2. The objective is to minimize the difference between the shape and a discrete target curve $\mathbf{x}^{target}$,

$$J = \left\|\mathbf{x}^{target} - \mathbf{x}\right\|_1 = \sum_{i=1}^{40}\left|x_i^{target} - x_i\right| \tag{3.18}$$

The steepest descent method with an adaptive step size is used. $\mathbf{A}$ is constructed based on a piecewise linear hat filter function (equation 3.19). The problem is solved for two filter radii $r = 3$ and $r = 6$.

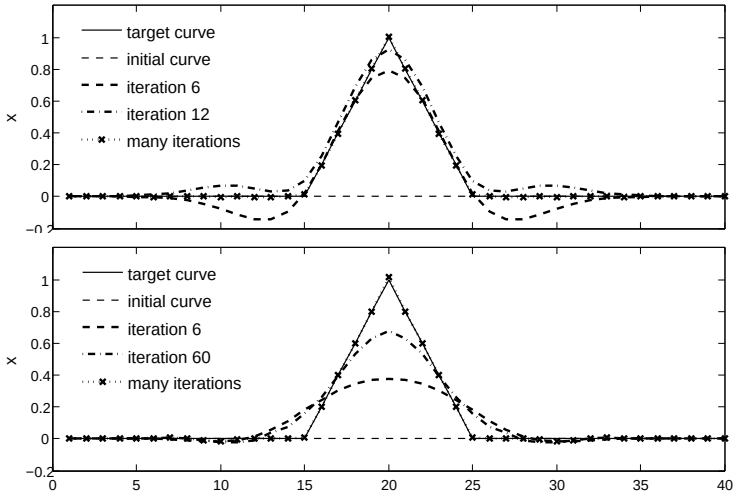


Figure 3.2: Evolution of the design curve throughout the optimization with two filter radii, 3 (top) and 6 (bottom)

$$N_{ij} = \begin{cases} \dfrac{1}{r} - \dfrac{|i-j|}{r^2} & , |i-j| \leq r \\ 0 & , \text{otherwise} \end{cases} \tag{3.19}$$

In the first few iterations of the optimization, the variation of the geometry takes place in shape modes with wavelengths equal or larger than the filter radius. That is the reason why the case with the smaller filter radius gets closer to the sharp edges of the target curve in less iterations. Of course, as one proceeds iterating, both methods monotonically reduce their difference to the target shape. Moreover, after many iterations, independent of the choice of filter, the design curve lays perfectly on the target curve despite its sharp edges. As a conclusion, the node-based methods preserve the dimension of the design space and the optimal solution, but affects the treatment of basic shape modes during the design evolution.

3.1.4 Selected local minimum

As explained in section 3.1.3, filtering affects the evolution history, but not the solution of convex optimization problems. However, almost all engineering optimization problems are non-convex. In this case, the established relation between the mesh points (filtering) guides the solution towards a desired local minimum which contains wavelengths larger or in the same range as the influence size of the filter function. This phenomenon is repeatedly seen in engineering node-based shape optimization and since is not the major topic of focus in this paper, we suffice to illustrate it with the following one dimensional example.

Consider a shape optimization problem with the same design space as the previous examples. This time, there exist two target curves. The objective is to get as close as possible to either of those curves (figure 3.3). Let us define the objective function as following:

$$J = \sum_{i=1}^{40} \left| x_i^{target1} - x_i \right| \left| x_i^{target2} - x_i \right| \tag{3.20}$$

This is a simple way to bring the non-convexity nature to this basic one-dimensional example. The existence of two local minima for each design variable x_i can be seen in figure 3.4. In this figure, the objective function values are plotted over different values of x for $i = 13, 19, 20$. The objective of the optimization problem is the sum of contributions of all the sections.

By performing steepest descent optimization with adaptive step size as in the previous example, one can direct the solution to a certain local optimum by the choice of the filter radius (figure 3.3). Smaller filters lead to locally optimal shapes with smaller features and high curvatures, whereas large filter radii seek for shapes with large geometrical modes.

3.1.5 Explicit and implicit filters

As discussed in section 2.3, solving ill-posed problems requires regularization (filtering). Dealing with many design variables, almost every

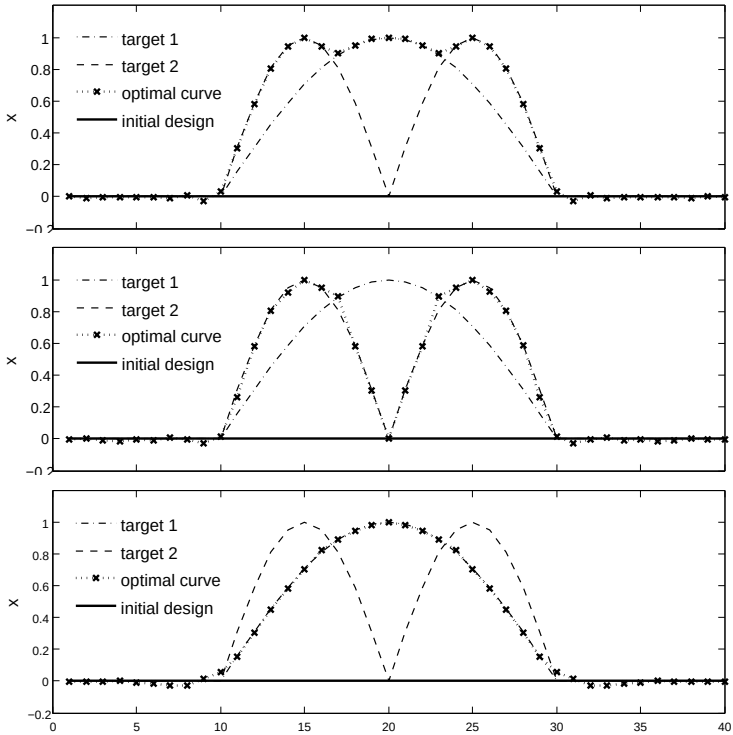


Figure 3.3: Three local minima captured by filter radii from top to bottom: $r = 1$, $r = 2$ and $r \geq 4$

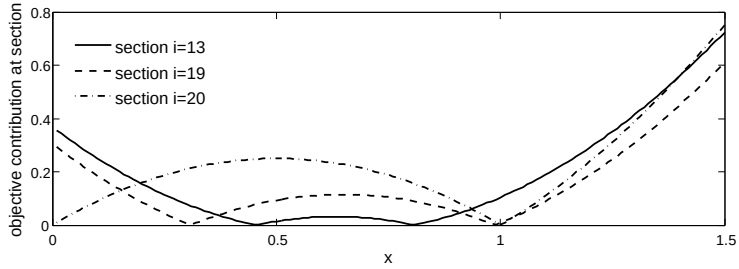


Figure 3.4: Contribution of three sample sections to the objective, each section has two minima

node-based shape optimal design requires a filtering step [57, 70, 86]. From the geometrical view point, filtering prevents non-smooth geometries and mesh dependency in design [104, 131]. Moreover, the filter size is used as a design handle in order to control the minimum curvature or wave length in the shape update vector. In section 2.3 regularization for shape optimization was classified in two general types, explicit and implicit. As the names imply, the explicit filtering operator is applied on the raw (known, noisy) field, whereas the implicit operator is applied on the smoothed (unknown) field.

A well established and commonly used implicit filtering method in adjoint shape optimization is Sobolev-gradient smoothing [57, 60, 86, 116]. This method suggests the filtered shape variations to be the solution of an elliptic equation,

$$(I - \epsilon\Delta)\delta\tilde{x} = \delta x. \tag{3.21}$$

$\delta\tilde{x}$ is the filtered shape variation, δx is the raw shape variation and ϵ is an arbitrary scalar penalizing high curvature in shape variation. Applying the same idea to the shape derivatives of a node-based shape optimization leads to a similar equation. Calculation of the "curvature operator" Δ, which is the second derivative of the geometry w.r.t. to surface coordinates, requires knowing the topology of the surface mesh and results in a "smoothing tensor" which can be used as a pre-conditioner to the sensitivity equations. As discussed before, this filtering can be mechanically interpreted as well, as the elliptic equation solved is similar to a linear elasticity problem.

In explicit filtering, the regular field is obtained by convolution of the raw field x with a kernel (section 2.3). In the discrete case, the integral converts to a weighted summation which can be represented as a matrix vector multiplication. That matrix is the filtering operator. The fact that this operator is constructed directly (explicitly), and not implicitly by inverting an other operator (e.g. curvature) is from the computational and implementation point of view advantageous. Similar to the prove about the equivalency of the kernel regularization and the least squares Tikhonov regularization in the field of statistics [142], Stück and Rung [131] show that explicit filtering with a Gaussian kernel (equation (3.27)) is first order equivalent to the implicit Sobolev-

gradient filter for shape optimization. The standard deviation of the kernel σ (which can be seen equivalent to the filter radius) corresponds to ϵ in equation (3.21).

Implicit and explicit filters are very similar, as they both provide smoothness by establishing a distance-based relation between the individual nodal values. The inverse of the implicit filtering operator of equation (3.21), $(I-\epsilon\Delta)^{-1}$ can be regarded as an explicit filter ($(I-\epsilon\Delta)$ must be invertible). In *Vertex Morphing*, the $\mathbf{A}$ matrix plays almost the same role as the explicit filtering operator. The only difference is that in addition to the shape derivative filtering, the transpose of that matrix is used to filter the geometry variation.

It was demonstrated by the simple optimization problem of section 2.1.2 that ideally the Hessian matrix should be used as the filtering operator, which upgrades the solution strategy to a second order one. However, finding such a filtering operator requires evaluation of the Hessian matrix itself. Some of the proposed filters try to provide an approximation of that ideal operator by analyzing the sensitivity equations, e.g. in [116], which is possible only if the physical equations are very simple. As a result, the effectiveness of the filtering operator strongly depends on the nature of the governing equations and a general comparison of different filtering operators is not possible.

The common point in all mentioned filtering methods is that they are tuned by a single scalar value, "filtering coefficient" (ϵ in equation (3.21), standard deviation σ of the Gaussian kernel, filter radius r of a hat function, penalization factor ϵ in Tichonov regularization of equation (2.43), etc.) which indicates the "filtering intensity". Practically, the choice of the filtering method itself (implicit, explicit or *Vertex Morphing*) is not as decisive as the choice of the filtering coefficient. This is demonstrated in the following 1D example. Here, the goal is again to reach the target curve from an initially flat geometry. The objective function is the sum of squares of the distance of each point

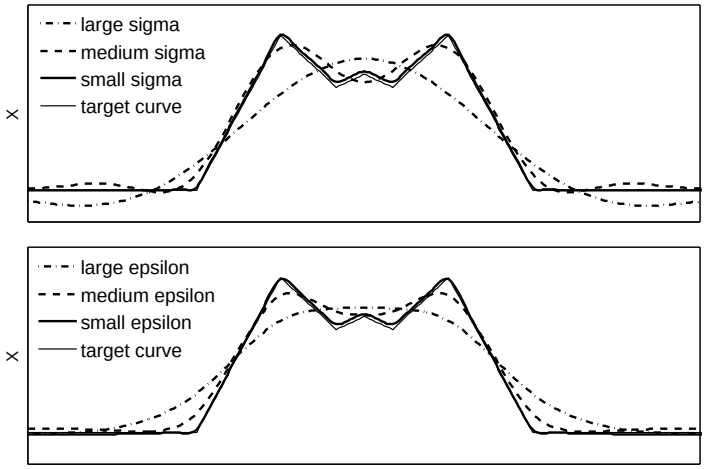


Figure 3.5: Improved shape after 25 optimization iterations. Influence of ϵ for implicit filtering (bottom) and the standard deviation σ for *Vertex Morphing* (top)

to the target curve:

$$J = \sum_{i=1}^{40} \left(x_i^{target} - x_i \right)^2 \tag{3.22}$$

The target curve is selected such that it includes 3 different geometrical modes. The objective is a quadratic equation with $H_{\mathbf{x}} = I$ and therefore the solution can be predicted with a single descent step together with a line search. The purpose of this study is to observe the resulting shapes by different smoothing methods and filtering coefficients. Otherwise, in such a problem, filtering would decelerate the solution as it changes the ideally scaled initial Hessian.

Similar to the 3D examples presented in chapter 6, the steepest descent algorithm with a fixed step size is used. The shape optimization is performed for 25 iterations, once with an implicit filtering (equation (3.21)), and once with *Vertex Morphing* based on a Gaussian kernel. The reason to select a Gaussian filter function is for the sake of consistency with

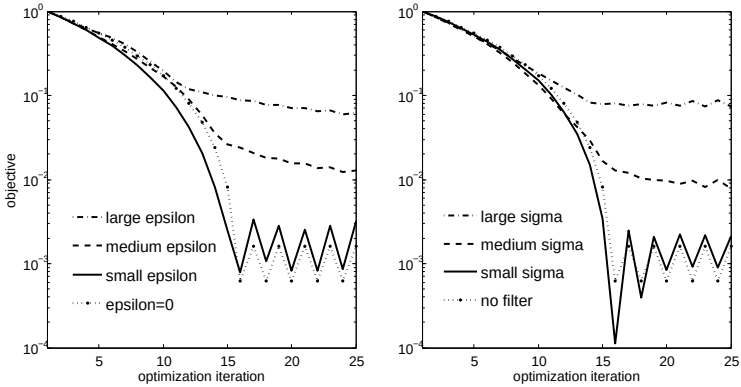


Figure 3.6: Optimization convergence history with fixed step steepest descent. Influence of ϵ in implicit filtering (left) and the standard deviation σ in *Vertex Morphing* (right)

the function used in chapter 6. Otherwise for example a hat filter would qualitatively perform similar to the selected Gaussian one. As it can be seen in figure 3.5, depending on the choice of the filtering coefficient, the resulting shape includes either one, two, or all three modes of the target curve. This is the case for both filtering techniques, and the slight difference seen in the geometries is due to the selection of different filtering operators. Figure 3.6 shows in semi-logarithmic diagrams the objective value in every iteration step for different filter coefficients (here ϵ and σ). Again, what rules the optimization history is the choice of the coefficient. Due to the optimization algorithm and the fixed update step, in both methods, after the convergence in shape modes larger than a certain size, the improvement gets very slow. This would help to restrict appearance of smaller shape modes in the design. For further convergence of the optimization problem, the choice of the filter function is decisive and has to be done according to the the governing state equation.

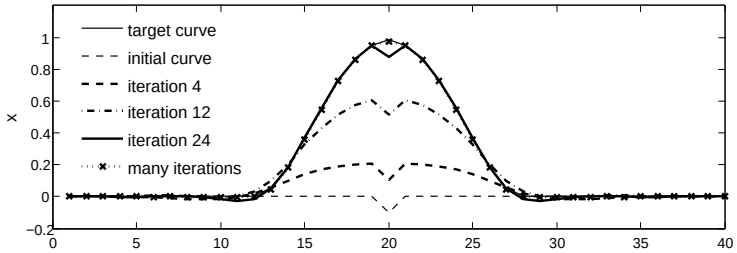


Figure 3.7: Design features of the initial geometry are preserved until late stages of shape optimization and will vanish as the the latest

3.1.6 Design features

A substantial aspect in industrial shape optimization is preservation of design features. These features usually consist of sudden transitions in the geometry, such as sharp corners at the intersection of different patches. It is often desired to morph the shape smoothly as a whole, respecting those features, rather than smearing them out, even though those features usually do not exist in the optimal shape.

In node-based optimization with filtering, the feature lines are preserved in the early shape updates since in those steps the update takes place only in large wavelengths. This can be seen in the example of figure 3.6, in which for every filtering coefficient, the curve has a clear slope reduction at a certain iteration. At this point the large scale shape modes are resolved. After that, due to the chosen optimization algorithm, the small wavelengths are resolved in a slower rate compared to the one of the large scales. This brings the feature keeping property to the method.

The mentioned property is visualized in the example of figure 3.7. Here, the goal is again to get as close as possible to the smooth target curve. There exists a kink at $i = 20$ as a design feature. In the first iterations, the shape gradually evolves towards the target curve, with almost untouched design feature. At iteration 24, although the whole shape nearly lies on the target curve, the feature is still perfectly preserved. This shape, which is of course slightly away from the optimum, is

usually the industrially preferred geometry. Evidently, if one continues iterating, the feature would eventually disappear and the curve would perfectly lie on the target.

3.1.7 Topology filtering

The shape (or sensitivity) filtering in shape optimization is equivalent to the density filtering used in topology optimization. In the context of topology optimization, filtering is discussed in even further details which is motivated by the fact that clear separation fronts between the material phases should be generated. The original approach was proposed by Sigmund [118] which performs a "sensitivity filtering" as it filters the gradient of the objective functional w.r.t. the material density. That technique is one-to-one equivalent to the shape derivative filtering in shape optimization. Some years later, [15, 70] and Sigmund himself [119] suggested to filter the material density instead of the sensitivity field. Writing the modified Sigmund filter in terms of equation (3.4) would be:

$$\rho(\xi_0) = \exp\left(\int_{-1}^{1} A(\xi_0, \xi) \ln(s(\xi)) d\xi\right) \tag{3.23}$$

3.1.8 Link to CAD

The proposed shape control method assumes a design space with the same dimension as the space of geometrical parameters. Each design entity influences linearly all (or some) of the geometrical parameters. This is very similar to the CAD parametrization. The only difference is that CAD parametrization of discrete surfaces has often much fewer parameters compared to the geometrical quantities. This is equivalent to a non-square $\mathbf{A}$ matrix, which reflects the dimension reduction from the number of geometry to control parameters and where the spline base functions take the role of the filter defining the minimum wave lengths of the generated shape. In such a case, equation (3.17) is not valid anymore, and the final result is clearly dependent on the choice of parametrization. Having that in mind, one can see the CAD parametrization as the case in which the control field s of figure 3.1 is discretized with a coarser mesh compared to the one of the geometry

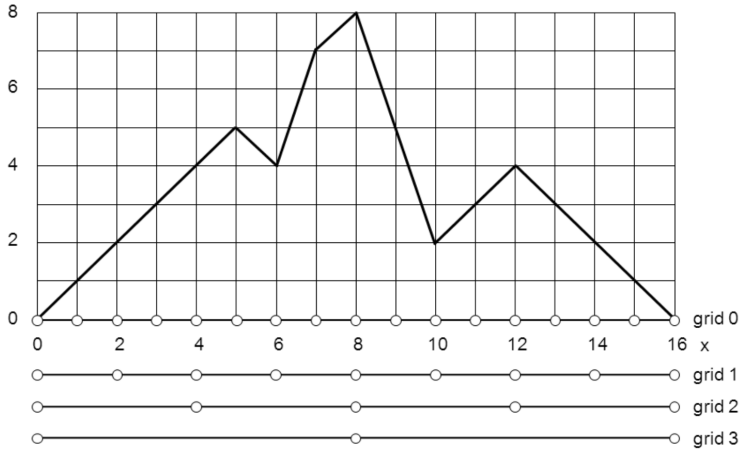


Figure 3.8: 1D example, with four different grid sizes for the control field

field x. In this way, each column of the non-square $\mathbf{A}$ matrix is constructed depending on the desired CAD function, e.g. NURBS.

In order to study the influence of the discretization of the control field, the following simple example is presented. Similar to the previous examples of this chapter, a 1D best fit optimization problem is solved. This time a very fine discretization for the geometry is selected, so that the influence of the discretization of the control field is better observed. Four different grid sizes are tested for the s field. The target curve is designed such that it can be produced by piecewise linear interpolations of coordinates at the discretization points of the finest grid, "grid 0" (figure 3.8). The filtering is done by a hat function whose radius is chosen to be equal to the size of the mesh length for s. For all four cases the optimization was performed by steepest descent until convergence. Figure 3.9 shows the optimal geometry generated in each case. The solution of grid 0 resembles exactly the target curve, as the number of control points is the same as the number of points used to build the piecewise linear target curve. The solution of the other cases is the best cubic B-Spline fit for the target curve with the given

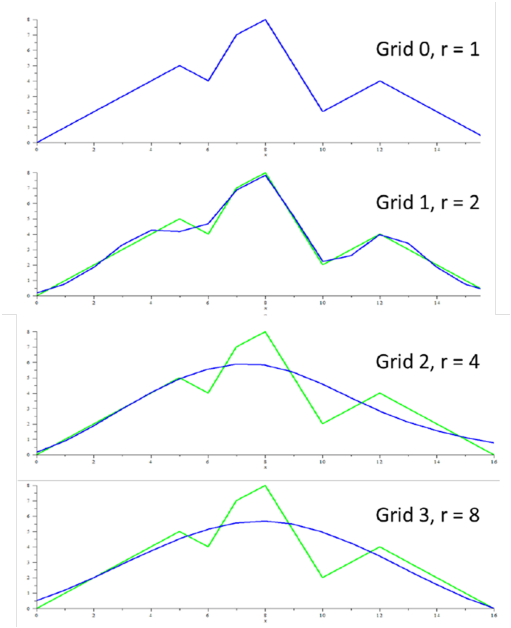


Figure 3.9: The converged best fit optimization, with 17, 9, 5 and 3 discretization points for s

number of control points, 9, 5 and 3.

Except CAD-based and node-based approaches, there exist in shape (and topology) optimization other methods which make use of an additional field in order to describe the geometry, e.g. level set [91] and phase field [132]. Those methods implicitly represent the geometry of the optimization surface by use of a background field which is defined on the 3D space. However, in *Vertex Morphing* the geometry is explicitly represented by the control field which is defined on the optimization surface itself.

3.2 3D Vertex Morphing

In this section, an enhanced version of the shape control method discussed in section 3.1 is introduced. As explained before, the main idea of the *Vertex Morphing* method is to formulate the shape optimization problem in the control space instead of the geometric space. The smooth shape transition (section 2.3.1) and surface mesh regularization (section 2.3.2) are tackled simultaneously. The out-plane update is dictated by the optimizer based on the gradient vector and shall be smoothed (filtered). In parallel to that and for the in-plane regularization problem, finding the tangential update component requires prediction of the mesh density distribution after applying the out-plane update. It is desired that the regularizing term is included directly in the shape control operator A for sake of consistency, efficiency and robustness. In other words, instead of including the surface mesh quality criterion as a supplementary optimization problem into the cost functional, the optimum tangential design velocity is predicted as a function of the current geometry and current shape derivative, through the shape control operator.

3.2.1 Dimensional reduction

The control field s is discretized by the same discretization as that of the geometry x. However, unlike the one dimensional case, in 3D, s and x do not have the same dimension. The A operator projects the three dimensional geometry field (vector field) onto the one-dimensional control field (scalar field). This dimensional reduction (projection) enforces weakly the mesh density conservation to the shape variation problem. Hence, for the discrete problem, $\mathbf{A}$ of equation (3.10) is not a square matrix anymore. Note that the original shape optimization problem has as many design variables as surface points (and not 3D surface coordinates), since the shape derivative itself is a scalar field. Therefore there is no restricting constraint enforced to the shape optimization problem.

Similar to equations (3.4) the 3D geometry variation at every point $\delta\bar{x}(\bar{\xi}_0)$ is related to the variation of the control field δs through an

integral over the optimization surface Γ:

$$\delta\bar{x}(\bar{\xi}_0) = \int_\Gamma \bar{A}(\bar{\xi}_0, \bar{\xi}) \left| \frac{d\bar{\xi}}{d\bar{x}_t} \right| \delta s(\bar{\xi}) d\Gamma \tag{3.24}$$

where

$$\bar{A}(\bar{\xi}_0, \bar{\xi}) = N\left(\|\bar{x}(\bar{\xi}_0) - \bar{x}(\bar{\xi})\|_2 \right) \frac{\bar{g}_1 \times \bar{g}_2}{|\bar{g}_1 \times \bar{g}_2|} \tag{3.25}$$

and

$$\bar{x}_t = (x_{t_1}, x_{t_2}), \bar{\xi} = (\xi_1, \xi_2), \bar{x} = (x_x, x_y, x_z) \tag{3.26}$$

$\left| \frac{d\bar{\xi}}{d\bar{x}_t} \right|$ is the determinant of $\frac{d\bar{\xi}}{d\bar{x}_t}$. Unlike in equation (3.4), the $\bar{A}$ operator of equation (3.24) is a 3D vector with one component for each spatial coordinate. In equation (3.25) the base vectors are evaluated at the second parameter of the $\bar{A}$ operator, i.e. at $\bar{\xi}$. N is an arbitrary filter function, here motivated by the probability density function:

$$N(\gamma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(\frac{\gamma^2}{2\sigma^2})} \tag{3.27}$$

σ is the filtering intensity (filter radius) and assumed to be constant, and γ is the Euclidean distance to the center of the filter. Including the surface normal direction in $\bar{A}$ of equation (3.25) is equivalent to averaging the normal surface direction scaled by the shape derivative. This makes the shape transition to take place smoothly in all three spatial directions without enlarging the optimization problem by a factor of 3. The integral of equation (3.24) is approximated using the same discretization as of the surface geometry.

In the discrete geometries, the most efficient and robust way to calculate the integral of equation (3.24) is to express it as a sum over the weighted nodal values i.e. trapezoidal rule, as it is done in most of explicit filtering techniques in shape and topology optimization. In other words, the integration is done in absence of δs, which is further multiplied to the approximated integral. This method is referred to as "post-scaling" and is explained in [12].

3.2.2 Normal and tangential components

One can study the behavior of the method by decomposing the shape variation vector into in-plane and out-plane components. We present this decomposition only to compare the method with the ones with separate in- and out-plane regularization steps. For the implementation it is enough to construct the discretized $\bar{A}$ operator which includes both normal and tangential directions.

$$\delta x_n(\bar{\xi}_0) = \int_\Gamma \frac{\bar{g}_3(\bar{\xi}_0)}{\left|\bar{g}_3(\bar{\xi}_0)\right|} \cdot \frac{\bar{g}_3(\bar{\xi})}{\left|\bar{g}_3(\bar{\xi})\right|} \delta s(\bar{\xi}) N\left(\|\bar{x}(\bar{\xi}_0) - \bar{x}(\bar{\xi})\|_2\right) \left|\frac{d\bar{\xi}}{d\bar{x}_t}\right| d\Gamma \tag{3.28}$$

$$\delta x_{ti}(\bar{\xi}_0) = \int_\Gamma \frac{\bar{g}_i(\bar{\xi}_0)}{\left|\bar{g}_i(\bar{\xi}_0)\right|} \cdot \frac{\bar{g}_3(\bar{\xi})}{\left|\bar{g}_3(\bar{\xi})\right|} \delta s(\bar{\xi}) N\left(\|\bar{x}(\bar{\xi}_0) - \bar{x}(\bar{\xi})\|_2\right) \left|\frac{d\bar{\xi}}{d\bar{x}_t}\right| d\Gamma \ , \ i = 1, 2 \tag{3.29}$$

This decomposition shows that the shape relevant (normal) component is evaluated by convolution of δs with a kernel N and additional consideration of surface curvature through a multiplier $\bar{g}_3(\bar{\xi})\bar{g}_3(\bar{\xi}_0)$. In the case of a constant curvature and uniform δs, the in-plane term $\delta\bar{x}_t$ is zero as the contributions of all the surrounding points around ξ_0 cancel out each other. However, if the surface curvature varies in the vicinity of ξ_0, the term $\frac{\bar{g}_i(\bar{\xi}_0)}{\left|\bar{g}_i(\bar{\xi}_0)\right|} \cdot \frac{\bar{g}_3(\bar{\xi})}{\left|\bar{g}_3(\bar{\xi})\right|}$ amplifies the contribution of the regions of higher curvature to the tangential direction. This leads to a $\delta\bar{x}_t$ in the opposite direction of the curvature gradient. As the result, mesh points are pushed away from the higher to lower curvature regions. Parallel to that, regions with smaller δs in a positive curvature would attract the surface points from the regions with larger δs due to their higher contribution to $\delta\bar{x}_t$. The length of the tangential update component $\delta\bar{x}_t(\bar{\xi}_0)$ is proportional to the variation of the surface curvature and the design update field δs. Therefore, in the prediction of the tangential update term, both mesh deterioration parameters mentioned in section 2.3.2 are considered. Here, compared to the equation (2.51), it is not possible to predict the exact mesh density change as δx_n is not known. We assume that δx_n and δs fields have comparable large scale distributions. This is not a very poor approximation as here the δx_n is nothing but the filtered δs. To summarize, in *Vertex Morphing*

the suitable tangential update component is predicted such that the updated surface $\bar{x}^{k+1}$ has a similar $\frac{d\bar{x}}{d\bar{\xi}}$ distribution as of the previous step $\bar{x}^k$. This leads to an indirect preservation of the mesh density for shape optimization of discrete surfaces, as it is practically proven by various challenging applications solved by this method.

The sensitivity vector is similarly calculated based on equation (3.8) as:

$$\frac{dJ}{ds(\bar{\xi}_0)} = \int_\Gamma \bar{A}(\bar{\xi}, \bar{\xi}_0) \frac{d\bar{\xi}}{d\bar{x}} \frac{\partial J}{\partial \bar{x}(\bar{\xi})} d\Gamma \tag{3.30}$$

One should notice that in contrast to equation (3.24), this time the second parameter of the A operator is $\bar{\xi}_0$ and therefore the base vectors of equation (3.25) have to be evaluated at $\bar{\xi}_0$.

Basically, the shape sensitivity matrix $\mathbf{A}$ is a large and dense matrix. However, one can localize the radius of influence in equation (3.27) by applying a top-hat filter function on N, which leads to a sparse $\mathbf{A}$. For calculation of the shape derivative, there is no need to calculate and store the matrix at once and individual rows can be sequentially calculated and used in a matrix-free algorithm.

3.2.3 Example

The performance of the method in real industrial examples is shown in section 6. Here, in order to evaluate the proposed method only w.r.t. the surface mesh quality, a simple shape optimization example of a 2D ducted flow is presented (figure 3.10). The objective is the power loss (explained in section 4.3, equation (4.27)) and the curved part of the upper wall is the design surface. As one can guess, the optimization tries to clear the clogged section of the duct by moving the curved part upward. In node-based shape optimization reduction of the surface curvature is a challenge as it easily leads to overlapping elements.

Three different cases were studied. The first case is a “standard” node-based procedure with explicit filtered shape derivatives and no in-plane

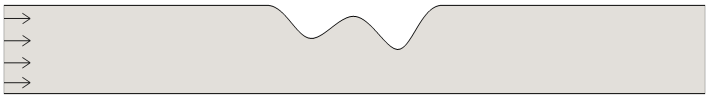


Figure 3.10: 2D example, ducted flow

treatment. The second case is similar to the first one, but with an additional surface mesh in-plane regularization [128] after each optimization step. The third case is solved by *Vertex Morphing*. Figure 3.11 compares the mesh density conservation in the first 25 optimization iterations, at which the duct wall is almost flat. The graphs visualize the worst ratio of the surface element to the initial mesh normalized by the length of the optimization surface, as an indicator of the worst mesh density. Therefore the value 1 indicates the perfect mesh density conservation and 0 is the collapse of the surface mesh.

In the first case, the mesh gets distorted after a few steps and the optimization cannot be continued. Adding in-plane regularization of the surface mesh would keep the surface mesh density (and quality) almost at its best. However, a linearized PDE on the optimization surface has to be solved at every iteration. This brings computational cost and possibly lack of robustness if the surface is not smooth or the mesh quality is low (as it is the case in many industrial examples). *Vertex Morphing* offers a relatively good mesh density preservation by including the in-plane update prediction in the link between the design variables and the geometry. The computational cost is the same as the case without any in-plane treatment, and it can perform well for non-smooth surfaces and low quality meshes. The in-plane term approximation is more accurate for larger filters as it is seen in figure 3.11. It should be added that neither *Vertex Morphing*, nor the proposed in-plane regularization methods can unconditionally guarantee that the surface mesh always stays admissible through the complete optimization procedure.

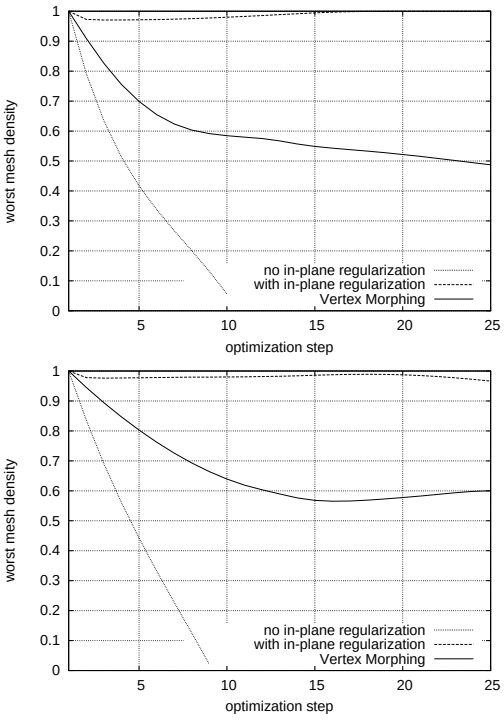


Figure 3.11: Comparison of the surface mesh density conservation for two filter radii (the filter of the bottom graph is 1.6 times larger of the top one.)

Chapter 4

Numerical flow simulation

This chapters discusses the other module in the shape optimization process, the numerical simulation of the physical phenomenon. It is clear that node-based shape optimization can be done only in combination with a gradient-based algorithm because of the large number of design variables. For the same reason, evaluation of the sensitivities in a black box manner, e.g. finite differences is not possible, and one should numerically solve the adjoint state equations. Therefore, in order to develop an adjoint based shape optimization workflow, a good insight to the physical equations and modeling details is needed. Although the proposed shape control method, *Vertex Morphing*, is purely geometrical and can be used in combination with any numerical simulation, the target application of this work is Computational Fluid Dynamics (CFD) with the focus on numerical wind simulations.

In order to optimize a transient system such as a wind flow problem, one should first define time-independent objective functions, unless the design can adapt itself to the instantaneous boundaries (automatic control). The straightforward choice is to work with averaged, peak, or other statistical quantities. Since objective functions defined by

minimum or maximum quantities are not differentiable, here we focus on optimizing the mean values, e.g. averaged flow forces applied on a structure. There are two ways to calculate the sensitivity vector of the averaged objective. The first way is to solve the transient equations and their adjoints, and then average the sensitivity values. This requires solution of the transient adjoint problem in the time domain with backward time stepping, which is requires an extremely high amount of memory. The state of the art approach for overcoming this issue is the check-pointing method [43], in which the memory requirement is reduced by introducing some additional processing (recalculation of the primal solution in certain intervals). The second way is to solve the fluid equations and their adjoints directly for the averaged quantities (steady RANS). Of course the latter way is computationally cheaper, but the level of accuracy in the averaged equation can be poor. Thanks to some decades of research, equivalent steady CFD models designed for specific applications are able to deliver a relatively good order of accuracy. This is shown in this chapter as well through a simple example. The main question however is that if a shape optimization based on the averaged adjoint equations can have the same level of accuracy. This question is discussed in section 6.3, for which the theoretical background of this chapter is needed.

In the following sections the key equations of turbulent flow simulations are firstly presented. Then, the basics of adjoint sensitivity calculation for this type of problems are discussed. At the end, the challenges in extending the work into an aeroelasticity optimization workflow, with consideration of the Fluid-Structure Interaction are mentioned.

4.1 Wind engineering

Prediction of wind loads on man-made structures has been a hot topic in engineering since many years. The well known *Davenport Wind Loading Chain* shown in Figure 4.1, nicely breaks down the design of wind-exposed structures into few steps [20]. Box 'a' is fed as the input condition to boxes 'b', 'c' and 'd'. Box 'e' uses the result of the previous step to evaluate (and in the case of this work to improve) the worthiness of the design. Here, by wind engineering simulations we refer to the three middle boxes, which describe the physical behavior of

the wind-structure system. It should be added that in a realistic wind model, there is a both-way interaction between boxes 'c' and 'd', as the deformation of the structure changes the surrounding wind field. Here, we focus on prediction of the wind effects on the structure, for a given wind scenario. This analysis is further used for shape optimization.

Computational Wind Engineering CWE is a relatively young sub-field in wind engineering which aims at numerically simulating the atmospheric wind flow around structures. Large computational effort and complexity of the atmospheric turbulence are the restricting factors in CWE simulations. However, as the computational power and parallelization techniques rapidly improve, CWE receives more and more attention due to its potentials. Classical wind load calculations are based on available standards and norms. These standards are the result of several experimental measurements in wind tunnels, etc. Despite the wide range of applicability of the standards, there are cases in which a more enhanced wind simulation is required. For example, when the shape of the structure is not a standard shape (square, cylinder, etc.) usually a size-reduced wind tunnel experiment is performed, which is time consuming, expensive, and has its owns complexities. But even a wind tunnel experiment cannot always accurately describe the realistic wind flow around the structure, for two main reasons. First of all, wind tunnel experiments are almost always performed with reduced wind speed together with correction factors, which includes inaccuracy. Secondly, in case of light-weight flexible structures, aeroelastic behavior of the wind-structure system is extremely difficult to be captured in the reduced experimental model. The mentioned aspects, together with many other advantages of numerical models, motivates the use of CFD in prediction of wind effects on structures.

4.2 Computational wind

CWE has several applications ranging from climate modeling and air pollution simulation in urban areas to wind turbines. Here we focus only on modeling of wind effects on structures, including the wind loads and the interaction between the structure and the wind. We also assume that all the required input parameters such as wind velocity, roughness information, etc. mentioned in boxes 'a' and 'b' of figure 4.1

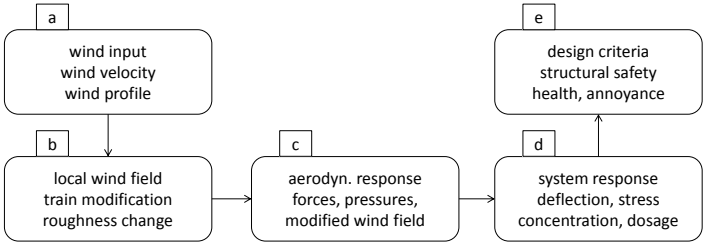


Figure 4.1: The event chain of wind effects (adapted from Davenport 1982 [20])

are provided. Having in mind the low viscosity of air and typical wind speed (which can reach up to multiple tens of meters per second) it is obvious that the wind flow has a large Reynolds number and is highly turbulent. There are some comments on turbulent modeling for CWE in the next section 4.2.1. In the aimed application, it is necessary to consider the viscosity of the air, and therefore inviscid Euler equations are not adequate to describe the wind field. This is due to the nature of the atmospheric boundary layer which is formed as a shear interaction of the air flow and the rough surface of the earth. Despite the high Reynolds number, the wind speed is yet much bellow the propagation speed of acoustic waves in air, which results in a relatively low Mach number and allows us to assume the flow to be incompressible. Based on the mentioned aspects, the incompressible set of Navier-Stokes equations for momentum and continuity are employed:

$$\begin{aligned} \frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} &= -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_k} 2\nu \mathcal{S}_{ik} \\ \frac{\partial u_i}{\partial x_i} &= 0 \end{aligned} \tag{4.1}$$

where

$$\mathcal{S}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{4.2}$$

$\mathcal{S}_{ij}$ is the symmetric strain rate tensor. u_i and x_i represent the three components of the velocity and position vector in the Cartesian coordinates. ν is the kinematic viscosity of the fluid, ρ the fluid density and p the pressure. Note that the indexes follow summation (Einstein)

convention. The rest of this chapter is spent on how to setup a CWE simulation and how to achieve the design sensitivities of a system governed by Navier-Stokes equations.

4.2.1 Steady and transient turbulence simulations

This chapter introduces basics of steady RANS and LES turbulence models, and their accuracy and applicability for CWE and shape optimization. Although in both methods the flow equations are solved based on averaging, the fact that RANS equations are steady makes it less accurate particularly in case of bluff body simulation. Considering that in shape optimization only averaged RANS equations can be applied, here (and in chapter 6) we verify the RANS results through a high fidelity flow simulation with transient LES.

In fluids engineering, the majority of the flows are turbulent (or non-laminar). Turbulent motion is governed by many vortices of different size and orientation. These vortices are called eddies and their behavior appears random. A large spectrum of length and time scales exists, with the largest scales referring to the scale of the bounding flow and the smallest scales referring to the influence of viscosity. The energy dissipation mechanism in turbulent flows can be described as the following. The large scale eddies transfer the energy to smaller ones, and so on, until the viscous characterized scales (Kolmogorov scale) dissipate the energy. The transmission of large scales to smaller scales can be described using the periodic behavior of an eddy. For illustration purpose, let us assume the simplified case in which the velocity of an eddy at a certain point and time instance follows the simple sinusoidal equation $u = u_0 \sin(\omega t + kx)$, where ω is the angular frequency, k the wave number and u_0 the velocity amplitude. Although the temporal and the viscous terms of the Navier Stokes equations would produce momentum oscillations with the same frequency as of the velocity field, the non-linear convective term $u_j \frac{\partial u_i}{\partial x_j}$ includes momentum oscillations with the angular frequency twice as big as ω, since the convective momentum of that velocity is: $-\frac{1}{2} u_0^2 k^2 \sin(2\omega t + 2kx)$.

Modeling turbulent flows with Direct Numerical Simulation (DNS) requires a mesh resolution which can capture the eddies in Kolmogorov

scale. This leads to an extremely fine mesh and small time step. The required number of grid points scales with $Re^{\frac{9}{4}}$, which means that with the current computing power, DNS is possible only for turbulent flows with small Reynolds numbers. Therefore, for engineering flows - whose Reynolds numbers can reach up to 10^9 - turbulence "models" (in contrast to turbulence simulations) are used. Those models make use of the fact that the behavior of the small scale eddies are almost independent of the larger scales which follow the bulk structure of the flow field. This makes it possible to assume a predictable effect of the smaller scales on the larger ones in form of an analytical relation. Turbulence modeling is based on averaging of the fluctuating flow fields, such as velocity and pressure. The averaging can be an ensemble averaging as in Reynolds Averaged Navier Stokes (RANS) equations, or a filtering in space as in Large Eddy Simulation (LES).

In RANS, each field is decomposed into an average field and a fluctuating term. Representing different samples of the ensemble averaging as instances of a time dependent system, the averaged field for example for the pressure would follow:

$$p(x,t) = \tilde{p}(x) + \grave{p}(x,t) \tag{4.3}$$

$$\tilde{p}(x) = \lim_{T\to\infty} \int_T p(x,t)dt \tag{4.4}$$

where t is time and T is the total averaging time, $\grave{p}$ the pressure fluctuation and $\tilde{p}$ the averaged pressure field. Note that the time derivatives of the time averaged values are zero. Deriving the Navier Stokes equations (equation 4.1) based on equation (4.3) would lead to:

$$\begin{aligned} \tilde{u}_j \frac{\partial \tilde{u}_i}{\partial x_j} &= -\frac{1}{\rho}\frac{\partial \tilde{p}}{\partial x_i} + \frac{\partial}{\partial x_k} 2\nu \tilde{\mathcal{S}}_{ik} - \frac{\partial \widetilde{(\grave{u}_i \grave{u}_j)}}{\partial x_j} \\ &\frac{\partial \tilde{u}_i}{\partial x_i} = 0 \end{aligned} \tag{4.5}$$

with the averaged symmetric strain rate tensor:

$$\tilde{\mathcal{S}}_{ij} = \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) \tag{4.6}$$

$\widetilde{\grave{u}_i \grave{u}_j}$ is the Reynolds stress tensor normalized by the fluid density. Similarly, in spatial filtering (LES equations), the filtered field is obtained by convolution of the field with a kernel function $N(\gamma, x)$:

$$\begin{aligned} \tilde{p}(x,t) = \int_{\Gamma} N(\gamma, x) p(x-\gamma, t) d\gamma \\ \int_{\Gamma} N(\gamma, x) d\gamma = 1 \end{aligned} \tag{4.7}$$

This time, according to the definition stated in equation 4.7, the Navier Stokes equation are filtered in space.

$$\begin{aligned} \frac{\partial \tilde{u}_i}{\partial t} + \tilde{u}_j \frac{\partial \tilde{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \tilde{p}}{\partial x_i} + \frac{\partial}{\partial x_k} 2\nu \tilde{\mathcal{S}}_{ik} + \frac{\partial \tilde{u}_i \tilde{u}_j}{\partial x_j} - \frac{\partial \widetilde{(u_i u_j)}}{\partial x_j} \\ \frac{\partial \tilde{u}_i}{\partial x_i} = 0 \end{aligned} \tag{4.8}$$

Note that unlike RANS equations, the temporal derivative is present in the LES equations. In both RANS and LES, equations are solved for the smooth fields $\tilde{u}_i$ and $\tilde{p}$ as the unknowns, and the term $\frac{\partial \widetilde{(u_i u_j)}}{\partial x_j}$ is modeled (approximated) based on some assumptions. The volumetric component of the turbulent stress tensor $\widetilde{\grave{u}_i \grave{u}_i}$ appears as a modification of the pressure and can be included in $\tilde{p}$. The deviatoric (shear) component $\widetilde{\grave{u}_i \grave{u}_j}$, $i \neq j$ has to be modeled by a turbulence model.

Many RANS models replace the Reynolds stress tensor by a single scalar value, the eddy viscosity ν_T, and add it to the molecular viscosity. This assumption is known as Boussinesq eddy-viscosity hypothesis. All RANS simulations presented in this work, as well as the sensitivity

analysis explained in section 4.3 belong to this family of RANS models. The eddy viscosity is estimated based on the averaged stress rate tensor $\tilde{\mathcal{S}}_{ij}$, equation (4.6). There are usually some turbulence related quantities which help predicting the eddy viscosity. The propagation of this quantities are described by a transport equation, similar to the momentum equations.

LES resolves the time domain as well and restricts the modeling to the small scales which are not captured by the mesh size (subgrid scales). This is a reasonable assumption, as those eddies behave more or less independent of the large scale flow field. The large eddies are problem specific and have larger characteristic time scales compared to the smaller ones and therefore their transient resolving can be done by "large" time steps.

In wind engineering, LES is preferred to RANS, because the transient effects are not negligible. However, LES simulations are much more expensive, because they firstly are time-dependent, and secondly they require a finer discretization at the wall boundaries. The grid requirement for an accurate wall-bounded LES scales with $Re^{\frac{9}{5}}$, and therefore in most of wind engineering simulations, a certain amount of error is inevitable and LES is solved on very coarse meshes, which subsequently increases the time step. In such a case a mesh dependency study, similar to the one performed in section 6.3.2 can predict the order of sub-grid modeling error.

Here we study the performance of RANS in comparison to LES in an experimental benchmark designed for validation of the presented optimization process. Since the experimental results are not available yet, only the CFD verification is briefly shown as a qualitative demonstration. The water flow in a bend with bulk velocity around $1\frac{m}{s}$ is modeled, and further optimized (not presented here) This benchmark can be seen as an extension of the low Reynolds number S-duct examples shown in chapter 6. The aim of the benchmark is to verify the capabilities of the optimization by measuring the pressure loss in the initial pipe as well as the "optimized" one. Therefore not only the accuracy of the numerical scheme in prediction of the pressure loss

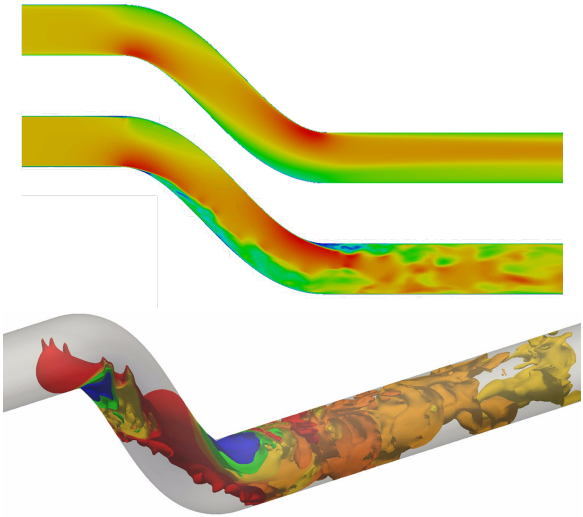


Figure 4.2: Comparison of the 3D transient LES flow and steady RANS for the water S-duct experimental benchmark. Top: RANS velocity contour, middle: LES instantaneous velocity contour, bottom: LES pressure instantaneous isosurfaces

is tested, but also the approximation level in the prediction of the improvement by the optimization loop can be quantified. In order to achieve a more general (non-tuned) comparison, the Spalart Almaras RANS model is selected, which is not designed for this type of applications. The LES model is solved on a finer mesh using Smagorinsky method. Figure 4.2 compares the velocity magnitude distribution in a section of the pipe for RANS and a time instant of LES. Moreover, the pressure isosurface of the bottom picture in the same figure indicate the three dimensional structure of the flow. According to the diagram of figure 4.3, the disagreement between the steady and averaged transient results is around 5%, which is perfectly acceptable for this type of problems. Having an accurate enough RANS solution, the shape of the pipe is successfully optimized for pressure loss. The optimization results are not presented here for the sake of briefness and in order to avoid repeated comments as the ones in chapter 6.

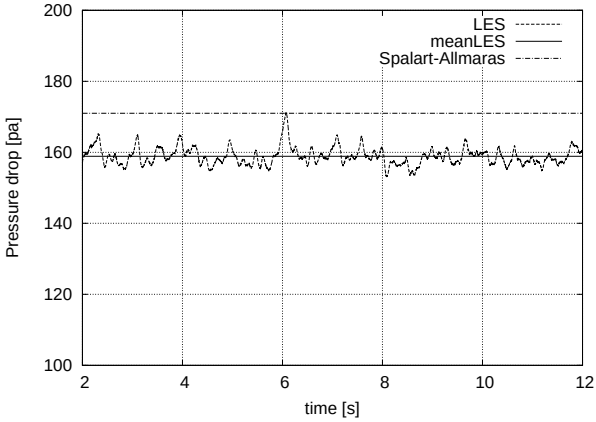


Figure 4.3: Comparison of the pressure drop for the water S-duct experimental benchmark, simulated by LES and steady RANS

4.2.2 Atmospheric boundary layer

As explained in section 4.1, reliable data about the wind flow such as the velocity profile, spatial fluctuation intensity, etc. (box 'a' in figure 4.1) is an essential input for wind simulation. This section discusses the physics of the air flow close to the earth surface and describes a meaningful wind flow modeling strategy for numerical simulation.

The temperature difference in earth's atmosphere caused by the solar radiation results in wind. The lowest part of the atmospheric wind, close to the earth, is called the atmospheric boundary layer ABL. This region is highly influenced by the objects on the earth surface, as well as the temperature, air humidity, etc. This tight interaction between the air and the earth makes the (ABL) a turbulent flow field with vertical and horizontal motion of the air particles. In order to analyze the physics of ABL, we model the flow field as a half space bounded from the bottom by a rough wall. The air travels mainly parallel to the wall where the velocity is zero. Furthermore, we assume that the

region far from the wall (higher elevations) are not influenced by the wall.

In turbulent boundary flows, the friction between the flow particles and the surface of the wall causes the wall shear stress, τ_w. This stress is transmitted through the flow domain. Very close to the wall, the flow particles cannot freely move and hence the turbulence intensity is low. In this region (the viscose sublayer) the shear force caused by the molecular viscosity appears as the reaction to the wall shear stress. Assuming a Newtonian flow, the horizontal velocity gradient in the height direction y at the wall has a slope proportional to the wall shear stress:

$$\tau_w = \rho\nu \left.\frac{du}{dy}\right|_y \tag{4.9}$$

ρ is the fluid density and ν the kinematic viscosity. Note that in equation (4.9) y direction is normal to the wall boundary. Within the viscous sublayer the horizontal velocity increases linearly with the distance to the wall. The height of the viscous sub-layer depends on the wind speed. However, normalizing the height dimension by the viscous length scale δ_ν makes it possible to have a problem independent statement about the height of this layer, i.e. $y^+ \lesssim 5$ [105], where y^+ is defined as:

$$y^+ = \frac{y}{\delta_\nu} = \frac{u_\tau y}{\nu} \tag{4.10}$$

and friction velocity u_τ describes the wall shear stress in velocity dimensions in accordance to the magnitude of the wall shear stress:

$$u_\tau \equiv \sqrt{\frac{\tau_w}{\rho}}. \tag{4.11}$$

Similarly, the dimensionless velocity is defined as:

$$u^+ = \frac{u}{u_\tau} \tag{4.12}$$

In wind engineering practice which deals with Reynolds numbers of above 10^6, $y^+ = 5$ usually does not exceed few millimeters and therefore the viscous layer of ABL does not have a considerable effect on the wind loads. However, in CFD it is very important to correctly resolve the viscous region close to the wall, because the influence of the wall boundary is transferred to the flow domain through this thin region. Moreover, the shear stress applied on the wall boundary is proportional to the velocity gradient (equation 4.9) and hence accuracy of evaluation of the fluid force on the structure depends on the quality of the near wall flow resolution.

In $5 \lesssim y^+ \lesssim 30$, the buffer layer, the flow particles are more free, and thus the molecular and the turbulent viscosity have the same order of magnitude. Further from the wall ($y^+ \gtrsim 5$), the turbulence plays a significant role in transmission of the shear stress compared to the molecular viscosity. This region is called the log-low region as the mean velocity profile has an almost logarithmic profile. The vast portion of the objects studied in wind engineering exist in this layer and therefore its modeling is of high importance. In the case of a smooth wall, the logarithmic mean velocity profile follows this distribution:

$$u^+ = \frac{1}{\kappa} \ln y^+ + B \tag{4.13}$$

or

$$\tilde{u} = u_\tau \left(\frac{1}{\kappa} \ln y^+ + B \right) \tag{4.14}$$

where B is a universal constant and κ is known as von Karman constant.

$$\kappa = 0.41 \quad , \quad B = 5.2 \tag{4.15}$$

In numerical simulation of the wall boundaries, instead of refining the discretization to the level that the viscous sublayer can be captured by the first row of discretization elements, one can analytically predict the velocity gradient at the wall by using equation (4.14). In such a case the so-called “low of the wall” or the “wall function” replaces the equation 4.9. This can save a tremendous amount of computation time by reducing the problem size, as well as increasing the time step. It

should be mentioned that equation (4.14) is valid only for a infinitely large and flat wall. Therefore using wall functions in curved geometries specially where flow separation takes place can lead to inaccuracy.

Roughness of the wall changes the friction force between the flow and the wall and hence the velocity profile. Rough walls are classified based on the size of the roughness elements on them. If the roughness exists only in the viscous sublayer, the wall is called hydraulically smooth. In transmission walls the roughness exceeds the viscous layer, but does not reach the logarithmic region, in contrast to a fully rough wall. In almost every wind simulation, the roughness elements existing on the earth surface are very large. Therefore, we focus on the boundary layer in the vicinity of a fully rough wall. Similar to the smooth wall, rough walls produce an almost logarithmic profile. Considering the fact that the roughness of the wall cannot influence the mean velocity in high elevations, one can conclude that the multiplier of the $\ln y^+$ (u_τ in equation (4.14)) should be the same for both smooth and rough walls.

$$y^+ \to \infty, \frac{u_{rough}}{u_{smooth}} \to 1 \tag{4.16}$$

This means that the roughness just shifts the logarithmic velocity profile with a constant value, which is added to B in equation (4.14). This value can be analytically calculated as a function of the roughness quantities, e.g. the roughness length z_0. Including the constant inside the ln as a division, [107] suggested the following profile to be used for ABL applications:

$$\tilde{u} = \frac{u_\tau}{\kappa} \ln\left(\frac{y + z_0}{z_0}\right). \tag{4.17}$$

In reference literature and tables, for example in [121], the values for surface roughness length z_0 in different landscapes can be found. equation (4.17) is used as function for rough wall bounded flows. However the friction velocity u_τ remains an unknown. A common practice to obtain the friction velocity is to use a reference height in which the mean velocity is known from regional measurements. Fitting the

logarithmic function to the reference mean velocity u_{ref} at the height y_{ref} would result to:

$$u_\tau = \frac{\kappa u_{ref}}{\ln\left(\frac{y_{ref}}{z_0}\right)} \tag{4.18}$$

Consequently, the turbulent kinetic energy k and the dissipation rate ε, which are necessary boundary conditions for many RANS simulations are calculated.

$$k = \frac{{u_\tau}^2}{\sqrt{C_\mu}}, \qquad C_\mu = 0.09 \tag{4.19}$$

$$\varepsilon = \frac{{u_\tau}^3}{k(y + z_0)} \tag{4.20}$$

As it can be seen in equation (4.19), the turbulent kinetic energy is assumed to have a constant value over the height. Moreover, based on equations (4.19) and (4.20), the turbulent frequency ω which is used in turbulence models such as $k-\omega$ and $k-\omega$ SST [82] is calculated as:

$$\omega = \frac{\varepsilon}{k} \tag{4.21}$$

When a transient simulation (like in LES) is desired, the mean velocity value is not enough and one has to model the oscillatory wind field, at least for the large scales. Such a field should satisfy not only the mean value quantities mentioned above, but also the statistical characteristics of the ABL. As mentioned before, the turbulent wind is a three dimensional oscillatory field and therefore a main statistical characteristic of it is the energy content of each frequency (or wave number) in each direction. For this purpose, it is desired that the wind model used for numerical simulation has more or less similar Power Spectral Density (PSD) distribution as the atmospheric wind. There exist different analytic functions that approximate the wind PSD, the most famous of which are the von Karman/Harris and the Kaimal functions. Here we do not discuss more detailed about spectral models since it is not on the main track of this document.

4.2.3 Numerical wind tunnel

In this section we review some modeling aspects used in the numerical example of chapter 6. The structure for which the fluid loads have to be evaluated is put in a virtual wind tunnel, which has bounding walls at the sides as well as wind input and output, similar to a real wind tunnel. The numerical modeling of the Silsoe cube [108] is used for demonstration. The Silsoe cube is a cubic structure with $6m$ edge length, built in a flat field in England together with various measurement devices for the evaluation of the loads on the cube as well as the characteristics of the approaching wind. The size of the cube is in the same scale as real engineering structures and the wind flow is a natural ABL, which has made it an interesting benchmark for both experimental and numerical wind models [109, 139].

It is crucial to have the computational domain large enough in order to prevent the boundaries to artificially influence the flow over the obstacle. For instance, for simulation of the Silsoe cube a minimum required distance to the wall is found to be 5, 10, 7 and 15 times the cube height, for sides, upstream, top and downstream directions, respectively. The lower boundary of the virtual wind tunnel is a no-slip wall, usually treated with wall function. The side walls and the top wall are sliding or symmetry planes. For the side walls it is also possible to apply periodic boundary conditions in order to reduce their influence on the turbulence. If the height of the domain is not large enough such that the velocity profile gets almost uniform, it is recommended to consider the shear stress at the roof boundary in order to prevent disturbance on the velocity profile [14]. The possiblity of using sliding or periodic bondary condition is an advantage compared to a real wind tunnel in which the fixed bounding walls form unwanted boundary layers. In this work a uniform constant pressure is used for the outlet boundary. The most challenging task however is the proper modeling of the inlet wind. If a steady simulation is aimed, the distributions of equations (4.17)-(4.21) are used. Figure 4.5 shows the comparison of the logarithmic mean velocity profile based on equation (4.17) with some measurement points at different heights. The distribution of the turbulence dissipation rate and the uniform kinetic turbulent energy are also plotted. Considering that the Silsoe cube has the typical height scale of wind engineering problems, it is clearly concluded from the

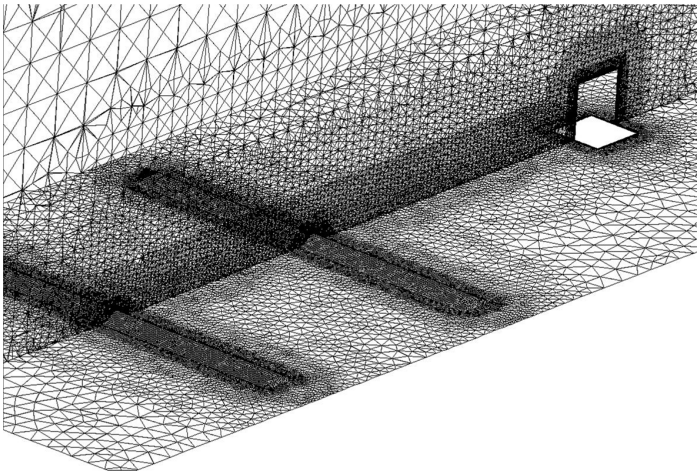


Figure 4.4: Use of rough objects for generating turbulence and their refinement requirement

figure that assuming a uniform mean velocity profile is an inaccurate assumption. Specifying the correct inlet boundary conditions is not adequate, because the quantities can change dramatically as they are transported through the wind channel. The numerical schemes used for solution of the discrete Navier Stokes equation, the mesh size, and the treatment of the boundary conditions (particularly the rough walls) are some influential parameters in how well the prescribed fields are preserved within the computational domain. Some practical recommendation for CFD wind simulations can be found in [35].

Transient modeling of the wind field involves more complexity since in addition to the mean fields, the vortices have to be modeled explicitly and as a function of time. A way to do so, is to use a very long numerical wind channel such that the velocity field develops itself towards a realistic turbulent wind. Inspired from the experimental wind tunnels, one can place roughness objects upstream of the obstacle to generate the desired amount of turbulence (figure 4.4). In general, modeling the process of turbulence generation numerically requires an extremely long computational domain, a high resolution throughout

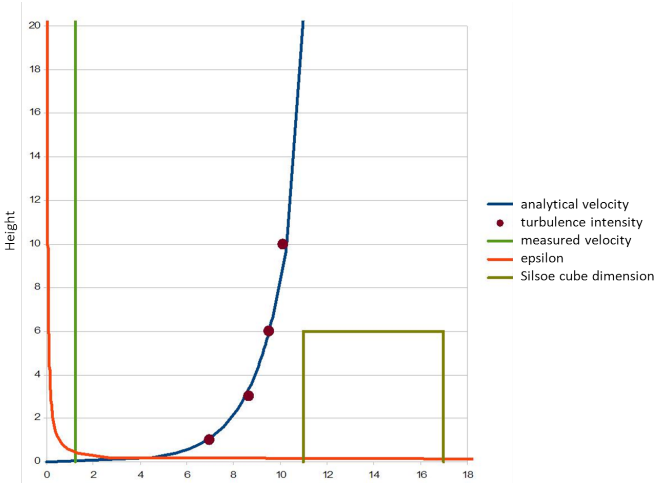


Figure 4.5: Comparison of the ABL profile in wind simulation with measurements [108]

the channel, accurate and non-dissipative numerical schemes, as well as a well tuned rough wall boundary condition. The difficulty in achieving such a model, motivates the use of so called numerical wind generators. [77] has proposed a wind generator based on the superposition of spatial velocity waves with different frequencies. This method is further applied in the numerical application presented in section 6.3. In the following paragraphs, the main idea of the method is presented. More details about numerical simulation of the wind field can be found in [24, 54, 111]. The numerical results of the Silsoe cube simulations are not reported here for the sake of briefness. A detailed report on the results can be found in [50]

The method is developed based on Taylor's frozen turbulence hypothesis since the time dimension is replaced by the space dimension in the mean stream direction. Taylor's hypothesis states that the time variation of a field variable ϕ (here velocity) is equal to the transport velocity multiplied by the space derivative of the field:

$$\frac{\partial \phi}{\partial t} = \tilde{u}_x \frac{\partial \phi}{\partial x} \tag{4.22}$$

The aim is to produce a turbulent velocity field which has similar second order statistics (e.g. variance) as the ABL. The wind field is generated in a virtual domain and in every time instance of the simulation t, the velocity at a cross section of the virtual wind domain at the position $x = \tilde{u}t$ is prescribed to the inlet boundary. The waves are generated in frequency domain and then Fourier-transformed to the space domain. Evaluation of the velocity components for each discretization point in the virtual wind domain is computationally not affordable, and hence three dimensional Fast Fourier Transform FFT should be used. For the wind generation in chapter 6.3.2 a very efficient FFT library is used, the details of which can be found in [61]. Note that FFT can efficiently be applied only for grids which are equidistant in all three dimensions. The contribution of each wave vector $\bar{k} = (k_x, k_y, k_z)^T$ to the velocity field is evaluated as a product of the $C(\bar{k})$ tensor calculated based on the atmospheric spectral density and a Gaussian stochastic complex vector $\bar{m}(\bar{k})$ bringing the randomness. The velocity vector $\bar{u}$ at each discrete location $\bar{x} = (x, y, z)$ on the wind domain mesh is calculated by the following Fourier transform [77]:

$$u_i(\bar{x}) = \sum_{\bar{k}} \mathrm{e}^{(\mathrm{i}\bar{k}.\bar{x})} C_{ij}(\bar{k}) m_j(\bar{k}) \tag{4.23}$$

$\sum_{\bar{k}}$ denotes the sum over all wave vectors. The C tensor is constructed as a function of wind bulk velocity, friction velocity, turbulence length scale, anisotropy ratios, etc. Details about estimation of the coefficients of C for isotropic as well as shear turbulence are described in [77]. This method is an effective means to produce transient atmospheric wind as the inlet condition for numerical wind simulations (figure 4.6). However, as mentioned, no matter how exact the properties of the wind at the inlet condition are, they changed through the CFD domain due to the discretization in time and space. Particularly, sustaining the high frequency oscillations throughout the domain requires a fine mesh and consequently small time step. Even tough accuracy of the wind velocity field at the inlet increases by the number of wave numbers considered in equation (4.23), using a finer mesh for the wind

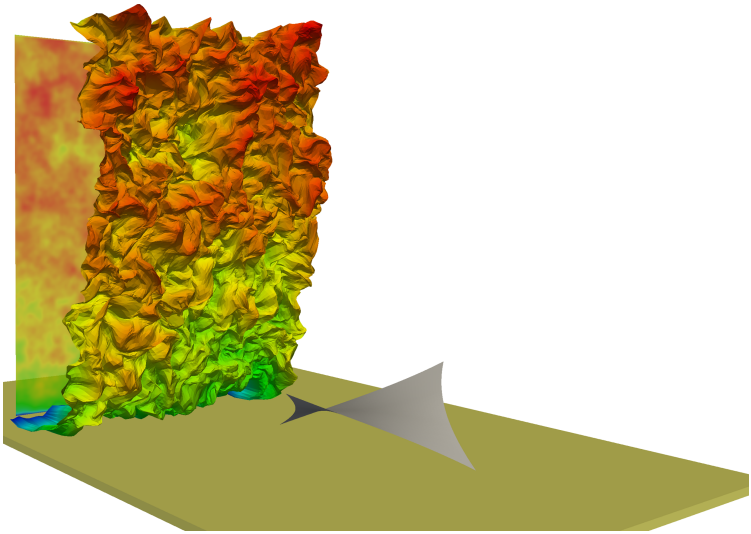


Figure 4.6: 3D structure of the generated transient atmospheric wind approaching the structure in the numerical wind channel (stucture size is exaggerated for demonstration

generation domain compared to the CFD domain would just waste the computation recourse, as the smaller waves will not be captured by the CFD mesh. Practically, one can tune the intensity of the generated turbulence in order to have the desired characteristics at the position of the obstacle in the CFD domain.

For the example shown in section 6.3 an in-house implementation of this method is used for generating the wind inlet boundary condition.

4.3 Sensitivity analysis

Surface sensitivity values form the gradient vector of the optimization problem and must be evaluated on every discretization point in node-based optimization. Dealing with many design variables, here the adjoint sensitivity analysis is preferred. Obtaining the surface sensitivity is often the most complicated part in CFD node-based shape

optimization. Unlike black-box sensitivity calculation methods such as finite differences, the adjoint method requires getting involved in the state equations and the numerical technique of solving them. Therefore, although sensitivity calculation is not the main topic of this project, its basic equations are introduced.

There are two general ways of numerically solving the adjoint of any "physical" state equation. The first way is to analytically derive the continuous adjoint equations based on the state equations, and then to discretize the adjoint equation for numerical solution. This way is the continuous adjoint method [56, 96, 131] . In the second way, the discrete adjoint method, discrete adjoint equations are directly obtained from the discrete state equations [6, 84, 89, 90]. The advantage of discrete adjoint is that the adjoint equations are always consistent with the state equations w.r.t. the numerical scheme, turbulence modeling, boundary treatments, etc. with almost no extra effort. On the other hand, since the adjoint equations have many similarities to the state equations, the CFD numerical framework can be easily adopted for solution of the continuous adjoint equations, whereas construction of the transpose state tangent Jacobian for discrete adjoint involves algorithmic and computational challenges which have to be tackled by techniques such as Automatic Differentiation (AD). Unlike structural optimization in which the discrete adjoint is almost always accepted as the method of choice, the discussion of continuous and discrete adjoint is still an ongoing topic between different scientific communities [88]. Here, opening this topic is avoided as it is not relevant. In the following paragraphs only the method used in this work, which is the continuous adjoint is described. Note that the implementation of the adjoint solver in the CFD code is provided by project partners as mentioned in section sec:software and is not done within this thesis.

According to the Boussinesq eddy-viscosity hypothesis, the incompressible steady-state RANS equations 4.8 can be written in the residual form as follows

$$\begin{aligned} R_{u_i} &= \tilde{u}_j \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{1}{\rho}\frac{\partial \tilde{p}}{\partial x_i} - \frac{\partial}{\partial x_j} 2\tilde{\nu}\tilde{\mathcal{S}}_{ij} \\ R_p &= \frac{\partial \tilde{u}_i}{\partial x_i} = 0, \end{aligned} \tag{4.24}$$

where

$$\tilde{\nu} = \nu + \nu_T. \tag{4.25}$$

$\tilde{\nu}$ is the effective viscosity which is the sum of molecular ν and turbulent viscosity ν_T [105]. For what follows the tildes used to define the turbulence quantities are omitted for sake of simplicity.

Generally, the objective function depends on the state variables $(\boldsymbol{u}, p)$ and on the normal to surface component η (figure 2.3). Note that η is not the design variable of the *Vertex Morphing* parametrization. The objective function is decomposed in two integrals, one over the fluid domain Ω and one over its boundary $\Gamma = \partial\Omega$ as follows

$$J = \int_\Gamma J_\Gamma + \int_\Omega J_\Omega. \tag{4.26}$$

In the applications presented in chapter sec:applications the following two objective functions are used: the power loss described in [92, 96] and the total drag. The power loss functional describes the difference in the energy flux between the inlet and the outlet of an internal flow and is formulated as:

$$J = -\int_\Gamma n_j u_j (p + \frac{1}{2}\rho u_i^2) d\Gamma, \tag{4.27}$$

where $n_j u_j$ is the normal component of the velocity. This functional is defined only on the boundary Γ which results to contributions of the response function only on the boundary conditions of the adjoint equations [130].

The drag functional is the sum of pressure and viscose forces applied on a wall boundary in a certain direction d_i (with unit length), and is evaluated as:

$$J = -\int_{\Gamma_w} (2\nu \mathcal{S}_{ij} - p\delta_{ij}) n_j d_i d\Gamma, \tag{4.28}$$

where Γ_w is the portion of the boundary on which the drag force has to be minimized. δ_{ij} is a three dimensional matrix with diagonal entities equal to one and off-diagonal ones equal to zero. Similar to the power loss function, the drag is defined only on the boundary.

In order to calculate the sensitivities of the response function w.r.t. the design variables, the Lagrange method is applied with the state equations as constraints. Initially the response function is augmented as follows

$$L = J + \int_\Omega (v_i R_{u_i} + q R_p) d\Omega. \tag{4.29}$$

The variables (v_i, q) are the adjoint variables of the problem with v_i being the adjoint velocity and q the adjoint pressure.

The total variation of the above functional $L = L(\eta, \boldsymbol{u}, p)$ should be zero which yields

$$\begin{aligned}\delta L &= \delta_\eta L + \delta_u L + \delta_p L \\ &= \delta_\eta J + \int_\Omega (v_i \delta_\eta R_{u_i} + q \delta_\eta R_p) \\ &+ \delta_u J + \int_\Omega (v_i \delta_u R_{u_i} + q \delta_u R_p) + \delta_p J + \int_\Omega (v_i \delta_p R_{u_i} + q \delta_p R_p) = 0\end{aligned} \tag{4.30}$$

Choosing (v_i, q) such that

$$\delta_u L + \delta_p L = \delta_u J + \int_\Omega (v_i \delta_u R_{u_i} + q \delta_u R_p) + \delta_p J + \int_\Omega (v_i \delta_p R_{u_i} + q \delta_p R_p) = 0, \tag{4.31}$$

the sensitivity of the response functional with respect to the surface normal direction η can be computed as

$$\delta L = \frac{\partial J}{\partial \eta} \delta\eta + \int_\Omega v_i \frac{\partial R_{v_i}}{\partial \eta} \delta\eta + \int_\Omega q \frac{\partial R_q}{\partial \eta} \delta\eta. \tag{4.32}$$

After substitution of the variation of the Navier-Stokes in equation (4.31) and integration by parts, the volume integral of the resulting equation results to the following adjoint equations

$$-\frac{\partial v_j}{\partial x_i} u_j - u_j \frac{\partial v_i}{\partial x_j} = \frac{\partial}{\partial x_j} 2\nu \mathcal{S}_{ij}(v) + \frac{\partial q}{\partial x_i} - \frac{\partial J_\Omega}{\partial u_i} \tag{4.33a}$$

$$\frac{\partial v_i}{\partial x_i} = \frac{\partial J_\Omega}{\partial p} \delta p. \tag{4.33b}$$

In case there is no volume contribution to the objective function, the partial derivatives $\partial J_\Omega / \partial u_i$ and $\partial J_\Omega / \partial p$ cancel out. This is the

case for most objective functions used in industrial applications like power dissipation and drag reduction. From the boundary terms of the resulting equation the adjoint boundary conditions are derived. The boundary conditions for the power dissipation objective function can be found in [92], while the respective boundary conditions for drag reduction are detailed in [123, 124].

Having computed the adjoint fields (v_i, p) the sensitivities are computed from equation (4.32) with further simplification detailed in [92] from the following relation

$$\frac{\partial J}{\partial \eta} = -\nu \frac{\partial v_i}{\partial n} \frac{\partial u_i}{\partial n}, \tag{4.34}$$

4.4 Fluid-Structure Interaction

Fluid-Structure Interaction FSI is a young topic in computational engineering whose goal is to model coupled systems involving both fluid and solid. The term FSI is used often when a both way interaction between the solid and the structure is considered in the model. Numerical FSI has various applications in biomedical engineering [34], aeronautics [40], sea structures [38], wind energy machines [9], etc. Light weight structures such as shells and membranes, when exposed to the wind loads, usually undergo large deformations. This deformation not only raises the need for the geometrical nonlinear modeling of the structure, but also changes the flow around the structure and, therefore, the fluid loads on the structure (non linear loading).

Despite the importance of consideration of the interaction between the light-weight structure and the fluid flow, the main reason for including this section in the text is that the workflow proposed in this work is built based on an FSI tool. The algorithmic similarities of these two problems will be discussed in chapter 5.

4.4.1 Partitioned FSI

The governing equations of the FSI problem are the same as the single field equations, i.e. Navier Stokes equations (equation 4.8) for fluid and the conservation of structural momentum. These equations are coupled

at the interface of fluid and structure through the interface stress tensor and the interface displacement z (or velocity) of the boundary:

$$\bar{u}_{f,i} = \frac{\partial \bar{z}_i}{\partial t} \tag{4.35}$$

$$\Upsilon_f \bar{n}_f = \Upsilon_s \bar{n}_s \tag{4.36}$$

$\bar{u}_f$ is the velocity vector at the interface (fluid boundary) Γ, z is the geometry vector at the interface as the unknown of the structural equation, Υ_f and Υ_s are the stress tensors at the interface for the fluid and structure field respectively, $\bar{n}_f$ and $\bar{n}_s$ the surface normal vectors of the fluid and structure field. The stress tensor is found by solving the flow equation. For instance, for a RANS turbulence viscosity model with Boussinesq's assumption, the interface fluid stress tensor is:

$$\Upsilon_{f,ij} = \tilde{p} I + \tilde{\nu} \left(\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} + \frac{\partial \tilde{u}_j}{\partial \tilde{x}_i} \right) \tag{4.37}$$

where I is the unit tensor. Note that the pressure $\tilde{p}$ contains both the hydraulic pressure as well as the volumetric part of the Reynolds turbulent tensor. Writing both state equations of the fluid F and the structural S in the residual form, the coupled FSI system of equation would be:

$$\begin{cases} F(w, z) = 0 \\ S(w, z) = 0 \end{cases} \tag{4.38}$$

where w refers to all fluid related variables, i.e. velocity, pressure and the turbulence related fields. The above nonlinear system of equations can be solved e.g. by Newton-Raphson iterations:

$$\begin{bmatrix} \dfrac{\partial F}{\partial w} & \dfrac{\partial F}{\partial z} \\ \dfrac{\partial S}{\partial w} & \dfrac{\partial S}{\partial z} \end{bmatrix} \begin{bmatrix} \delta w^{n+1} \\ \delta z^{n+1} \end{bmatrix} = \begin{bmatrix} F(w^n, z^n) \\ S(w^n, z^n) \end{bmatrix} \tag{4.39}$$

This approach of solving coupled problems is called monolithic and shown to be robust and efficient [39, 48, 52, 133], particularly in strongly coupled problems [68]. However, both structure and fluid equations should be solved in the same software, and also the discretization schemes should allow for forming the coupled system of equation. Furthermore, the coupling coefficients (off diagonal terms in equation (4.39)) have to be explicitly modeled, which is usually not straightforward. The alternative is the fixed point iterations in which the single field equations are solved separately and consecutively. The fluid quantities provide a Neumann boundary condition for solution of $S(z) = 0$ and interface motion is given to the flow equation $F(w) = 0$ as a Dirichlet boundary condition. This is called partitioned approach for solving FSI problems [29, 31, 101, 102, 141]. In many real size engineering examples the partitioned approach is preferred as it gives more freedom to the modeling decisions in the single field solvers. The partitioned FSI method is much easier to be applied on general engineering problems, and therefore it has received a lot of industrial attention lately, aiming to solve the FSI ptoblem by the use of black-box single field software. For the same reason, and in order to tackle FSI problems with strongly coupled fields there have been many coupling algorithms suggested with improved stability and efficiency [21, 22, 67, 136].

In general there is no necessity for single field spatial discretizations to coincide at the interface. In such a case a mapping operator between the non-matching interface meshes is required. Note that almost never an analytic description of the interface surface is available, and therefore the discrete approximated surfaces of the two fields should directly communicate with each other. In chapter 5 this communication and the mapping operator for non-matching grids is compared to the mapping between the geometry and control field in *Vertex Morphing*.

4.4.2 FSI shape optimization

If the flexibility of the object exposed to fluid flow cannot be neglected, the influence of FSI should be considered in the optimization as well. [126] shows that neglecting the coupling effects can even cause a wrong direction in the surface sensitivity. No mater what definition the objective function has, as long as it is a function of state variables

(almost always), a coupled optimization framework is required. This is because the state equations and consequently the state variables are coupled, thus naturally the response surface is the result of the interaction between those two fields.

Of course, the geometrical aspects of shape optimization of coupled system are the same as of single field optimization, and therefore the *Vertex Morphing* can be used as shape control in FSI as well. However, what makes gradient based optimization of FSI very difficult is the sensitivity analysis for the coupled state equations. If the adjoint method is desired the adjoint coupled equation system including the off-diagonal coupling terms has to be constructed:

$$\begin{bmatrix} \dfrac{\partial F}{\partial w} & \dfrac{\partial F}{\partial z} \\ \dfrac{\partial S}{\partial w} & \dfrac{\partial S}{\partial z} \end{bmatrix}^T \Psi = \begin{bmatrix} \left[\dfrac{\partial F}{\partial w}\right]^T & \left[\dfrac{\partial S}{\partial w}\right]^T \\ \left[\dfrac{\partial F}{\partial z}\right]^T & \left[\dfrac{\partial S}{\partial z}\right]^T \end{bmatrix} \begin{bmatrix} \Psi_w \\ \Psi_z \end{bmatrix} = \begin{bmatrix} \dfrac{\partial J}{\partial w} \\ \dfrac{\partial J}{\partial z} \end{bmatrix} \tag{4.40}$$

Ψ is the adjoint variable of the coupled FSI problem and Ψ_w and Ψ_z the adjoint variables related to the fluid and structure fields respectively. Looking at the left matrix in equation (4.40) one understands that if the monolithic coupled Jacobian matrix of equations (4.39) is available, calculation of the FSI adjoint is not very complicated. Out of the few successful implementations of FSI adjoint problem, most of them applied the monolithic formulation and used the transpose of the full matrix including the cross terms, $\dfrac{\partial F}{\partial z}$ and $\dfrac{\partial S}{\partial w}$. However, in a partitioned FSI framework which is usually the only choice for general industrial size cases the solution of equation (4.40) is not anymore trivial. The diagonal terms of the middle matrix, $\left[\dfrac{\partial S}{\partial z}\right]^T$ and $\left[\dfrac{\partial F}{\partial w}\right]^T$ are the single field adjoint operators which can be available in many software. The problematic parts are the off-diagonal terms which are in a fixed-point iteration solution never evaluated. Their effect is modeled by exchanging boundary conditions (force and boundary motion) in every iteration. Particularly calculation of $\dfrac{\partial F}{\partial z}$ which indicates the

effect of boundary motion on the residuals of the flow equations is very complicated, as it includes the mixture of the grid motion and Navier-Stokes equations in a typical ALE-based FSI [26]. Therefore the black-box coupling algorithms of FSI cannot be extended to adjoint FSI, and more elaborate coupling techniques is required [23, 30, 78, 79, 81]. A successful FSI adjoint optimization work based on the same workflow presented in this project can be found in [126].

As an alternative to the adjoint method, one can perform direct sensitivity analysis. This can be done for instance by global finite differences. The other approach would be to solve the direct coupled sensitivity equation system:

$$\begin{bmatrix} \frac{\partial F}{\partial w} & \frac{\partial F}{\partial z} \\ \frac{\partial S}{\partial w} & \frac{\partial S}{\partial z} \end{bmatrix} \begin{bmatrix} \frac{\partial w}{\partial s_i} \\ \frac{\partial z}{\partial s_i} \end{bmatrix} = \begin{bmatrix} \frac{\partial F}{\partial s_i} \\ \frac{\partial S}{\partial s_i} \end{bmatrix} \tag{4.41}$$

The advantage compared to the adjoint system is that the tangent matrix of equation (4.41) is the same as the original FSI problem, and not transposed. Therefore the same coupling approach as of the FSI can be used for coupling the sensitivity fields $\frac{\partial w}{\partial s_i}$ and $\frac{\partial z}{\partial s_i}$. However, for every design variable s_i a separate coupled sensitivity equation has to be solved.

In this section shape optimization of a coupled wind-structure interaction problem is presented. The geometry of a membrane roof subject to wind loads has to be improved, for a lift-drag combination objective function [50]. As shown in figure 4.7 the roof is supported by two stiff frames at the sides. Wind blows with the ABL profile described in section 4.2.2 at a speed of $9.6\frac{m}{s}$ at the tip of the roof. The simulation is done by steady RANS $k-\omega$ SST model on a tetrahedral mesh. The side frames are modeled as rigid beams and the roof itself by pre-stressed membrane elements. the two ends of the roof are connected to two pre-stressed cables.

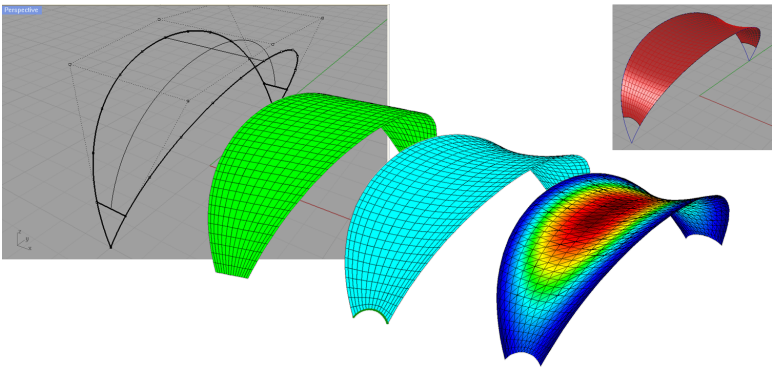
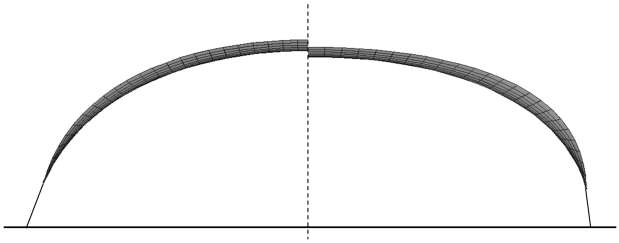


Figure 4.7: Parametrization steps for design and sensitivity analysis

The geometry is parametrized through a chain of design parameters, such that there exist only two design variables for the optimization problem. The side frames are described by NURBS functions and the location of the control points are the design variables. Note that the height of the middle control point is kept constant so that the overall height of the structure remains almost constant. Moreover, considering the symmetry in two directions as well as restricting the motion of the control points to the same plane as of the curved frame, the shape of the frames can be controlled by a few design variables. The pre-stress is kept constant throughout the optimization and therefore the doubly curved membrane surface is found by Form-Finding [13, 71, 140]. In order not to limit the design space to small shape variations, an in-plane shape regularization step is included in the design chain [129]. In such a parameterization, the whole geometry and the mesh follow variations of the design parameters "smoothly", even for very large amplitudes.

The gradient vector is evaluated by solving the coupled direct sensitivity equations (4.41) using fixed point iterations (partitioned). The single field sensitivities are evaluated decoupled and by finite differences. As seen in figure 4.8, the coupled shape optimization successfully reaches the minimum by applying a slight geometry modification.



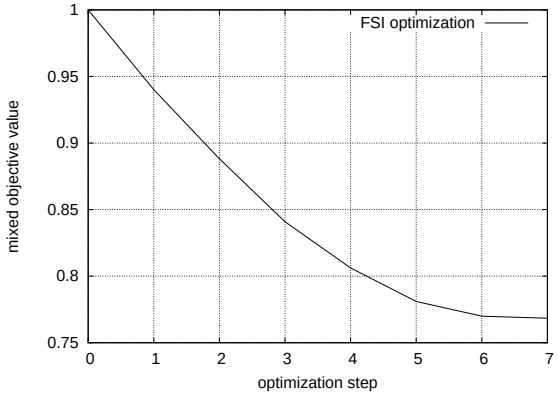


Figure 4.8: Top: comparison of the initial shape (left half) and the optimized one (right half). bottom: objective reduction history for coupled optimization of the membrane roof

Chapter 5

Optimization workflow

The workflow shall provide the software platform as well as the data management for CFD shape optimization of large problems. A discretized (external or internal) CFD case is given to this workflow as the input. A part or the whole wall boundary of the flow domain is considered as the design surface. This surface is parametrized by the *Vertex Morphing* and shall be improved. In this section, some details of the shape optimization work flow are explained.

Figure 5.1 shows the different modules and their sequence in the optimization loop. In the right box which represents the CFD environment, first the state primal equations are calculated, 'c'. Based on that solution and in 'd' first the adjoint equations are solved, and then the surface sensitivity is evaluated in the design surface. Following the chain rule of sensitivity, in the middle box the gradient vector of the optimization problem is found. The design update is transformed to surface deformation, again through the chain rule. The updated surface mesh of 'a' is further sent to the volume mesh motion module, 'd'. At this stage the new geometry and discretization are ready to be passed to the CFD box for the next optimization iteration.

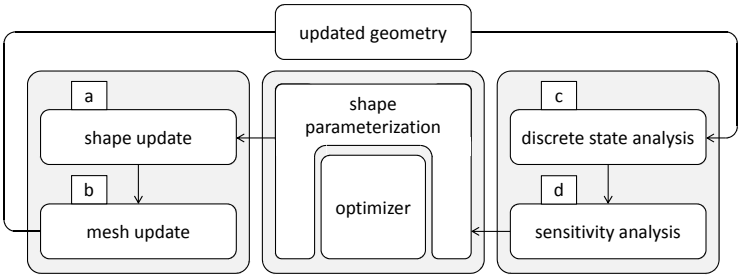


Figure 5.1: Design optimization modules and their algorithmic sequence

5.1 FSI and optimization algorithms

The optimization workflow used for this work has been build upon a Fluid-Structure Interaction workflow, due to their similarity, as mentioned briefly in section 4.4. Additionally, this decision makes extending the software environment for optimization of coupled problems easier [126]. In fact many of the optimization routines are taken one to one from the FSI code, even though at the first glance they seem totally different problems. Indeed from the physical view point there is almost no relation between these two problem types, because the shape optimization is the inverse problem of finding the optimal shape whereas FSI deals with the direct numerical modeling of a physical phenomenon. However from the algorithmic view point, in both problems, the unknown is the surface of the fluid domain which keeps it in equilibrium with an external interaction module. This external module, which is the solid domain in the case of FSI and is the optimizer in the case of shape optimization, sends out a response, in the form of boundary deflection, and as a function of fluid quantities. This change in the shape of the fluid domain changes the fluid quantities and therefore the response of the external module. The goal is to find the equilibrium state between these interacting domains, which would mean the force equilibrium for FSI and zero gradient for optimization. Although the

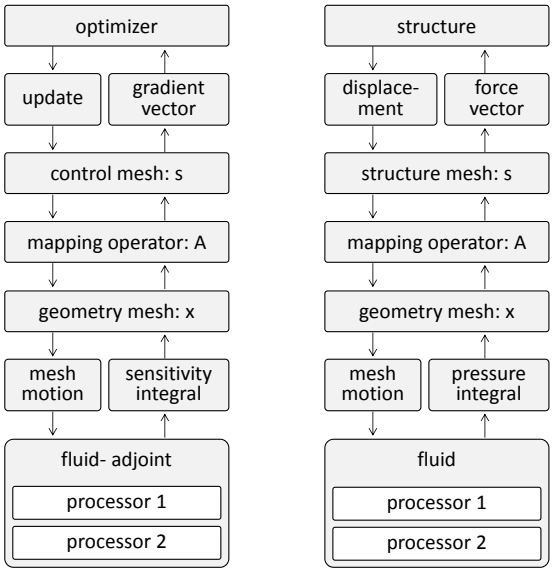


Figure 5.2: Comparison of data flow in shape optimization (left) and FSI (right)

flow quantities exchanged at the interface are different, the response of a filtered node based optimizer to the surface sensitivity values is very similar similar to the response of a structure to the surface stress. As mentioned in chapter 2.3, even in some node-based optimization methods, exactly the stiffness matrix of an imaginary structure at the design surface is used as the filtering operator. This subject fits very well with the design dynamics interpretation of shape optimization [66]

If we carefully observe, the role of the control field in *Vertex morphing* has again a large algorithmic similarity to the non-matching structural mesh in an FSI problem. The data exchange for non-matching grids in FSI takes place through a mapping operator. This operator, which plays the role of the A operator in *Vertex morphing* can be based on

linear interpolation, NURBS functions, Radial Basis Functions (RBF), etc. The mentioned fact is visualized in figure 5.2.

In some references [66, 70] design optimization is compared to a dynamic process, in which the time dimension is replaced by the design evolution dimension. Based on that the term "design velocity" is introduced, which indicates the rate of change in the design for each design parameter. Similarly, other shape optimization quantities can be seen equivalent to structural dynamic quantities such as inertial, damping and internal forces. Even though the type of design optimization in the current project is well suited to this interpretation, we intentionally do not mention the equivalent interpretations to avoid verbosity and confusion.

5.2 Software

The workflow is designed completely modular and with clearly defined interfaces. This eases exchanging the current CFD and/or adjoint solver with an alternative software more suitable for the specific application, even for other physical simulations such as structural and thermal analysis. There exist two separate executables which run in parallel. The inter-code communication is done by Message Passing Interface (MPI). Note that due to the large problem size, the CFD and adjoint code run always in several processors which communicate through MPI as well. As it can be seen in figure 5.3, the interface between the optimization and the CFD codes are only the scalar vectors of surface sensitivity and surface deflection. In order to avoid extensive unnecessary communication costs, the volume mesh motion is done internally in the fluid code, in every processor. Since the two modules treat the other one as a total black box, reprogramming of the interface communication logic for a new CFD solver is straightforward.

The optimization core is programmed in C++ and based on the in-house research code Carat (Technische Universität München, Lehrstuhl für Statik). The implemented optimization code in this work is object-oriented and enhancing new algorithms and techniques would require a small programming effort. Shape control and optimization modules

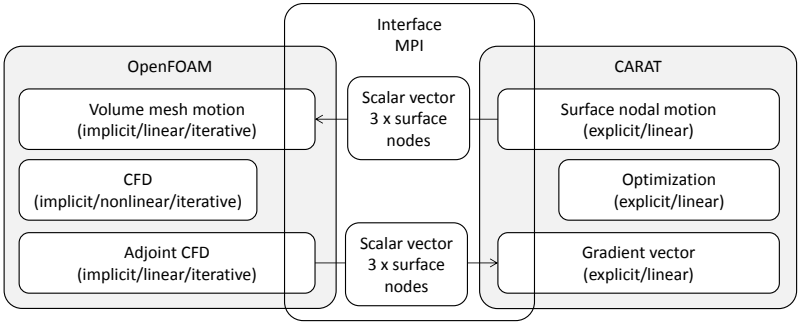


Figure 5.3: Software packages for the optimization loop and their interaction

are implemented matrix-free and vector containers are treated with sparsity consideration. A considerable challenge in evaluation of design variables is the neighbor search and distance evaluation. Here, the neighbor search is done by use of octrees for the sake of efficiency [76].

For the flow simulation, a finite volume based continuous Navier-Stokes adjoint solver, implemented in OpenFOAM [138] is used [92, 96]. The adjoint solvers are implemented and provided by the project partners ICON Ltd and Engys Ltd, and in collaboration with Volkswagen. The adjoint codes and the examples used in this work are partly developed within the "7th Framework Programme" EU-project, FLOWHEAD (Fluid Optimisation Workflows for Highly Effective Automotive Development Processes). For the turbulent S-duct test case (section 6.2.1) a fully adjoint evaluation of the Reynolds averaged Navier-Stokes equations with Spalart-Almaras turbulence model is used [145], which

is provided by ICON Ltd. The main developments in OpenFOAM were done on the interfacing and geometrical aspects such as mesh deformation. Therefore we avoid getting deeper in the details of CFD implementation.

The surface sensitivity is calculated normal to the surface for each face on the design surface. These face values shall be interpolated on the surface points. The point surface sensitivity is the scalar product of the interpolated sensitivity value and the unit normal vector. Calculation of the normal vector requires an interpolation of the neighboring face normal vectors as well. Here, the sensitivity value and normal vector are both linearly interpolated with a weighting proportional to the inverse face area of the neighboring faces.

After updating the fluid surface mesh based on the displacement field sent by Carat to OpenFOAM, the volume cells are adapted by solution of a Laplace equation with non-uniform diffusivity [59]. In this mesh motion method, the boundary displacement is diffused through the fluid domain with a diffusion coefficient proportional to the inverse of the distance to the wall boundary. A key aspect of this implementation is the use of flow fields from the previous optimization iteration. This means that despite the modified geometry and discretization, the finite volume cells are initialized by the flow quantities of the last iteration which leads to a much faster convergence of the equations. Starting the solution from a constant initial field e.g. due to remeshing in each optimization iteration, would increase the computational cost one or even two orders of magnitude.

5.3 Optimization algorithm

Defining the shape optimization problem as a geometrical evolution of the design in several small increments, here the steepest descent algorithm is selected. However, in *Vertex Morphing* the optimization is solved in the design space, and the search direction is already enhanced by some approximation of the Hessian matrix. The response surface of the type of problem aimed in this work includes many non-convexities. This is due to the physics of the problem and nature of the PDEs.

On the other hand, the large number of design variables makes the evaluation or even approximation of the Hessian very expensive [97]. From the geometrical point of view and as explained before, during the optimization, the surface coordinate system on which the design variables are defined varies. Hence, usage of optimization history information for the calculation of the new design step would require evaluation and storage of the deformation gradient of every surface coordinate, in each step. The choice of steepest descent makes the optimization simple, history independent and needless of higher order gradient information. Combination of small step size and many design variables (design modes), allows the optimization process to explore the design space fast and freely. But if desired, a line search can be used in order to reduce the number of optimization iterations. Note that almost the whole computation time is spent on the iterative solution of the primal and adjoint equations, and since the solution of every optimization step is used as the starting point of the next step, smaller perturbation in the geometry results in smaller computation time. Therefore, the optimization cost is proportional to the total length of geometry change and is almost independent of optimization iterations. So, an elaborate line search or more complex optimization algorithm does not improve the performance much.

The accurate calculation of the state and the adjoint variables can require many iterations of the solution algorithms. However, the general surface sensitivity pattern is obtained much before the total convergence of the primal and adjoint fields. Considering the fact that the gradient vector of the optimization is normalized for calculation of the update step, one does not need to perform the solution iterations until a perfect convergence, and a reasonable approximation of the gradient vector is good enough to find the search direction for the small increment of each optimization step. Note that taking several small progressive steps towards the optimum gradually corrects the slight inadequacies existing in the sensitivity calculation. Performing the shape update before the total convergence of the equations can speed up the optimization significantly [47, 95]. Note that in typical industrial CFD optimization applications, evaluation of adjoint sensitivities takes up to 2 (or even 3) orders of magnitude more time than the shape control.

Chapter 6

Applications

The shape parametrization method and the optimization workflow presented in the previous chapters are designed to work for very large 3D examples robustly and efficiently. This claim is tested through several applications in this chapter. First some basic conceptual examples in 2D and 3D are studied, so that the feasibility of the results and the quality of the improved shape can be easily judged. After that, a series of large size 3D applications are presented. In order to give an order to this chapter, those examples are divided into two sections, automotive industry and structural design, even though from the optimization view point there is no difference in their treatments.

In all the presented test cases the computational time for the geometry control (interpolation of shape derivatives, neighbor search, construction of design variables, calculation of design and geometry update vectors, inter-code communication, mesh motion, etc.) is below 5% of the computational time for the solution of CFD (primal and adjoint). Note that the simple 2D examples can be successfully optimized with other filtering techniques as mentioned in the text. However, the large examples are more difficult to be compared to other works, since there

is no available node-based publication for this type of problem, with large shape variations and high geometry complexity.

In all the application a Gaussian filter is used. The filter radius is defined as the distance of the center of the kernel to the point where the magnitude of the Gaussian function is 10% of the center peak. The value of the filter function is set to zero outside the filter radius. Moreover, the integral under the filter function is set to 1 by suitable scaling.

There are several examples shown and in order to avoid repeating explanations about all parameters for every example, in each group of applications the discussion is focused on a certain aspect and other factors are not outlined. In the conceptual examples, section 6.1, where the goal is to demonstrate the node based optimization by use of *Vertex Morphing*, the details of the physical analysis or the shape optimization are not mentioned. In the automotive applications, section 6.2, the effectiveness of the method on industrial examples is to be studied. However, the CFD calculation has been treated almost as a black box, which is provided by the simulation experts of that field. Therefore the focus is put on the optimization aspects, such as smoothing, objective improvement rate, influence of the filter size, influence of the design space, preservation of the feature lines, quality of the mesh, etc. Discussions about the details of CFD modeling, such as the near-wall treatment, turbulence model and steadiness assumption are left for the last section, 6.3, in which the whole design and optimization chain is presented in details with no black box. That section contains comments about the structural design, wind simulation, adjoint calculation, optimization and physical analysis of the improved shape.

6.1 Conceptual examples

Three conceptual examples are presented. The first two are 2D with a single wall boundary in an infinity large space. The third example is a 3D shape optimization in a half space boundary flow. In these examples, except the regions that the design surface is connected to other boundaries, the whole surface is subject to shape variation.

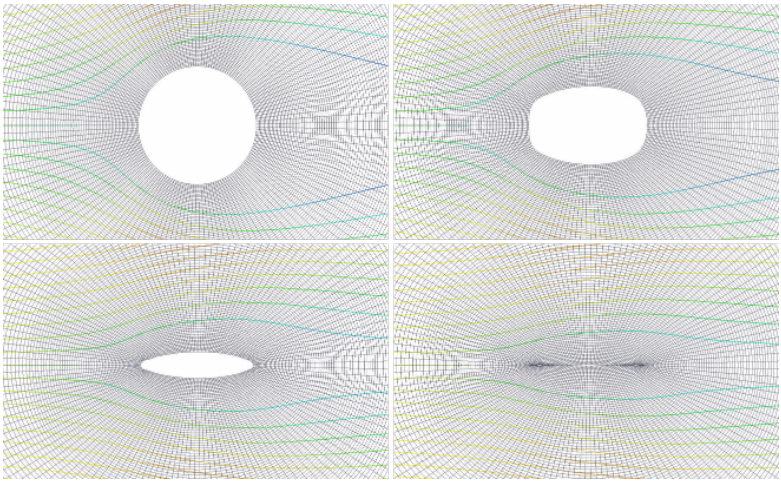


Figure 6.1: The circular cylinder transforms to a flat plate for drag minimization. Top left: initial design, top right step 35, bottom left: step 70 and bottom right step 100.

6.1.1 Drag on circular cylinder

The drag force caused by the laminar flow around on a circular cylinder has to be minimized by modifying the shape of the cylinder. In OpenFOAM, 2D geometries are modeled as one layer of finite volume cells. The thickness of this cell layer is chosen to be larger than the filter radius, so that there is no interaction between the nodes of the two layers. This setup is perfectly equivalent to a 2D simulation and the model remains homogeneous in the third dimension until the end of the optimization.

Since no geometrical nor physical constraints exist, the optimal solution is trivial. The cylinder should disappear so that the drag force is reduced to zero. However, representing such a large change in the geometry variation requires a proper treatment of the geometry as well as the mesh. Otherwise, the irregularity caused by the large number of design variables deviates the process to divergence. The optimization is performed in about 100 small steps and with a filter, 10 times smaller

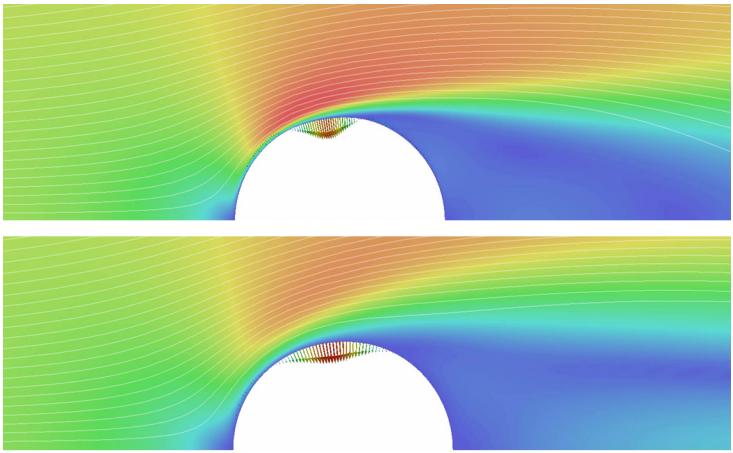


Figure 6.2: Geometry update mechanism for Reynolds number 1000 (top) and 100 (bottom)

than the diameter of the cylinder. As seen in figure 6.1 the cylinder collapses to a flat plate smoothly and with no geometry nor mesh irregularity.

Ideally, the flat plate should also shrink in the stream direction and disappear in order to vanish the viscous forces on the object as well. However, when the geometry is flat, the normal to surface shape update does not lead to the shrinkage of the plate. In such a case the geometrical sensitivities as the partial (no total) derivative of the objective function w.r.t. the geometry $\frac{\partial J}{\partial \overline{x}}$ play the main role. The geometrical sensitivities are present when the objective functional is defined on the optimization surface [115]. The physical interpretation is straightforward as well. Assuming the pressure and velocity fields to be frozen, changing the area in a surface element changes the drag force, as the integral of normal and shear stresses applied on the surface. This part of the gradient vector is neglected in the adjoint CFD solver used in this work. The influence of geometrical sensitivity to the shape derivative is proportional to the inverse of the curvature radius (equation (2.51)), which is very large at the two ends of the flattened curve.

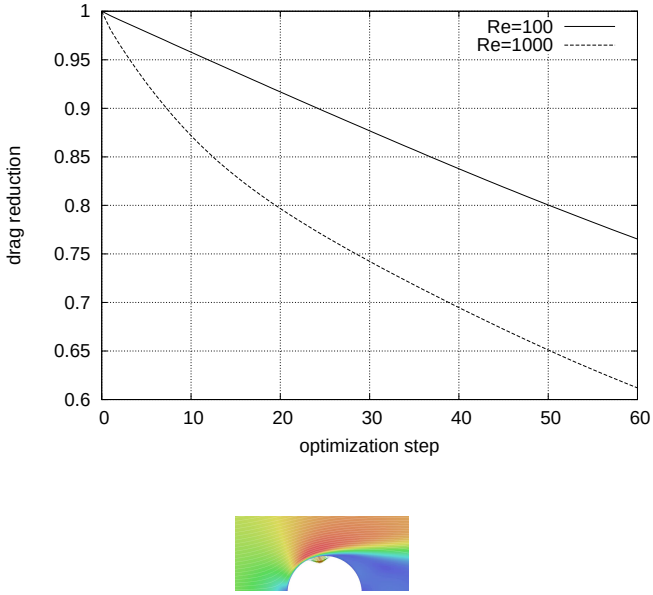


Figure 6.3: Normalized drag (objective) history for cylinder drag minimization

Despite the fact that the optimal shape for minimization of drag is trivial, there are interesting aspects about the evolutionary steps that the geometry undergoes towards this optimum. Depending on the flow regimes around the cylinder, different surface sensitivity distributions are to be expected and hence different behaviors in the optimization process. Figure 6.2 compares the update pattern on the surface of the cylinder for two different Reynolds numbers, i.e. 100 and 1000. Due to the symmetry half of the domain is presented. In figure 6.3, the optimization history of the first 60 steps for those two cases can be seen. The high Reynolds number case, which contains more non-linearity and complexity in the flow behavior, has a more non-linear improvement curve as well. Consequently, a proper modification in its geometry can cause a larger improvement in the objective. It should be mentioned that if a CAD-based shape optimization was used, in contrast to the presented results of figure 6.2, the limited number of shape parameters

would prevent the detailed shape modes in the update vector. As the result, the shape changes would take place in more or less similar manner for both flow regimes.

6.1.2 Lift on airfoil

The lift maximization in a symmetric airfoil (NACA 0020) is studied. Note that shape optimization of airfoils is an old, well establish and complex field in aerospace engineering. The presented simple test case does not contain the necessary considerations of a realistic airfoil design, and is not intended to represent a complete and industrially usable shape optimization of airfoils. The only purpose is to demonstrate the usability of *Vertex Morphing* in the aeronautical applications.

After availability of flow adjoint solvers, node-based shape optimization of airfoils was studied extensively as an alternative to the classical spline parametrized airfoils which could be optimized by zero-order methods. The large design space makes it possible to achieve more and more efficient geometries which cannot be captured by polynomials. The fact that node-based methods can represent sharp edges in the geometry is an other positive property for this type of optimization. The shape and curvature variations in such problems are usually limited. Therefore, there is a high chance that the optimizer finds the same optimal design independent of the filtering coefficient. Unlike the large size examples presented in this work, aerodynamic optimization is usually performed until the perfect convergence of the design. For such a case, it is important to choose an optimal value for the filtering coefficient, such that the solution is found in as least steps as possible. Alternatively, the filtering coefficient can be included as a design variable in the optimization problem. The step size is usually chosen by an approximated Newton line search. In contrast to this work, most of published works on airfoil shape optimal design use implicit smoothing. Moreover, considering the limited geometry variation, in-plane regularization has a smaller importance compared to other applications.

The optimization is performed in 36 constant steps, and zero angle of attack. As seen in figure 6.4, the lower edge of the airfoil is gradually flattened and the upper edge is curved. This produces a higher total force in the upward angle. In parallel, the whole body is rotated

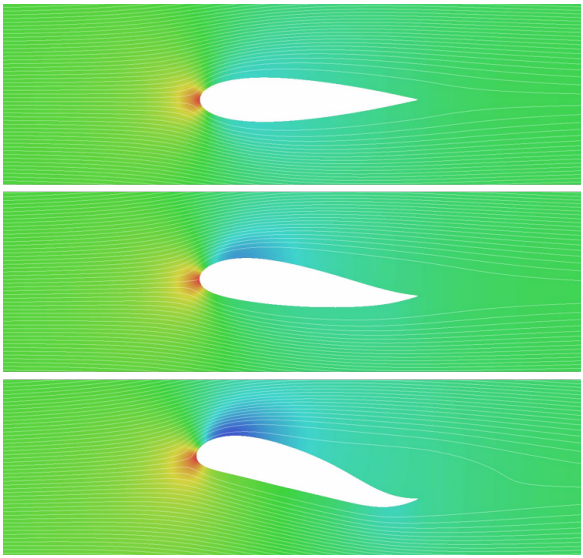


Figure 6.4: Shape evolution of an airfoil with lift maximization as objective

clockwise, so that the streamwise momentum of the flow produces lift force. The lift is zero at the beginning due to the symmetry (figure 6.5). As the lift increases, the increase of the effective area increases the drag force. Note that there is no constraint on drag enforced and therefore continuing the optimization until the convergence would not lead to a meaningful airfoil.

6.1.3 Drag on 3D Ahmed body

The main advantage of topology optimization is that one can get an optimal design without having a deep insight to the physics of the problem and the response of the system. In many cases the solution of a topology optimization is not predictable due to its enormously large design freedom. By use of node-based shape optimization, together with a strong shape control technique, one can partially create the mentioned advantage in a shape optimal design as well. Considering the fact that compared to topology optimization, shape optimization has a lower computational cost and a better definition of the geometry,

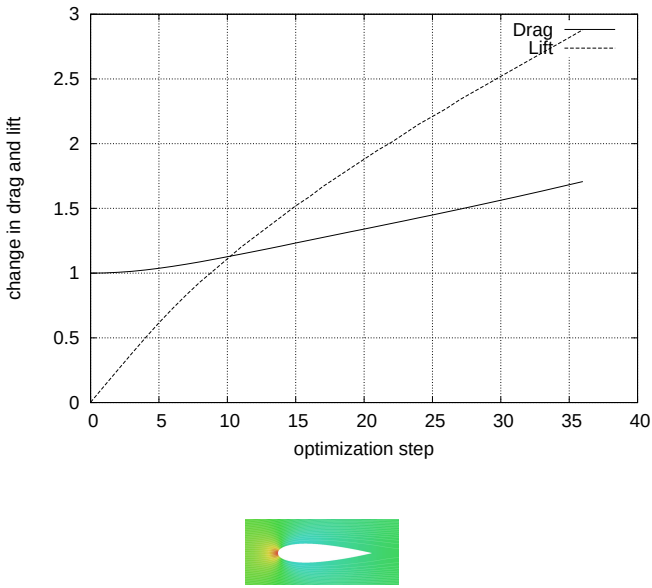


Figure 6.5: Drag and lift history for the airfoil case. Both drag and lift values are normalized by the drag value of the initial geometry

its combination with a large and free design space forms the most robust and efficient tool for very large problems [12].

The shape of the 3D Ahmed body, which is a well known benchmark geometry for bluff body simulation (specially car aerodynamics) is subject to change, in order to reduce the total drag on it. The geometry is just a box with a fillet round edges in front (figure 6.6 right) and a chamfer at the upper edge of the back side. The box is connected to the ground by four small vertical cylindrical stands (reminding of the wheels of a car). The object is located on the slip boundary of a half space. Since the body is slightly lifted by the stands, the fluid can flow in all 4 sides of it. This is an extremely simplified version of an external aerodynamic simulation of a vehicle.

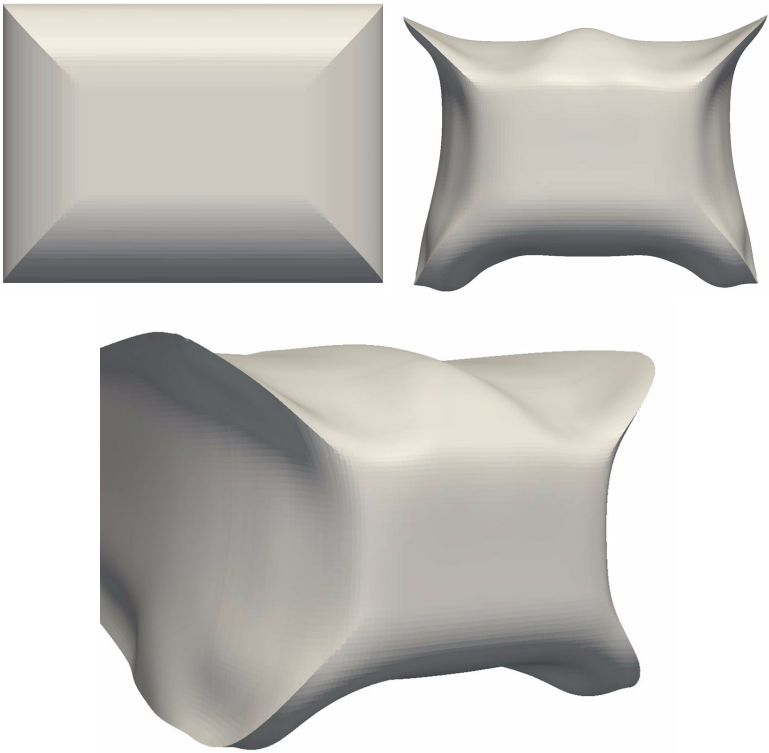


Figure 6.6: Drag reduction on 3D Ahmed body (top left: initial shape)

Here we try to explore the possibility of moving towards the goal mentioned in the first paragraph of this section. The edges of the Ahmed body, including the round ones in front, are the design features which shall remain sharp (section 3.1.6) and the large flat surfaces between those lines are supposed to deform in order to lower the drag. There is no external constraint applied on the feature lines and as mentioned in section 3.1.6 they are treated as the other surface points. The only constraint is that the four connections to the stands shall stay in their position. An octree automatically generated volume mesh is used and the surface mesh contains all sort of elements (triangle, quadrilateral, and polygons). There exist around 275000 surface coordinates in the design space.

The optimization is performed intentionally with a relatively large filter radius (around on sixth of the width), so that no small scale details are formed. The improved shape after some steps can be seen in fig 6.6, top-right and bottom. As expected, the sharp corners stay in the geometry and the flat surfaces deform such that the effective area is reduced and also the shape is aerodynamically smoother. The fixed parts at the bottom are connected with a smooth transition to the rest of the body. It is interesting that there is a small bump formed at the top and also a much smaller one at the bottom. The curvature around and behind the stands also gives an organic sense to the structure. All in all, it can be concluded that this method has the capacity of transforming very basic geometries to physically efficient and aesthetically valuable industrial designs. Obviously a more realistic example would require more elaborate considerations.

6.2 Automotive applications

This section verifies the usability and applicability of the method in real size automotive problems with 3D complex geometries. The geometries are chosen from a series of test cases provided by Volkswagen AG in the framework of a collaboration project, as well as the "7th Framework Programme" EU-project, FLOWHEAD (Fluid Optimisation Workflows for Highly Effective Automotive Development Processes). Therefore, they reflect the up to date challenges and needs in CFD shape optimal design in vehicle industry. The first series of examples is power loss reduction for an internal flow, and the second series is the drag reduction for external aerodynamic of a full car model. Table 6.1 shows some information about these cases.

6.2.1 S-duct

This example is a part of an air duct of the car air conditioning system. The shape of the bend has to be modified in order to reduce the air power loss, which is described in equation (4.27).

The geometry and the boundaries are not symmetric and the flow structure inside the duct has a fully 3D form (figure 6.7). Shape

case	volume cell type	surface coordinates (3× design variables)
laminar S-duct	tet	$25K$
turbulent S-duct	hex	$226K$
side mirror, scenario 1	tet	$32K$
side mirror, scenario 2	tet	$20K$
side mirror, scenario 3	tet	$11K$
upper body Passat	tet	$780K$
full body Polo	poly	$3.3M$

Table 6.1: Key information about the automotive examples

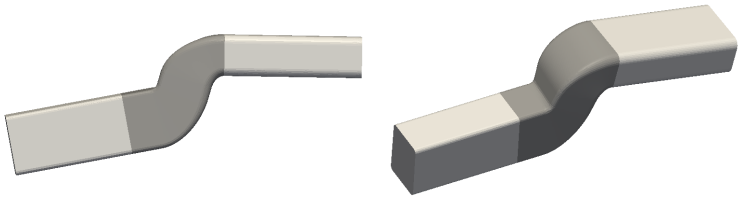


Figure 6.7: Initial geometry of the S-duct. The design surface is plotted darker.

optimization is performed in two flow regimes, a laminar case with $Re \sim 300$ and a turbulent case with $Re \sim 3000$ (table 6.1).

Laminar 3D S-duct

Shape optimization was performed for forty iterations. After that the sensitivities are very small and the objective does not vary much. Evolution of design can be seen in figure 6.8 and diagram 6.9. The shape is smooth throughout the optimization and each design instance can be used as an improved shape. Until step 20, at which around 25% of improvement is achieved, the shape variation is mainly due to the proper reduction of curvature at the bend, which would be expected by an experienced designer. However, further improvement of the design, which leads to a non-intuitive free-form shape (figure 6.8) can be done only by applying the surface sensitivity to a rich shape parametrization together with mesh regularity consideration.

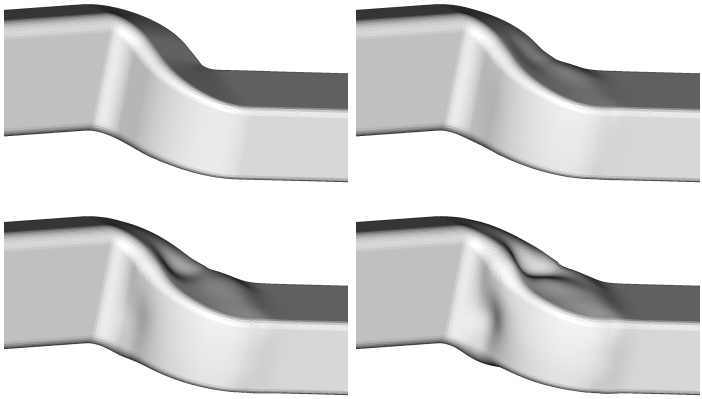


Figure 6.8: Evolution of the S-duct shape from the initial design, at top-left, to top-right, bottom-left and bottom-right

Figure 6.10 compares a cross section of the initial design with the one of the improved shape. During the second half of the optimization process, as the optimizer is exploring innovative design updates for non-trivial further refinements, several update patterns and even consecutive changes in the curvature sign happen. Considering the large spatial gradients of surface sensitivity, combined with the curved geometry of the duct, in-plane treatment of the mesh plays a key role, to the extent that a normal to surface update strategy would not survive more than few steps. Figure 6.11 compares the surface mesh of the initial design with the improved shape, and as it can be seen, despite the large design variation, the mesh is almost as good as the initial one, according to the OpenFOAM mesh quality measures.

Turbulent 3D S-duct

Compared to the laminar duct, the air flows faster in this test case. Hence, a relatively large separation region is formed, which dissipates energy. This is a common phenomenon in many engineering CFD problems. At the separation point, there is a sudden transition of surface sensitivity, from a large positive value to a large negative value, streamwise. This means that pulling out the wall boundary before the

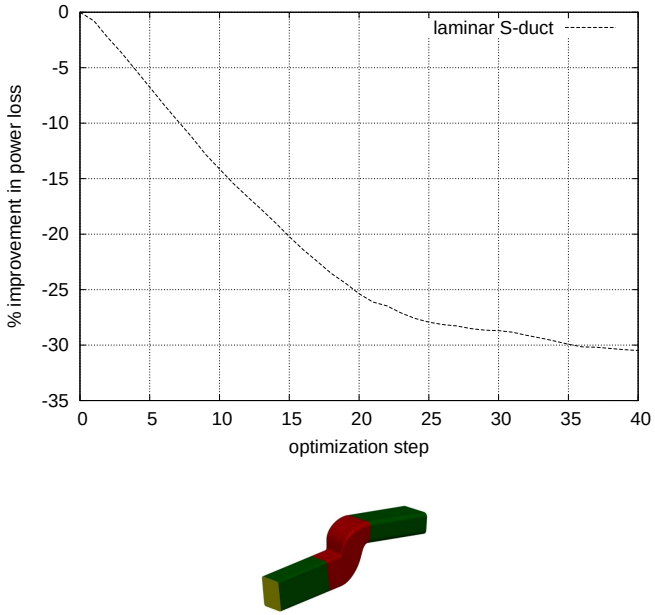


Figure 6.9: Optimization history for the laminar S-duct case

separation and pushing it in at the separated region can reduce the dissipated power. In the case of flow over a curved boundary, this is equivalent to flattening the curvature, on which the flow detaches from the wall. Update vectors of the surface points at the first optimization step are seen in figure 6.12. The sudden change of surface sensitivity sign has been transformed to the smooth change of the update direction by the use of *Vertex Morphing* method. This update pattern would not be feasible in a simple CAD parametrization of the bend geometry. The advantage of the proposed method becomes more clear as the optimization loop goes on. Updating the shape by the vectors shown in figure 6.12 reduces the sharpness of the curve and moves it slightly to the right. In the next optimization step, there will be a similar update pattern on the moved curve, and thus further movement of the curve. This procedure produces a "design wave" from upstream to downstream, as the shape evolves towards removing the separation.

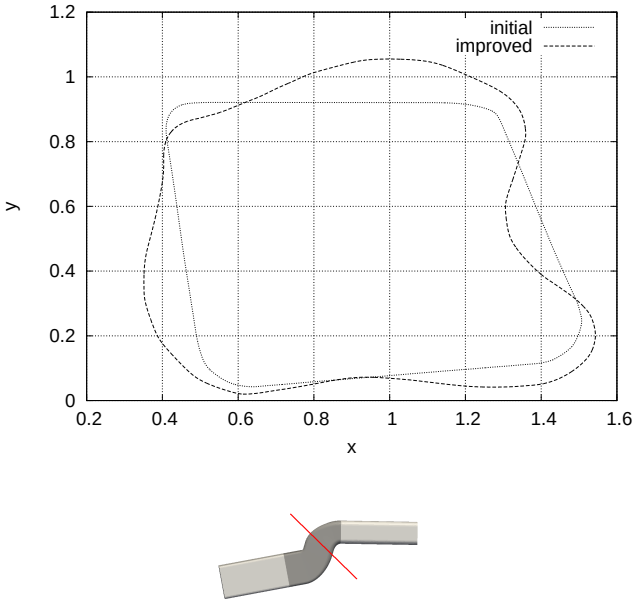


Figure 6.10: Cross sectional plot of the laminar S-duct geometry before and after shape optimization

Representation of this design wave requires an enormous number of shape modes, which is well provided by the presented method.

Additionally, as it can be seen in figure 6.12, the in-plane component of the update vector explained in chapter 3 prevents any mesh distortion within the complex shape evolution.

In 60 optimization steps, an improvement of 70% in power loss has been achieved. After the first 50% of improvement, the geometry looks very similar to the initial design, with some improvement on the curved parts. For the last 20% of improvement however, the shape undergoes considerable free-form modifications, as in the laminar case.

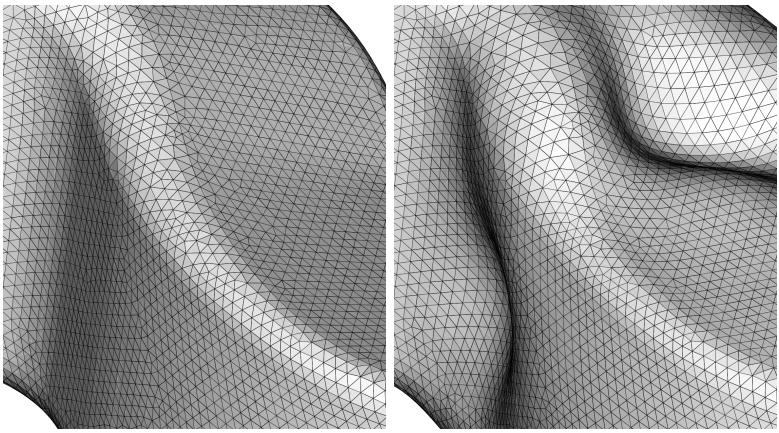


Figure 6.11: Preservation of the surface mesh density and quality after large shape and curvature variation of the laminar S-duct

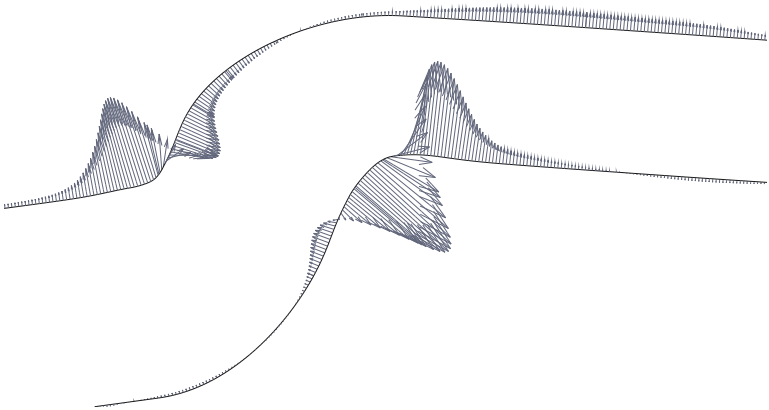


Figure 6.12: Smooth in and out plane update vectors (scaled) at flow separation regions at a longitudinal cross section of the turbulent S-duct. The design surface is larger compared to the laminar case. The fluid flows from left to right.

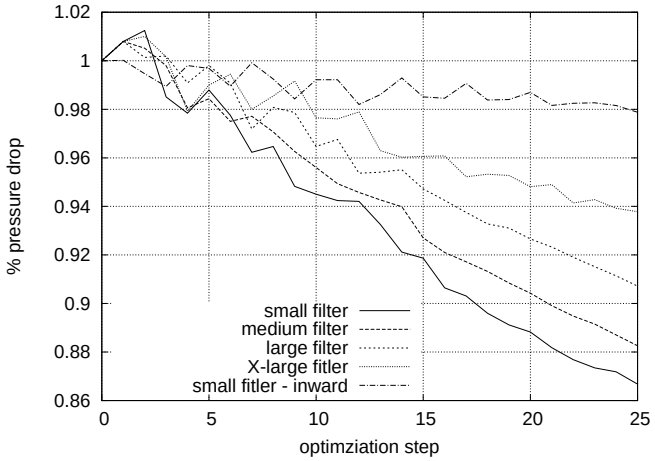


Figure 6.13: Optimization history for the inlet manifold

6.2.2 Practical aspects

Here we review some practical aspects which have been frequently observed in automotive optimization cases. The optimization history of a ducted flow is used for demonstration. The case is a shape optimal design for an inlet manifold, with a relatively limited design surface compared to the S-duct. We do not present more details about the description of the case, for the sake of briefness.

In steady CFD simulations, it is common to set the solution tolerance according to the level of accuracy required by the specific study. When tolerance is not small enough, the objective value shows an oscillation with domains up to few percents of its value. Of course these oscillations do not have any physical meaning and are result of the poor level of convergence in the non linear solution procedure. This simulation might

be further used for shape optimization. Having in mind that in real life industrial optimization, the order of improvement is very often only few percents, building up the optimization loop upon such a simulation might seem senseless. However, as practically proven by several test cases, it is possible to start the optimization even with a poor level of convergence. As mentioned in section 5.3, since the overall distribution of the surface sensitivities is formed much before the final convergence of the primal and adjoint fields, the update pattern would not be much different compared to a perfectly converged case. This approximate update strategy is absolutely enough for the first optimization steps. If a more precise and detailed optimization is needed, one can achieve the better level of convergence during the optimization by choosing a smaller tolerance for the sub-step CFD iterations. This would save a considerable amount of computation and set-up time. The diagram of figure 6.13 clearly shows the mentioned process. Despite the oscillatory objective value at the beginning, the optimization is moving to the right direction. Further on, the oscillations gradually disappear as the tolerance of the optimization process is smaller than the one set for the initial simulation.

As discussed in the example of figure 3.2, smaller filters allow for faster approaching to the optimal geometry, specially if the optimum includes high curvatures. This is also observed in real test cases, e.g. figure 6.13. Usually due to production limits, aesthetics, etc. a more smooth design with less geometrical details is desired. This can be fulfilled by applying a large filter. But even if there is no limitation about the minimum curvature and detail size, the filter cannot be smaller than a certain limit for stability reasons. Practically, the filter should encircle "enough" surface points, in order to guarantee the regularity. Therefore, despite the fast rate of convergence, a very small filter might end up to divergence, before the desired level of improvement is achieved. A practical solution to overcome the question of the smallest possible filter size, is to define the filter size based on the surface element size, rather than a fixed distance.

Due to packaging and dimension restrictions, it is often the case that the part subject to optimal design cannot get bigger (smaller in case of external flows). Many parts of the car exterior as well as some interior

parts should strictly remain withing the initial design dimensions. The proposed optimization technique uses surface coordinates, so it is not possible to directly impose the packaging as a constraint. Furthermore, solving the problem by a constrained optimization algorithm would dramatically decrease the efficiency and robustness, and hence it would not be anymore usable for the presented challenging cases. A pragmatic way to impose the dimension constraint is to penalize the outward movement of the surface points. In fact, instead of the normal motion, the positive values of the control parameter s are penalized, which leads to a smooth design. Note that there is no guarantee that a non-positive value for s would not cause any outward motion of the geometry x. However, practically it is almost never the case that a point moves outward and even if it does, it will be extremely small and perfectly negligible. It is clear that such a restriction would strongly decrease the improvement level (figure 3.2). However, it is still very valuable to gain few percents of improvement by making a duct smaller or an external object larger.

6.2.3 Car body

In this section, the shape optimization for a car body model is desired. Although in all presented cases the full car body is modeled, in each case a different region of the car exterior is allowed to deform, as the design surface. In the first test case, the whole upper body of a Volkswagen Passat (figure 6.14) is regarded as the design space. Next, the entire body of a Volkswagen Polo, including the under body is optimized. Further on, the geometry of the side mirror of Volkswagen Passat (figure 6.14) is improved in three different scenarios based on the design surface. In all the cases the objective is the total drag on the full car body. Table 6.1 contains some information about these cases.

Compared to the ducted flow, this problem is much larger in computational size and includes more geometrical complexity. One important aspect in shape optimization of the car body (and many similar industrial designs) is preserving the main features of the design. More precisely, changing the shape should not affect the “design character”, aesthetic and geometrical features, such as sharp edges must be kept during the optimization. To this end, it is desired to use large enough filters such that the improvement pattern is smooth, and the overall

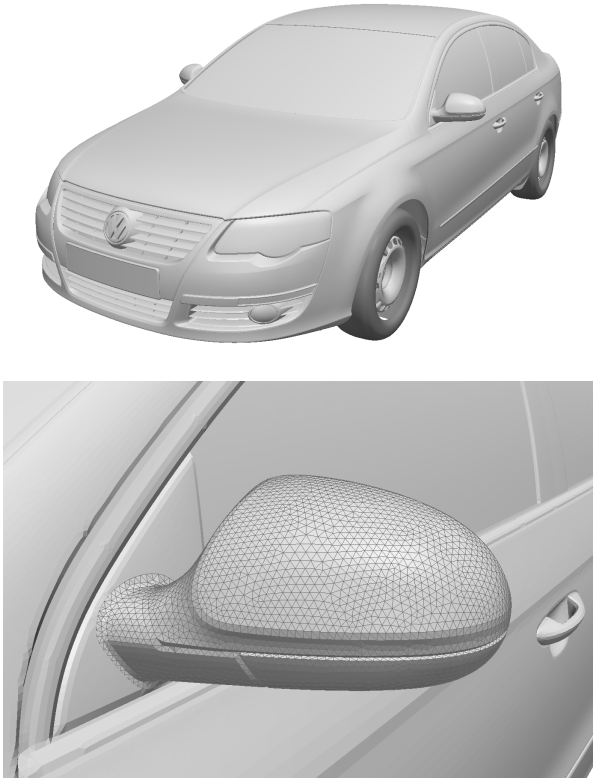


Figure 6.14: Side mirror geometry and discretization (bottom). Car body geometry (top)

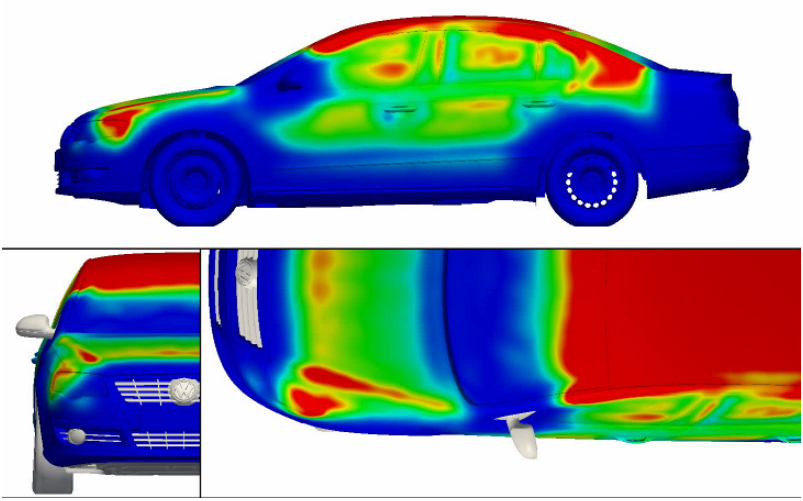


Figure 6.15: Update length distribution on Passat upper body. Blue indicates zero update length and red the maximum step size in each optimization iteration, here $2mm$.

shape of the vehicle is not disturbed. Note that the oscillations seen in the optimization history diagrams of this example are numerical and due to the large tolerance in solution of flow equations (section 6.2.2) and have nothing to do with the shape optimization.

Passat upper body

Even though optimization of the whole upper body of a car is not of practical interest, it is chosen as a challenge for the *Vertex Morphing* method and the designed workflow. Here, there exists a large number of design variables and various geometrical complexities, such as high curvatures, trimmed surfaces, sharp corners, regions of low quality mesh, etc. Gray regions in figure 6.15 (bottom left and bottom right) are the fixed regions. The optimization has been run for several steps without any smoothness or mesh regularity problems and within 20 steps, with maximum update length of $2mm$ per step, 1.6% improvement in drag has been achieved (figure 6.16). Fig 6.15

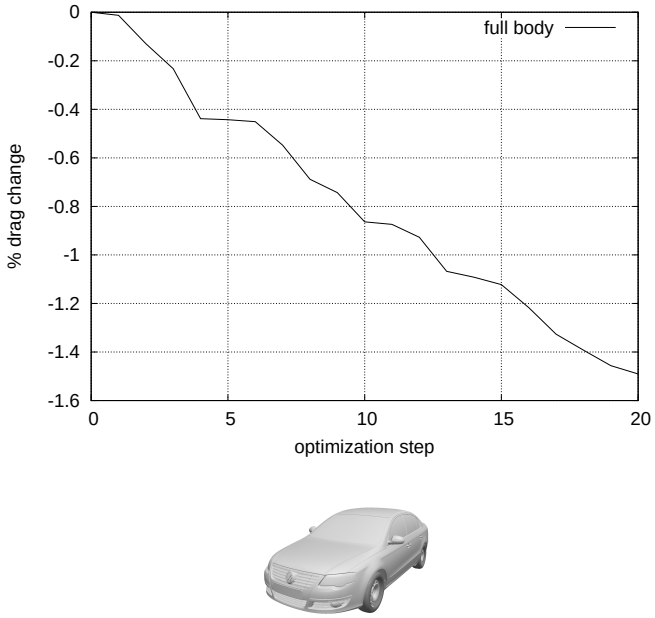


Figure 6.16: Optimization history for the upper body

shows the update vector length distribution on the surface of the car for one of the first optimization steps. In this case a filter radius of $10cm$ is used which is 20 times larger than the surface mesh edge length.

More important than the trivial drag reduction by lowering the size of the car, is the preservation of the car feature lines, which is provided automatically by *Vertex Morphing*. All the features smaller than the radius of influence are only subject to a bulk and rigid motion, without considerable shape deformation. Even after several design updates, the car body has all the design characteristics of the original one. Another crucial aspect in problems of this size is the computational time. Due to the explicit nature of the method and the efficient data exchange, the whole optimization time for the 20 steps has been almost the same

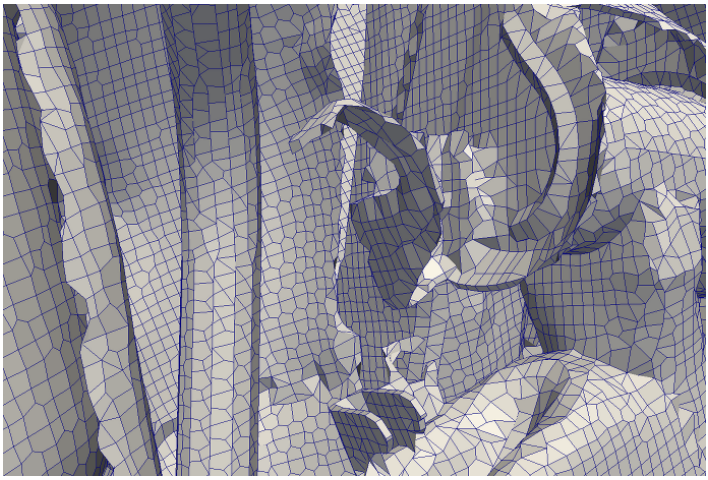


Figure 6.17: Surface mesh of an automatically generated mesh for small geometrical details

as the CFD and adjoint computation time. Note that in such a problem talking about the optimum is not meaningful and as mentioned before the aim is to apply limited geometrical improvements to the shape. The aerodynamic efficiency is one of the many important factors in designing the car body, and hence it is not possible to directly use the result of this type of optimization in car body design. However, the deformation patterns in individual regions can provide the designers with a very valuable guideline in improving the physics of their design.

Polo entire body

In a more extreme test case, the full car body, including the under body, suspension system, etc. is subject to shape optimization. The mesh is an automatically generated octree based polyhedral mesh which includes severe distortions. Particularly at the under body where the CAD model contains tiny objects, various kinds of mesh irregularities exits, for instance, elements laying on each other, highly skewed polygons, edges with almost zero length, etc. In such a case, in contrast to the smooth surface of an airfoil, the poor definition of the surface makes it practically impossible to use a topology based implicit smoothing.

Applying separate in-plane mesh regularization would immediately fail as well.

There exist more than 3.3 million shape coordinates which are controlled during the optimization. This is a very large number compared to the typical shape optimization problems. The only fixed region is the bottom of the tires where the road surface is touched. Similar to the upper body case, many optimization steps are performed successfully and the objective value decreases monotonically. Having efficiently solved such a problem, optimization of specific regions of the car body would be only a simplified version of this one, w.r.t. size and complexity.

Side mirrors

In a more industrially relevant case, the geometry of side mirrors has been improved to reduce the drag of the complete car body, in three different scenarios. In scenario 1 and as the first study, the shape of the whole mirror, including the glass part is optimized. In scenario 2 and 3, the cover of the mirror is subject to shape changes, with the difference that the scenario 3 has a more limited design surface. The design surface of each scenario can be seen in figures 6.18-6.22. More information about these cases can be found in table 6.1. In all three scenarios, the shape of both side mirrors is improved simultaneously, but individually, and since the geometry of the upper body is almost symmetric, the shape optimization has resulted in symmetric shapes.

The optimization of the whole mirror, scenario 1, was performed in 16 steps, with a maximum update length of $1mm$ per step. A relatively slight modification in the geometry of the side mirror reduces the drag on the mirror by 7% (figure 6.18). However, the objective function of the optimization problem is the total drag (and not only the mirror drag). Therefore, the shape modification aims in reducing the overall drag, which includes the effects on the downstream flow around the rest of the car as well. In this case, the wind load on the rest of the car body is also reduced by 0.2%, which brings a total drag reduction of 0.6% (figure 6.18).

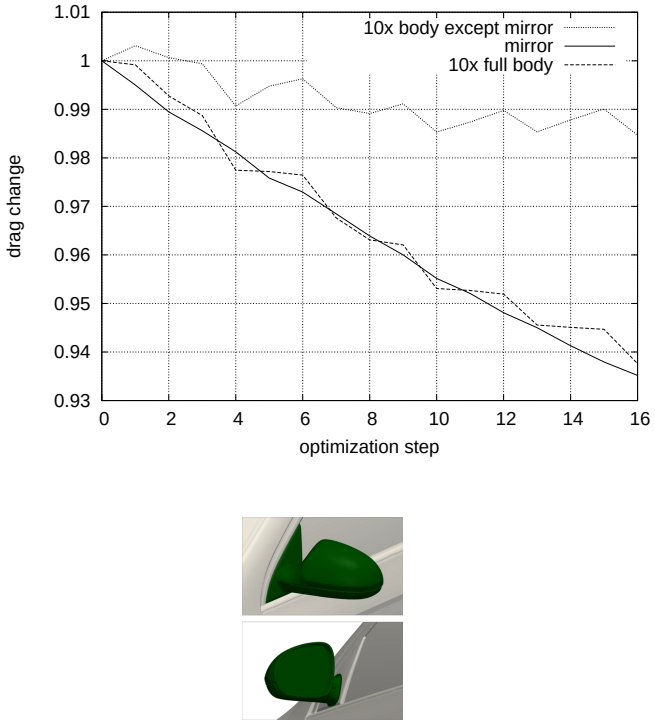


Figure 6.18: Optimization history for the side mirror, scenario 1. The design surface is plotted in dark green.

In scenario 1, the glass part was allowed to deform, but due to its small surface sensitivity it remains almost flat. However, a large portion of improvement is obtained by the shrinkage of the object, which is obviously favorable for drag reduction. By excluding the glass part and its frame from the design surface (scenario 2), automatically the size of the glass is constrained. With such a constraint, simply shrinking the mirror cover would worsen the aerodynamic behavior and hence the shape optimization is not trivial at all. The front view of the initial design compared to the improved one can be seen in figure 6.20. The top part of the mirror is reduced in size and a small valley shape is

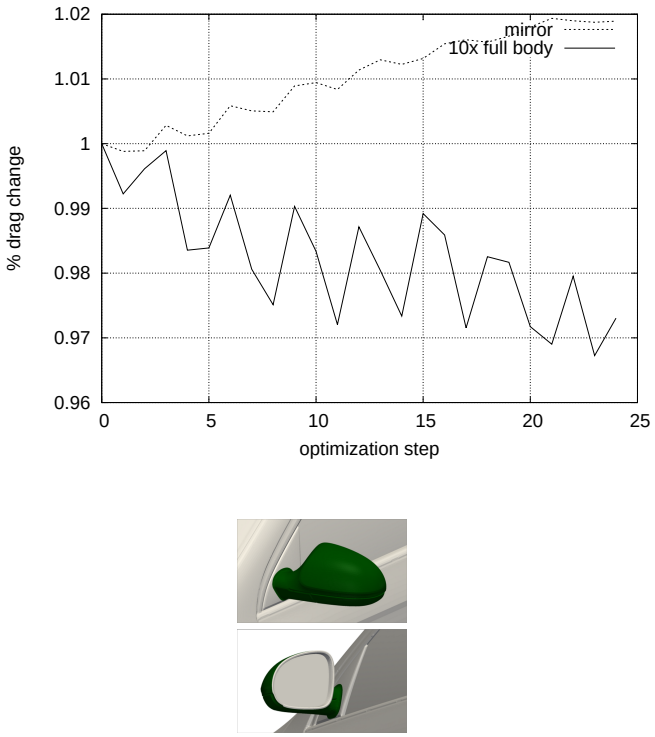


Figure 6.19: Optimization history for the side mirror, scenario 2. The design surface is plotted in dark green.

formed at the neck of the mirror. The improved shape has an organic flavor and is elegant, which is maybe the most important requirement for car body design.

The diagram of figure 6.19 has a very interesting message about this case. The improved mirror covers cause a higher drag value on the side mirrors (around 4%), although the objective value of the optimization problem, the total drag is lowered by more than 0.2%.

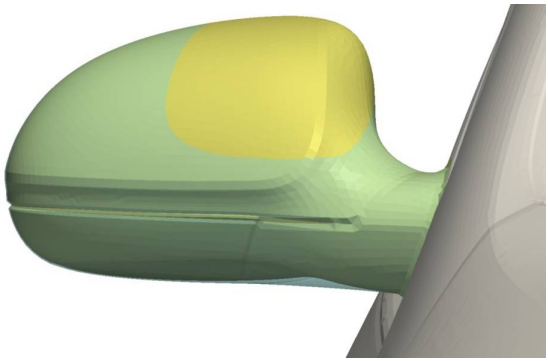


Figure 6.20: Side mirror, the intial geometry (transsparent blue) vs. the improved one (solid yellow), scenario 2

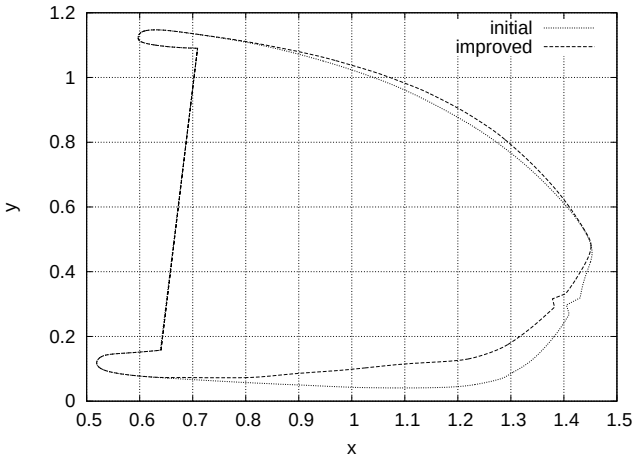


Figure 6.21: A cross section of the car side mirror geometry before and after shape optimization, scenario 2

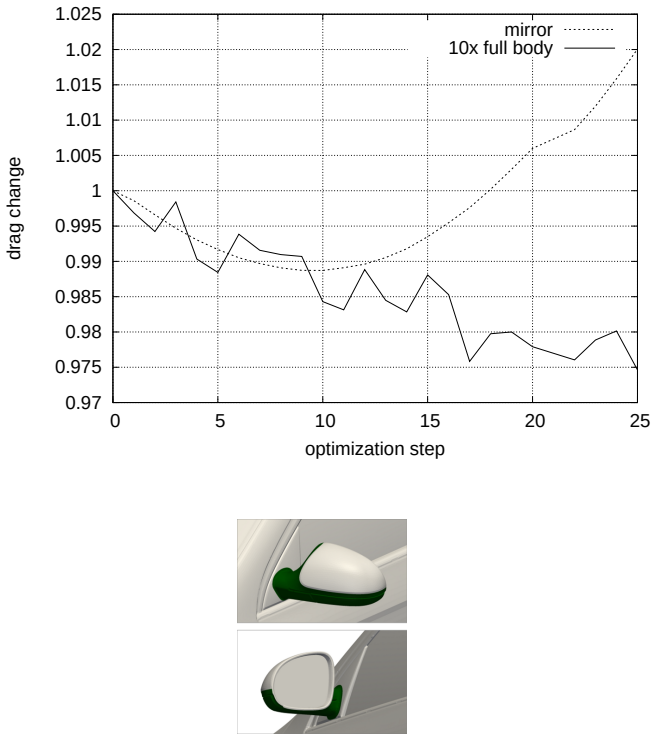


Figure 6.22: Optimization history for the side mirror, scenario 3. The design surface is plotted in dark green.

Figure 6.21 compares a cross section of the mirror at the initial design and the improved shape. In this figure, at around $(x, y) = (1.4, 0.3)$, a design feature line of the mirror is seen. According to the same figure, this feature line is perfectly preserved in the improved geometry, and is not smoothed out.

In scenario 3, the design surface is a narrow patch with fixed boundaries, which gives a limited space for improvement. The only region that the shape can deform a bit is the part close to the car body. Because

of that, the influence of the downstream effects on the rest of the car is even more pronounced and is actually the main improvement mechanism (figure 6.22).

6.3 Structural design

The final example is shape optimization of a large structure, subject to atmospheric wind. The purpose of the example is to explore applicability of the workflow to wind engineering problems. First, there will be some brief remarks on computational modeling of the case. Second, the shape is optimized for two different objectives and various aspects about the physics as well as the geometry are discussed.

The structure is an imaginary thin and light weight roof over a tennis stadium (figure 6.23). The roof has a simple geoemtry and is used only as a demonstrating example. However, it has been tried to include some of the key features in its design both from architectural and structural point of view. The CAD modeling is done by use of NURBS surfaces in the toolbox Rhino. Each layer of the roof (top and bottom) consists of left and right high order NURBS surfaces with C^0 continuity at their junction in the middle. Lower and upper faces are connected at the sides by flat surfaces. The lower surfaces are trimmed by the intruding cylindrical columns.

The structure has a free form doubly curved geometry for efficient load carrying. It stands on four thick columns at the corners which span a distance of $140m$ and $50m$ in width and $40m$ in depth. The highest point of the roof is $35m$ above the ground and the lowest elevation which is at the position of the columns is around $6.5m$. In order to hold the bending moments, the middle front part of the roof is the thickest region with $5m$. The thickness is gradually reduced toward the side edges down to $1.3m$.

Such a light weight structure has a big capacity for structural shape optimization due to its size and free form geometry, which is out of our main focus in this work. However, the wind force is certainly the largest load applied on this structure and plays an important role in

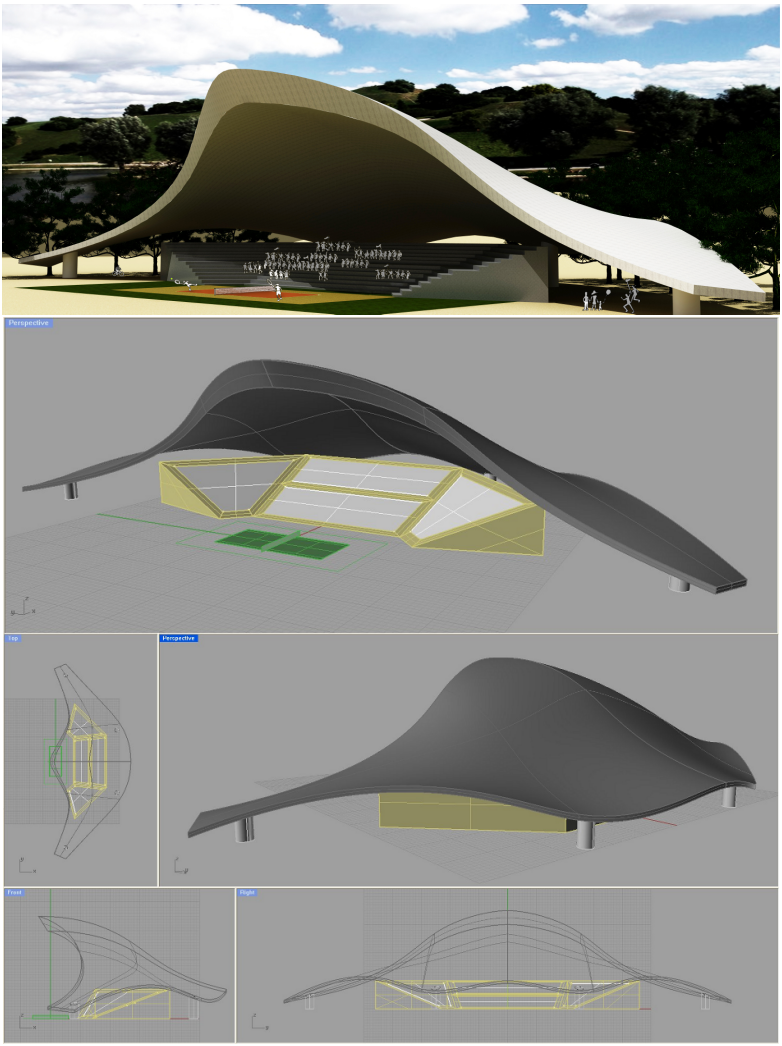


Figure 6.23: CAD design of the tennis stadium roof

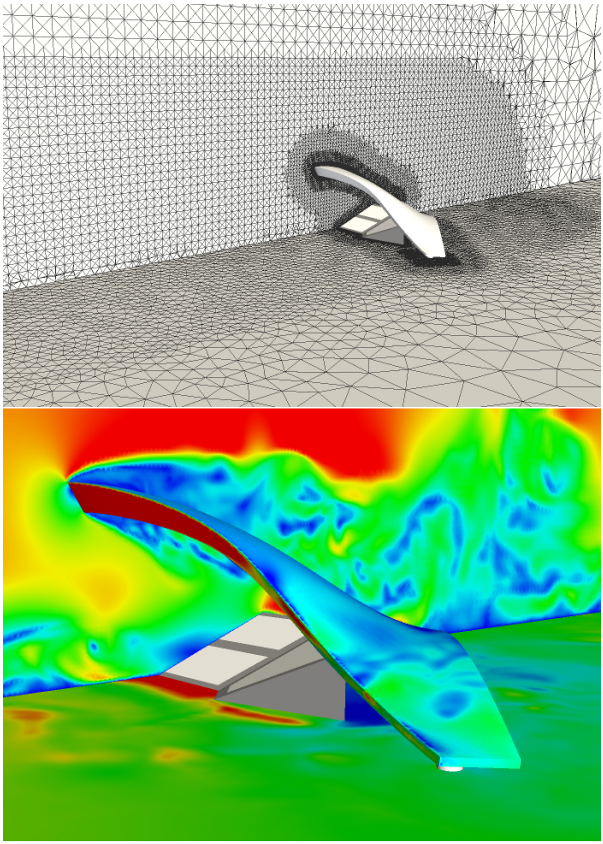


Figure 6.24: Fine mesh region from the inlet to the obstacle for simulations with turbulent wind generator (top). Velocity at the lateral mid section, pressure on the wall surfaces (bottom)

its design. Therefore having a computational framework for tackling shape optimal design w.r.t wind loading for this type of structures is highly demanded.

6.3.1 Wind flow around the roof

For simulation of the wind flow around the roof, the mentioned aspects of chapter 4 are considered. Unlike the automotive cases, the

CAD design, meshing and case setup has been done internally. The NURBS surface is exported to the software ICEM CFD for volume mesh generation. The computational domain is $500m$ long, $400m$ wide and $200m$ high, which has been chosen according to the dimensions of the structure and based on an estimation of the bulk flow behavior (section 4.2.3). It has been tried to design the mesh as efficient as possible using the experience of similar simulations, in order to have accurate enough results with the smallest problem size possible, especially to avoid very long times for the optimization. The volume is meshed by tetrahedral cells and at the wall boundaries four prism layers are built. Close to the surface and in regions with high gradients (e.g. the vortices at the wake) the mesh is refined. For simulations with transient inlet wind boundary condition (equation (4.23)), it is necessary to have a fine mesh in the region between the inlet and the obstacle (figure 6.24) so that the generated vortices are captured by the mesh and transferred to the obstacle with least amount of dissipation. This increases the problem size dramatically.

A boundary layer wind with the mean velocity of $10\frac{m}{s}$ at the reference height $10m$ is blown to the roof. A roughness length of $0.1m$ for the earth surface upstream is considered which is a realistic value for such a case based on [121]. The details about boundary conditions, wall treatment, inlet wind etc. can be found in chapter 4.2.3.

For adjoint optimization a steady RANS simulation with $k-\omega$ SST turbulence model [82], with adjoint frozen turbulence is used. The mesh has around 8 million finite volume cells. For the initial convergence of the fields 2000 iterations were performed (each primal and adjoint). This step takes about 8 hours using 36 processors, almost 5 of which is spent on the primal calculation and the rest on adjoint.

As known from literature, RANS simulations of bluff buddies can be inaccurate. Therefore, a series of LES simulations was performed in order to evaluate the quality of the RANS solution. In addition to the mesh used for the RANS case, two other meshes were generated for the LES calculations. The first mesh is around the roof structure identical to the coarse mesh described in the previous paragraph. The

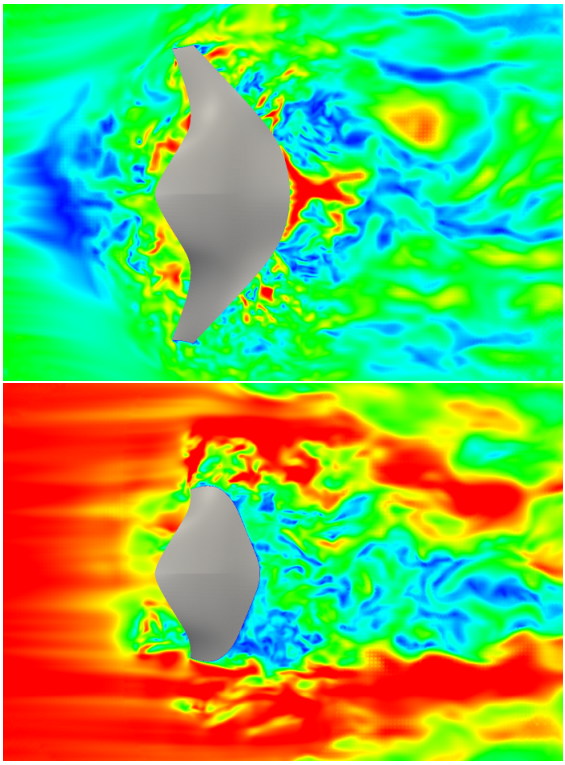
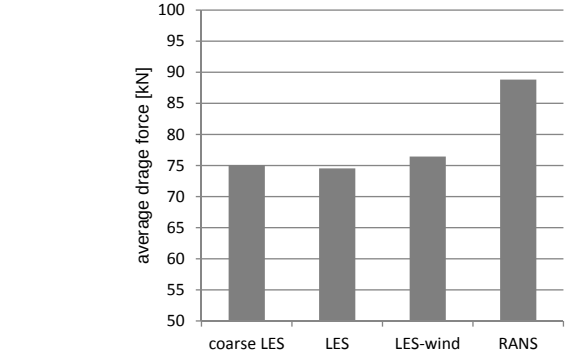


Figure 6.25: Velocity around the roof simulated by LES in two elevations

difference is the refined region between inlet and obstacle which allows for transient wind inlet. The second mesh has about 26 million cells and in addition to finer boundary cells, has a better resolution for capturing the down stream flow structures. LES simulations were done for the physical duration of $200s$. After $100s$ the field averaging starts. The time step is 0.001 which leads to a total number of 2×10^5 time steps. LES simulation with the same mesh as of the RANS takes about 175 hours on 36 processors. Note that there is no adjoint calculation in case of LES. Therefore, the coarsest LES performed (which has a way coarse mesh for an accurate LES simulation) is 35 times more computationally costly compared to steady RANS. The total simulation time for the large case is about 740 hours on 36 processors. Due to some restriction



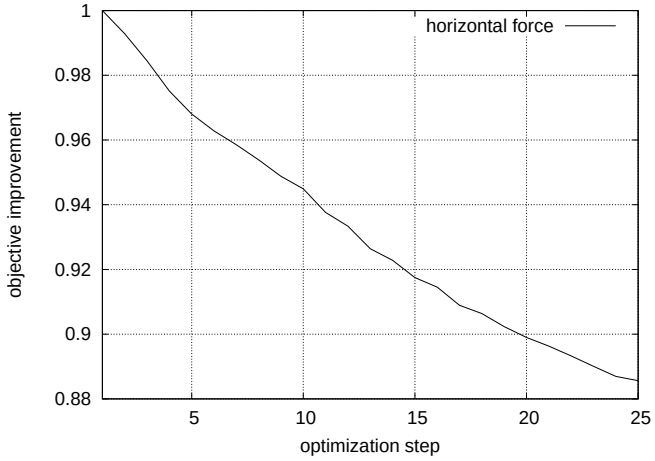


Figure 6.26: Comparison of the average drag for different simulations (top). Objective history for drag reduction on the roof (bottom)

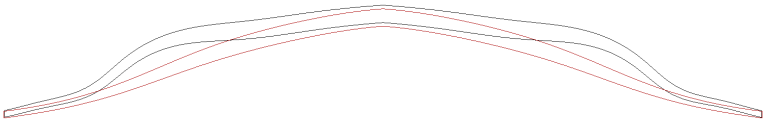


Figure 6.27: Normal to stream roof cross section, initial shape (red) vs. improved for drag (black)

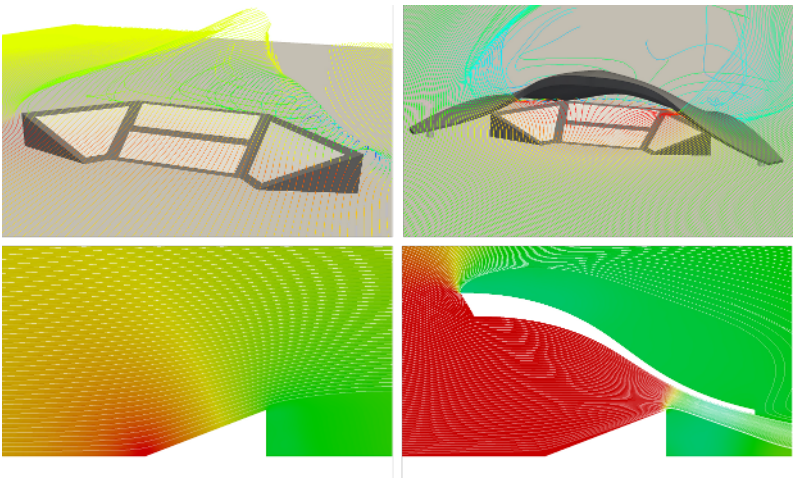


Figure 6.28: The existence of the roof has a big influence on the air flux at the seats location

in domain decomposition algorithm, forced by the implementation of the wind generator, the simulation with transient wind takes much longer and a fair computational time comparison is not possible.

6.3.2 Drag reduction

Shape optimization of this structure is similar to the examples shown in section 6.2.3. Therefore, the focus of this section is not on the optimization itself, but on the physical worthiness of its results. The outcome of a verification study of the wind loads on the roof is presented and

at the end of the section some important and critical conclusions are drawn. The drag force on the roof has been chosen as the objective for sake of simplicity. Note that there might exist objectives which are more relevant to the design of this structure, such as the moment at the supports.

A comparison between the averaged drag forces shown in figure 6.26 indicates that the LES simulations have approximately equal average drag values. This can mean that the mesh is capable of estimating the average loads. For instance, the difference between the small and large model is bellow 1%. Moreover, the mean drag value is not influenced much by the steadiness of the inlet condition, as the average drag of the model with transient inlet is very close to the ones with mean velocity boundary condition. More importantly, the RANS simulation reports a relatively close value to the LES simulations. The difference is bellow 15% which is a good enough accuracy for shape optimization of such a complex flow, specially considering the computational cost. Note that comparison of the averaged values does not prove the quality of the simulation as different errors might be balancing each other out. However, the experience from other test cases (e.g. the pipe flow presented in section 4.2.1) approves the mentioned conclusion. It should be added that in a more realistic design optimization, values such as peak wind load are more critical compared to average values. In such a case a fully transient simulation is required. Moreover, the role of having a realistic wind inlet condition will be more pronounced, since the simultaneous values strongly depend on the inlet condition. For instance, in the presented simulations, the standard deviation of the drag force is 2.5 times larger when a transient wind model is used. Similarly, the ratio of the peak-to-mean ratio of the drag force is about 4 times larger compared to the mean velocity inlet condition.

The roof has around 237000 surface coordinates and can freely move in all regions, except the connection to the columns. Figure 6.26 shows the objective history for 25 optimization steps, with maximum $25cm$ update length per step. As it is seen in figure 6.27, the optimization tries to lift the corners of the roof such that the wind can pass through easier. Moreover, this deformation flattens the roof so it receives less force in mean wind direction.

The volume mesh is deformed during the optimization to adapt to the deformed geometry. If this deformation changes the objective calculation accuracy, the correctness of the optimization would be questioned. In order to study the possibility of this deficiency, a new volume mesh was generated for the resulting improved shape. The drag force of the deformed mesh and the newly generated mesh have a negligible difference. This means that there is no error introduced by the optimizer in the solution of the discretized RANS model.

The next question is if the acceptable level of agreement for the RANS model compared to the reference model (LES) would be extendable to the variation of the objective function, i.e. drag improvement. To find an answer to this question, a new LES simulation was performed on the improved geometry. The LES simulation of the improved shape shows about 5% of change in drag whereas the RANS calculation claims the improvement to be around 11% (figure 6.26). From the view point of error analysis it is not a surprise, since the improvement in the RANS model (11%) is in the same order of magnitude as the difference of RANS and LES models (14%). This mismatch between the "delta values" indicates that the improvement mechanism (or the gradient vector) of the simplified model, RANS, is not necessarily as effective for the reference model, LES. In other words, there is no guarantee that the variation of the objective can be approximated by a reduced model as accurate as the objective value itself.

A remedy to overcome the explained shortage is to calculate the sensitivities of the transient simulation which is not computationally feasible as explained in chapter 4. Further discussions in this direction would be out of the concept of this work and is a current field of research in adjoint optimization.

6.3.3 Passive cooling

Wind engineering is a wide research field and of course is not limited to force evaluation of structures. Consequently, optimization of wind engineering related structures deals with other type of problems as well.

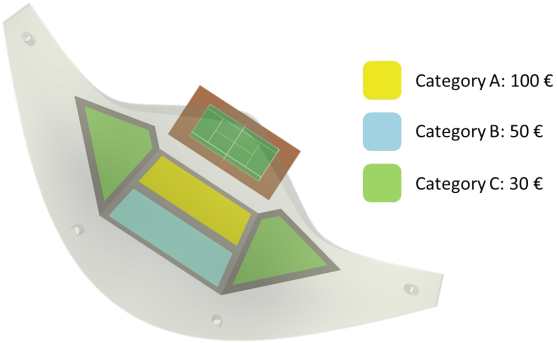


Figure 6.29: Objective weighting for different seat regions

In this section we define a new problem definition on the same structure, i.e. improvement of wind passive cooling. Unlike the previous section, here the main focus is on the geometrical capabilities of *Vertex morphing*.

As visualized in figure 6.28 construction of the roof over the stadium seats does not only protect the fans from sun and rain, but also changes heavily the wind flow around the seats. The roof traps the air under it which causes a high pressure region. As the result, the roof acts an obstacle and large portion of approaching wind elevates at the upstream and passes over the roof. This worsens the cooling and ventilation conditions at the seat positions. The goal of the shape optimization is to modify the roof geometry such that a mild wind can ventilate the seats effectively. Therefore the integral over the parallel to surface wind velocity close to the seat locations is chosen as the objective function. In order to study a weighted multi-objective case, and also for the sake of change, the objective is prioritized (weighted) in three categories based on the ticket price in each region, shown in figure 6.29. Note that there might be other objective alternatives such as flow rate under the roof, but the exact definition of the target function does not matter here, as it does not change the aspects discussed in this section. Moreover, clearly it is not sufficient to study only one wind direction and velocity, and a robust shape optimal design should consider various wind scenarios.

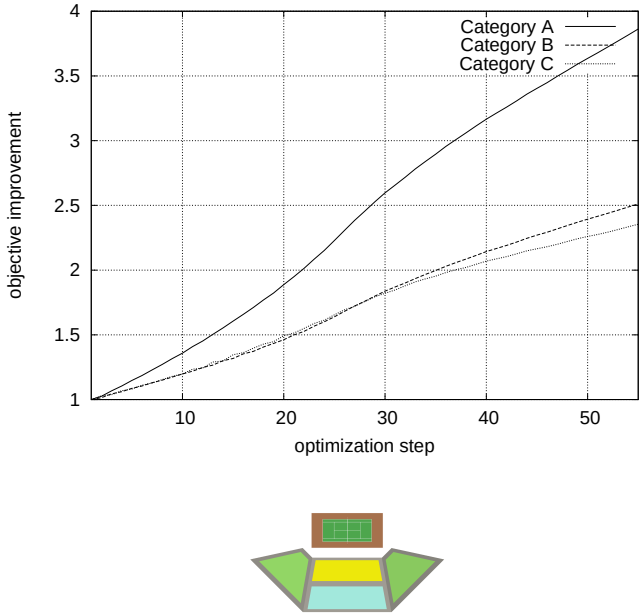


Figure 6.30: The improvement rate for each priority region

The outcome of the optimization seen in figure 6.30 shows that the improvement rate is more or less proportional to the priority factor specified for different seat categories. In general, it cannot be always expected, since the objective functions can be highly correlated in such a problem. Despite the deep curving of the structure the mesh stays almost as good as the initial mesh (figure 6.24) and if needed, even much larger shape deformation would be possible.

The process increases the parallel to surface velocity at the seat positions by two mechanisms (figure 6.32): First opening the narrow gap between the roof and the lower structure by pulling the roof upward, which increases the flux passing under the roof, and second, pushing the roof down towards the seats slightly upstream of the narrow gap so

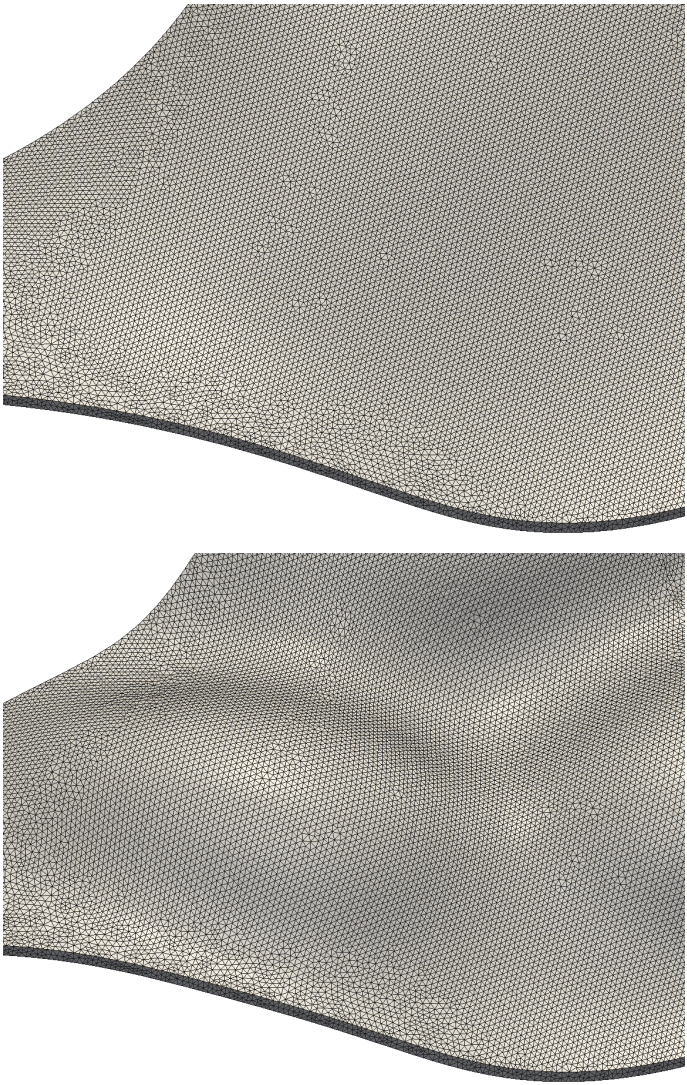


Figure 6.31: The surface mesh of the roof before and after optimziation

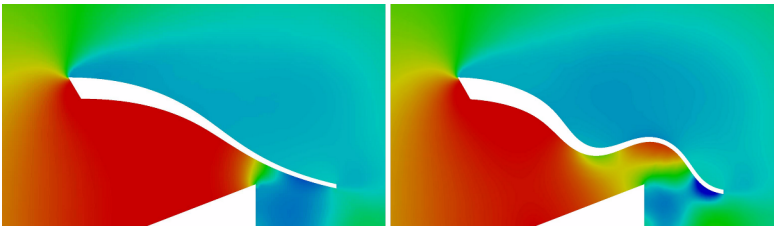


Figure 6.32: Pressure field in a lateral section for initial and improved roof

that the air passage is smaller, which leads to a higher velocity. The mixture of these two mechanisms creates an interesting challenge that how would the roof deform in order to make use of both improvement mechanisms. In other words, the improved shape shall bring a smooth transition between the increased height at the edge of the lower structure and the decreased height right before that. Node-based shape optimization provides a good basis for generation of such forms.

The filter size is the parameter which decides how smooth or sharp the transition should be. Figure 6.34 clearly shows that the evolution of the shape when a small filter is used is more local, and mainly close to the narrow gap. In contrast, the large filter results in a "smoother" shape with larger curvature radius, which means that the deformation is transformed to further parts of the roof as well. Similar to the example of section 6.2.2, a smaller filter size would make a faster shape transition in the direction of the gradient and hence a faster decrease of the objective value is to be expected. This can be observed in the graph of figure 6.33.

In the section plots of figure 6.35, it can be seen that when the filter size is smaller or in the same range as the thickness of the roof, the roof thickness is also modified. Considering the fact that the sensitivity of the upper surface is much smaller than the lower one, changing the thickness is a logical path towards the optimum. It should be mentioned that the thickness reduction takes place only in ticker regions of the roof (figure 6.35). More importantly, thinning almost stops at a certain stage, and both upper and lower faces deform synchronously.

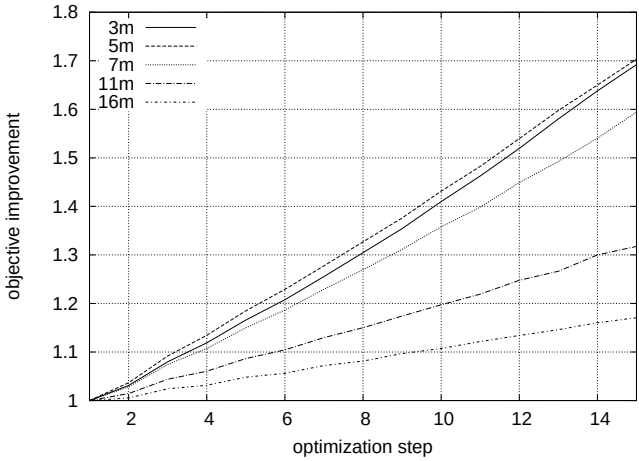


Figure 6.33: Influence of the filter size on improvement rate

Therefore, the roof does not get thinner than a limit, unless many optimization iterations are performed, which is not the case, in such a problem, as the roof looses its form completely.

This observation shows an other application of the feature preservation, discussed in section 3.1.6 as a positive and important property of this method. Here, the features are not only a design element, but also a means to keep the resulting shape of the optimization physically meaningful. This property does not exist, when the smoothing operator is defined based on the surface topology, e.g. the implicit curvature based smoothing of equation (3.21). Note that this property is not related to the formulation of the filtering problem (implicit or explicit), but to the way the filtering operator is constructed. Example of figure 6.36 compares the update pattern on a section of the roof for two different cases, in both of which an explicit smoothing is used. In the first case (top), the node distances are evaluated by the help of mesh topology, whereas the second case (bottom) uses a global distance

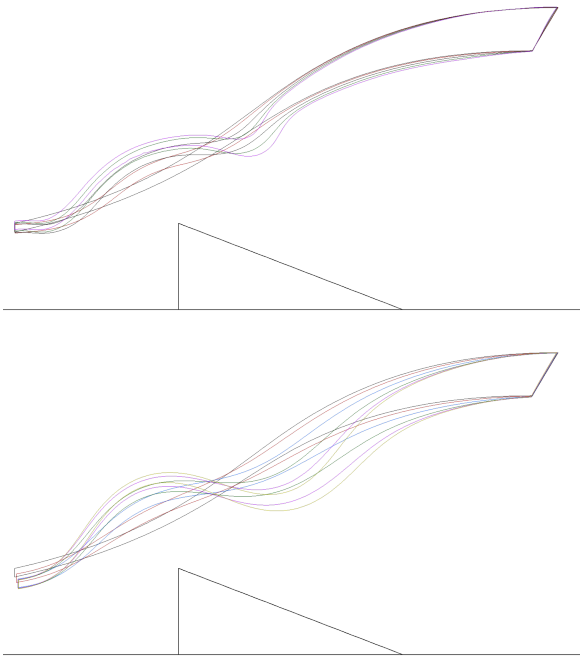


Figure 6.34: Shape development modes for filter sizes 5 (top) and 16 (bottom)

evaluation (section 5.2), as all other presented examples in this chapter. Nodes of the lower surface have a large topological distance to the nodes of the upper surface. Therefore, in the first case, the large sensitivities of the lower surface result in deformation only in nodes of the lower surface. This means that the and upper faces stays almost fixed, and the lower surface moves upwards. The optimization continues iterating even when the lower surface has deeply penetrated the upper one. Although this has no realistic meaning, but for the CFD calculation, there is no problem to solve the Navier-Stokes equations on overlapping spaces. This is not the case for the global distance evaluation. In the second case, the wall thickness is seen as a design feature. In other words, the surface points which are close to each other sense each other, no matter to which side of the wall they belong. As the result and

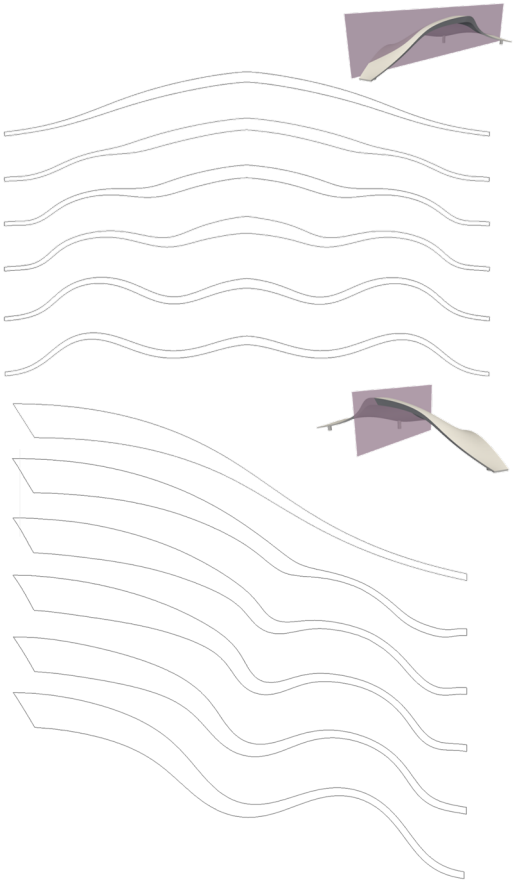


Figure 6.35: Normal to stream (top) and lateral (bottom) sections of the initial roof geometry (first row) and improved ones with filter sizes from top to bottom 3, 5, 7, 11 and 16 meters

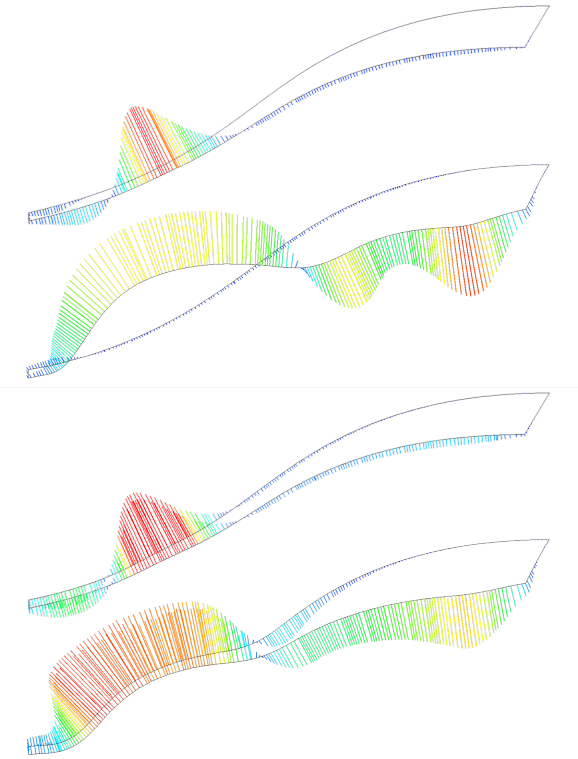


Figure 6.36: Performance of a surface-based smoothing (top) vs. global smoothing (bottom)

from the beginning, the two surface of the roof move together. Still, the thick regions of the wall can freely change thickness as it is the case in the bottom plot of figure 6.36 where the upstream part of the roof starts moving down to further tighten the air passage.

Chapter 7

Conclusions and outlook

This contribution tried to observe the numerical shape optimization as an engineering tool that can be used within the design process, rather than an effort to replace the designer by an intelligent algorithm which finds the "best possible" design. Based on this interpretation, the shape optimization tool shall effectively transform the result of the numerical analysis (physical and sensitivity) into a meaningful geometrical improvement. To this end, a novel node-based parametrization, *Vertex Morphing* was developed which takes into account the geometrical requirements of a general industrial design. The mathematical interpretation of the technique was analyzed through a consistent notation, so that the reader can establish the link to the theory of optimization and regularization.

A design control field as the essence of *Vertex Morphing* method was introduced and the optimization problem was formulated in the control space, instead of the geometry. This notation makes it possible to see the CAD objects or subdivision surfaces as a coarse discretization of the control field. It was proven as well that the optimal solution is independent of the parametrization when the standard version of

the method is used in which the control and geometry fields share the same discretization. Furthermore, it was stated that one can guide the optimization path towards a desired local minimum by selecting the right filter radius. This property is very important since most of the engineering applications of shape optimization deal with highly non-convex response surfaces, with many local minima. *Vertex Morphing* leaves the decision of "which local minimum fits the design requirements (manufacturing, aesthetics, etc.) the best?", to the user, by providing a comprehensible physical variable as the input parameter: the filter radius. In other words, by using a spatial low-pass filter, the designer can decide about the minimum geometrical wavelengths existing in the shape variation pattern. This matches well the goal of this project, mentioned at the beginning of this section.

Advantages and challenges of node-based compared to the other parametrization techniques were discussed. The need for regularization in the ill-posed node-based shape optimization problem was explained. Different remedies were reviewed and the filtering approach of the proposed parametrization was compared to other known techniques in shape and topology optimization. It was concluded that for the type of optimization aimed here, more decisive than the smoothing formulation is the filtering intensity, which is a common input variable in all the reviewed methods. Moreover, the problem of mesh distortion and its main causes were explored. Simultaneous treatment of the shape and mesh regularity in *Vertex Morphing* was studied and demonstrated by a test case.

State equations of the target problem (CFD) were presented and some comments on turbulence modeling with focus on wind engineering were followed. Some modeling remarks about wind simulations including generation of the turbulent wind inflow were stated. Evaluation of the surface sensitivities by a continuous adjoint formulation of Naver-Stokes equation was explained. Furthermore, the coupled fluid-structure interaction problem and its solution techniques were introduced and the challenges in optimization of this type of problems were investigated.

The developed method was put into a numerical shape optimization

workflow and successfully tested through several examples. First, simple conceptual test cases were examined for the sake of verification and comparison with other works. The second group of applications were from automotive industry, including internal flows (air channel and intake manifold) as well as external aerodynamics (car body and side mirrors). Despite the high level of geometrical complexity, low mesh quality and large number of design variables, the method showed significant success in improving the design. At the end, the shape of a large structure exposed to wind flow was optimized in two different scenarios. The first scenario, which was a drag minimization, analyzed the level of error to be expected by steady assumption in the adjoint equation. It was seen that the error in estimation of the improvement (sometimes called "delta value") can be remarkably larger than the error in evaluation of the absolute objective value, caused by the steady solution of the primal equations. This is an inadequacy in resolving the correct physics of the phenomenon and is not related to the proposed method. The objective of the second optimization scenario was to increase the wind passive cooling. The special set up of the problem provided the basis to analyze some geometrical aspects of the method.

To sum up, it was observed that the *Vertex Morphing* method, together with the adjoint sensitivity analysis form a strong combination which offers a great performance for shape optimization of very large problems. The simultaneous treatment of normal and tangential directions shows a significant advantage over previously published decoupled techniques. The choice of algorithms for neighbor-search, optimization and integration are influential features in efficiency of the implementation. Simplicity of the notation and generality of the method makes it possible to be applied on various fields of engineering as a "solver-independent" shape control tool.

A worthwhile continuation of the presented work would be to apply the morphing module to other computational programs, particularly in solid mechanics, and as a further step to fluid-structure interaction. Moreover, in order to increase the usability of the developed method in industries, it is necessary to include typical engineering constraints such as packaging. To the author's opinion, upgrading to a higher order approximation of the surface, more accurate integration or applying

more advanced optimization algorithms would remove the simplicity and the explicit nature of the method, and thus the observed level of robustness and efficiency could not be expected.

Bibliography

[1] M. Abu-Zurayk and J. Brezillon. Development of the adjoint approach for aeroelastic wing optimization. In A. Dillmann, G. Heller, H.-P. Kreplin, W. Nitsche, and I. Peltzer, editors, *New Results in Numerical and Experimental Fluid Mechanics VIII*, volume 121 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 59–66. Springer Berlin Heidelberg, 2013.

[2] J. Arora. *Introduction to optimum design*. Academic Press, 2004.

[3] J. S. Arora and Q. Wang. Review of formulations for structural and mechanical system optimization. *Structural and Multidisciplinary Optimization*, 30(4):251–272, 2005.

[4] D. Ashlock. *Evolutionary computation for modeling and optimization*, volume 103. Springer, 2006.

[5] H. Azegami and K. Takeuchi. A smoothing method for shape optimization: traction method using the Robin condition. *International Journal of Computational Methods*, 3(1):21–34, 2006.

[6] R. Balasubramanian and J. C. Newman. Discrete direct and adjoint sensitivity analysis for arbitrary Mach number flows. *International Journal for Numerical Methods in Engineering*, 66 (2):297–318, 2006.

[7] Y. Bao, X. Guo, and H. Qin. Physically based morphing of point-sampled surfaces. *Computer Animation and Virtual Worlds*, 16 (3-4):509–518, 2005.

[8] B. Barthelemy and R. T. Haftka. Accuracy analysis of the semi–analytical method for shape sensitivity calculation. *Mechanics of Structures and Machines*, 18(3):407–432, 1990.

[9] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wüchner, and K.-U. Bletzinger. 3D simulation of wind turbine rotors at full scale. Part II: Fluid–structure interaction modeling with composite blades. *International Journal for Numerical Methods in Fluids*, 65(1-3): 236–253, 2011.

[10] M. P. Bendsoe and O. Sigmund. *Topology Optimization. Theory, Methods and Applications*. Springer, 2003.

[11] K.-U. Bletzinger. *Formoptimierung von Flaechentragwerken*. PhD thesis, Institut für Baustatik, Universität Stutgart, 1990.

[12] K.-U. Bletzinger. A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape. *Structural and Multidisciplinary Optimization*, 49(6):873–895, 2014.

[13] K.-U. Bletzinger and E. Ramm. A general finite element approach to the form finding of tensile structures by the updated reference strategy. *International Journal of Space Structures*, 14(2):131–145, 1999.

[14] B. Blocken, T. Stathopoulos, and J. Carmeliet. CFD simulation of the atmospheric boundary layer: wall function problems. *Atmospheric Environment*, 41(2):238–252, 2007.

[15] B. Bourdin. Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, 50(9):2143–2158, 2001.

[16] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6): 350–355, 1978.

[17] D. Chenais. On the existence of a solution in a domain identification problem. *Journal of Mathematical Analysis and Applications*, 52:189–219, 1975.

[18] G. Cheng and N. Olhoff. *New method of error analysis and detection in semi–analytical sensitivity analysis.* Springer, 1993.

[19] K. K. Choi and N.-H. Kim. *Structural sensitivity analysis and optimization 1: Linear systems*, volume 1. Springer, 2006.

[20] A. G. Davenport. *The relationship of wind structure to wind loading.* National Physical Laboratory, 1966.

[21] J. Degroote. *Development of algorithms for the partitioned simulation of strongly coupled fluid-structure interaction problems.* PhD thesis, Ghent University, 2010.

[22] J. Degroote, R. Haelterman, S. Annerel, P. Bruggeman, and J. Vierendeels. Performance of partitioned procedures in fluid–structure interaction. *Computers & Structures*, 88(7):446–457, 2010.

[23] J. Degroote, M. Hojjat, E. Stavropoulou, R. Wüchner, and K.-U. Bletzinger. Partitioned solution of an unsteady adjoint for strongly coupled fluid-structure interactions and application to parameter identification of a one-dimensional problem. *Structural and Multidisciplinary Optimization*, 47(1):77–94, 2013.

[24] M. Di Paola. Digital simulation of wind field velocity. *Journal of Wind Engineering and Industrial Aerodynamics*, 74:91–109, 1998.

[25] M. Donatelli. A multigrid for image deblurring with Tikhonov regularization. *Numerical Linear Algebra with Applications*, 12 (8):715–729, 2005.

[26] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran. *Arbitrary Lagrangian–Eulerian methods.* Encyclopedia of Computational Mechanics. Wiley Online Library, 2004.

[27] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* John Wiley & Sons, 2012.

[28] L. F. R. Espath, R. V. Linn, and A. M. Awruch. Shape optimization of shell structures based on NURBS description using automatic differentiation. *International Journal for Numerical Methods in Engineering*, 88(7):613–636, 2011.

[29] C. Farhat, M. Lesoinne, and N. Maman. Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids*, 21(10): 807–835, 1995.

[30] A. Fazzolari, N. R. Gauger, and J. Brezillon. Efficient aerodynamic shape optimization in MDO context. *Journal of Computational and Applied Mathematics*, 203(2):548–560, 2007.

[31] C. A. Felippa, K. C. Park, and C. Farhat. Partitioned analysis of coupled mechanical systems. *Computer Methods in Applied Mechanics and Engineering*, 190(24):3247–3270, 2001.

[32] M. Firl. *Optimal shape design of shell structures.* PhD thesis, Technische Universität München, 2010.

[33] M. Firl, R. Wüchner, and K.-U. Bletzinger. Regularization of shape optimization problems using FE-based parametrization. *Structural and Multidisciplinary Optimization*, 47(4):507–521, 2013.

[34] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics: Modeling and Simulation of the Circulatory System.* MS & A. Springer, 2009.

[35] J. Franke, C. Hirsch, A. G. Jensen, H. W. Krüs, M. Schatzmann, P. S. Westbury, S. D. Miles, J. A. Wisse, and N. G. Wright. Recommendations on the use of cfd in wind engineering. In *Cost Action C*, volume 14, page C1, 2004.

[36] D. Fudge, D. W. Zingg, and R. Haimes. *A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization.* University of Toronto, 2004.

[37] K. Fukunaga. *Introduction to statistical pattern recognition.* Academic Press, 1990.

[38] R. P. Gao, Z. Y. Tay, C. M. Wang, and C. G. Koh. Hydroelastic response of very large floating structure with a flexible line connection. *Ocean Engineering*, 38(17):1957–1966, 2011.

[39] M. W. Gee, U. Küttler, and W. A. Wall. Truly monolithic algebraic multigrid for fluid–structure interaction. *International Journal for Numerical Methods in Engineering*, 85(8):987–1016, 2011.

[40] P. Geuzaine, G. Brown, C. Harris, and C. Farhat. Aeroelastic dynamic analysis of a full f-16 configuration for various flight conditions. *AIAA journal*, 41(3):363–371, 2003.

[41] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3-4): 393–415, 2000.

[42] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3(2):95–99, 1988.

[43] A. Griewank and A. Walther. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 26(1):19–45, 2000.

[44] J. Hadamard. *Four Lectures on Mathematics*. Number 5. Columbia University Press, 1915.

[45] R. T. Haftka and R. V. Grandhi. Structural shape optimization–A survey. *Computer Methods in Applied Mechanics and Engineering*, 57(1):91–106, 1986.

[46] G. Hansen, A. Zardecki, D. Greening, and R. Bos. A finite element method for three-dimensional unstructured grid smoothing. *Journal of Computational Physics*, 202(1):281–297, 2005.

[47] S. B. Hazra, V. Schulz, J. Brezillon, and N. R. Gauger. Aerodynamic shape optimization using simultaneous pseudo-timestepping. *Journal of Computational Physics*, 204(1):46–64, 2005.

[48] M. Heil. An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193(1):1–23, 2004.

[49] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.

[50] M. Hojjat, E. Stavropoulou, T. Gallinger, U. Israel, R. Wüchner, and K.-U. Bletzinger. Fluid–structure interaction in the context of shape optimization and computational wind engineering. In H.-J. Bungartz, M. Mehl, and M. Schäfer, editors, *Fluid–Structure Interaction II: Modelling, Simulation, Optimization*, NATO ASI Series, pages 351–381. Springer, Berlin Heidelberg, 2010.

[51] M. Hojjat, E. Stavropoulou, and K.-U. Bletzinger. The Vertex Morphing method for node-based shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268:494–513, 2014.

[52] J. Hron and S. Turek. A monolithic FEM/Multigrid solver for an ALE formulation of fluid-structure interaction with applications in biomechanics. In H.-J. Bungartz and M. Schäfer, editors, *Fluid-Structure Interaction*, volume 53 of *Lecture Notes in Computational Science and Engineering*, pages 146–170. Springer Berlin Heidelberg, 2006.

[53] J. Huan and V. Modi. Optimum design of minimum drag bodies in incompressible laminar flow using a control theory approach. *Inverse Problems in Engineering*, 1(1):1–25, 1994.

[54] A. Iannuzzi and P. Spinelli. Artificial wind generation and structural response. *Journal of Structural Engineering*, 113(12):2382–2398, 1987.

[55] S. Jakobsson and O. Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, 36(6):1119–1136, 2007.

[56] A. Jameson. Aerodynamic shape optimization using the adjoint method. Lectures at the Von Karman Institute, Brussels, 2003.

[57] A. Jameson and J. C. Vassberg. Studies of alternative numerical optimization methods applied to the brachistochrone problem. *Computational Fluid Dynamics Journal*, 9(3):281–296, 2000.

[58] A. Jameson, L. Martinelli, and N. A. Pierce. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and Computational Fluid Dynamics*, 10(1):213–237, 1998.

[59] H. Jasak and Z. Tukovic. Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA*, 30 (2):1–20, 2006.

[60] A. Jaworski and J.-D. Müller. Toward modular multigrid design optimisation. In C. H. Bischof, H. M. Bücker, P. Hovland, U. Naumann, and J. Utke, editors, *Advances in Automatic Differentiation*, volume 64 of *Lecture Notes in Computational Science and Engineering*, pages 281–291. Springer Berlin Heidelberg, 2008.

[61] S. G. Johnson and M. Frigo. A modified split-radix fft with fewer arithmetic operations. *Signal Processing, IEEE Transactions on*, 55(1):111–119, 2007.

[62] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. In *ACM Transactions on Graphics (TOG)*, volume 26, page 71. ACM, 2007.

[63] J. Kaipio and E. Somersalo. *Statistical and computational inverse problems*, volume 160. Springer, 2005.

[64] J. M. Kiendl. *Isogeometric Analysis and Shape Optimal Design of Shell Structures*. PhD thesis, Technische Universität München, Munich, Germany, 2011.

[65] L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13(8):743–761, 1996.

[66] V. Komkov, K. K. Choi, and E. J. Haug. *Design sensitivity analysis of structural systems*, volume 177. Academic Press, 1986.

[67] U. Küttler and W. A. Wall. Fixed–point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72, 2008.

[68] U. Küttler, M. Gee, C. Förster, A. Comerford, and W.A. Wall. Coupling strategies for biomedical fluid–structure interaction problems. *International Journal for Numerical Methods in Biomedical Engineering*, 26(3-4):305–321, 2010.

[69] K. C. Lam, C. Wen, and L. M. Lui. Conformal-based surface morphing and multi-scale representation. *Axioms*, 3(2):222–243, 2014.

[70] C. Le, T. Bruns, and D. Tortorelli. A gradient-based, parameter-free approach to shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 200(9):985–996, 2011.

[71] J. Linhard and K.-U. Bletzinger. "Tracing" the equilibrium - Recent advances in numerical form finding. *International Journal of Space Structures*, 25(2):107–116, 2010.

[72] N. Litke, A. Levin, and P. Schröder. Trimming for subdivision surfaces. *Computer Aided Geometric Design*, 18(5):463–481, 2001.

[73] E. Lund and N. Olhoff. Shape design sensitivity analysis of eigenvalues using "exact" numerical differentiation of finite element matrices. *Structural Optimization*, 8(1):52–59, 1994.

[74] E. Lund, H. Møller, and L. A. Jakobsen. Shape design optimization of stationary fluid-structure interaction problems with large displacements and turbulence. *Structural and Multidisciplinary Optimization*, 25(5-6):383–392, 2003.

[75] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 181–188. ACM, 1996.

[76] S. Maneewongvatana and D. Mount. An empirical study of a new approach to nearest neighbor searching. *Algorithm Engineering and Experimentation*, pages 172–187, 2001.

[77] J. Mann. The spatial structure of neutral atmospheric surface-layer turbulence. *Journal of Fluid Mechanics*, 273:141–168, 1994.

[78] J. Martins, J. Alonso, and J. Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523–530, 2004.

[79] J. Martins, J. Alonso, and J. Reuther. A coupled-adjoint sensitivity analysis method for high–fidelity aero–structural design. *Optimization and Engineering*, 6(1):33–62, 2005.

[80] K. Maute, M. Nikbay, and C. Farhat. Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. *AIAA journal*, 39(11):2051–2061, 2001.

[81] K. Maute, M. Nikbay, and C. Farhat. Sensitivity analysis and design optimization of three-dimensional non-linear aeroelastic systems by the adjoint method. *International Journal for Numerical Methods in Engineering*, 56(6):911–933, 2003.

[82] F. R. Menter. Zonal two equation k-turbulence models for aerodynamic flows. *AIAA Paper*, 1993.

[83] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005.

[84] B. Mohammadi. Shape optimization for 3D turbulent flows using automatic differentiation. *International Journal of Computational Fluid Dynamics*, 11(1-2):27–50, 1998.

[85] B. Mohammadi and O. Pironneau. Shape optimization in fluid mechanics. *Annual Review of Fluid Mechanics*, 36:255–279, 2004.

[86] B. Mohammadi and O. Pironneau. *Applied shape optimization for fluids*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2009.

[87] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *ACM SIGGRAPH 2005 Courses*, page 78. ACM, 2005.

[88] S. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA Paper*, 667, 2000.

[89] A. Nemili, E. Özkaya, and N. Gauger. Discrete adjoint based sensitivity analysis for optimal active flow control of high-lift configurations. *PAMM*, 13(1):347–348, 2013.

[90] E. J. Nielsen, J. Lu, M. A. Park, and D. L. Darmofal. An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids. *Computers & Fluids*, 33(9):1131–1155, 2004.

[91] S. Osher and J. A. Sethian. Fronts propagating with curvature–dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[92] C. Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International Journal for Numerical Methods in Fluids*, 58(8): 861–877, 2008.

[93] C. Othmer, E. de Villiers, and H. G. Weller. Implementation of a continuous adjoint for topology optimization of ducted flows. In *18th AIAA Computational Fluid Dynamics Conference*, 2007.

[94] C. Othmer, Papoutsis-Kiachagias M. E., and K. Haliskos. CFD optimization via sensitivity-based shape morphing. In *NAFEMS NORDIC Conference 2012*, 2012.

[95] E. Özkaya and N. R. Gauger. Single-step one-shot aerodynamic shape optimization. *Optimal Control of Coupled Systems of Partial Differential Equations*, pages 191–204, 2009.

[96] D. I. Papadimitriou and K. C. Giannakoglou. A continuous adjoint method with objective function derivatives based on boundary integrals, for inviscid and viscous flows. *Computers & Fluids*, 36(2):325–341, 2007.

[97] D. I. Papadimitriou and K. C. Giannakoglou. Computation of the Hessian matrix in aerodynamic inverse design using continuous adjoint formulations. *Computers & Fluids*, 37(8):1029–1039, 2008.

[98] D. I. Papadimitriou and K. C. Giannakoglou. Third–order sensitivity analysis for robust aerodynamic design using continuous adjoint. *International Journal for Numerical Methods in Fluids*, 2012.

[99] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, pages 1065–1076, 1962.

[100] C. S. Peskin. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics*, 10(2):252–271, 1972.

[101] S. Piperno and C. Farhat. Partitioned procedures for the transient solution of coupled aeroelastic problems–Part II: Energy transfer analysis and three-dimensional applications. *Computer Methods in Applied Mechanics and Engineering*, 190(24):3147–3170, 2001.

[102] S. Piperno, C. Farhat, and B. Larrouturou. Partitioned procedures for the transient solution of coupled aroelastic problems–Part I: Model problem, theory and two-dimensional application. *Computer Methods in Applied Mechanics and Engineering*, 124 (1):79–112, 1995.

[103] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(01):97–110, 1974.

[104] O. Pironneau. Optimal shape design for elliptic systems. *System Modeling and Optimization*, pages 42–66, 1982.

[105] S. B. Pope. *Turbulent flows*. Cambridge University Press, 2000.

[106] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. *Journal of Aircraft*, 36(1):51–60, 1999.

[107] P. J. Richards and R. P. Hoxey. Appropriate boundary conditions for computational wind engineering models using the k-epsiolon turbulence model. *Journal of Wind Engineering and Industrial Aerodynamics*, 46:145–153, 1993.

[108] P. J. Richards, R. P. Hoxey, and L. J. Short. Wind pressures on a 6m cube. *Journal of Wind Engineering and Industrial Aerodynamics*, 89(14):1553–1564, 2001.

[109] P. J. Richards, R. P. Hoxey, B. D. Connell, and D. P. Lander. Wind-tunnel modelling of the silsoe cube. *Journal of Wind Engineering and Industrial Aerodynamics*, 95(9):1384–1399, 2007.

[110] T. T. Robinson, C. G. Armstrong, H. S. Chua, C. Othmer, and T. Grahs. Optimizing parameterized CAD geometries using sensitivities based on adjoint functions. *Computer-Aided Design and Applications*, 9(3):253–268, 2012.

[111] R. Rossi, M. Lazzari, and R. Vitaliani. Wind field simulation for structural engineering purposes. *International Journal for Numerical Methods in Engineering*, 61(5):738–763, 2004.

[112] J. A. Samareh. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA journal*, 39(5):877–884, 2001.

[113] J. A. Samareh. Aerodynamic shape optimization based on free-form deformation. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004.

[114] M. Scherer, R. Denzer, and P. Steinmann. A fictitious energy approach for shape optimization. *International Journal for Numerical Methods in Engineering*, 82(3):269–302, 2010.

[115] S. Schmidt. *Efficient large scale aerodynamic design based on shape calculus*. PhD thesis, Universität Trier, 2010.

[116] S. Schmidt, C. Ilic, N. Gauger, and V. Schulz. Shape gradients and their smoothness for practical aerodynamic design optimization. *Optimization and Engineering, submitted (Preprint No. SPP1253-10-03)*, 2008.

[117] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics*, volume 20, pages 151–160. ACM, 1986.

[118] O. Sigmund. *Design of material structures using topology optimization*. PhD thesis, Technical University of Denmark, 1994.

[119] O. Sigmund. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33(4-5):401–424, 2007.

[120] O. Sigmund and K. Maute. Sensitivity filtering from a continuum mechanics perspective. *Structural and Multidisciplinary Optimization*, 46(4):471–475, 2012.

[121] E. Simiu and R. H. Scanlan. *Wind effects on structures*. Wiley, 1996.

[122] J. Sobieszczanski-Sobieski. Sensitivity of complex, internally coupled systems. *AIAA Journal*, 28(1):153–160, 1990.

[123] O. Soto and R. Löhner. CFD shape optimization using an incomplete–gradient adjoint formulation. *International Journal for Numerical Methods in Engineering*, 51(6):735–753, 2001.

[124] O. Soto and R. Löhner. On the computation of flow sensitivities from boundary integrals. *AIAA Paper*, 112, 2004.

[125] O. Soto, R. Löhner, and C. Yang. A stabilized pseudo-shell approach for surface parametrization in CFD design problems. *Communications in Numerical Methods in Engineering*, 18(4): 251–258, 2002.

[126] E. Stavropoulou. *Sensitivity analysis and regularization for shape optimization of coupled problems*. PhD thesis, Technische Universität München, Munich, Germany, 2015.

[127] E. Stavropoulou, Hojjat M., R. Wüchner, and K.-U. Bletzinger. A global mesh regularization approach for two and three dimensional grids. In *Proceedings of the V European Conference on Computational Fluid Dynamics ECCOMAS CFD*, Lisbon, Portugal, June 2010.

[128] E. Stavropoulou, M. Hojjat, and K.-U. Bletzinger. In-plane mesh regularization for node-based shape optimization problems. *Computer Methods in Applied Mechanics and Engineering*, 275: 39 – 54, 2014.

[129] E. Stavropoulou, M. Hojjat, and K.-U. Bletzinger. In-plane mesh regularization for node-based shape optimization problems. *Computer Methods in Applied Mechanics and Engineering*, 275: 39–54, 2014.

[130] A. Stück. *Adjoint Navier–Stokes methods for hydrodynamic shape optimisation*. PhD thesis, Technische Universität Hamburg-Harburg, Hamburg, Germany, 2011.

[131] A. Stück and T. Rung. Adjoint RANS with filtered shape derivatives for hydrodynamic optimisation. *Computers & Fluids*, 47 (1):22–32, 2011.

[132] A. Takezawa, S. Nishiwaki, and M. Kitamura. Shape and topology optimization based on the phase field method and sensitivity analysis. *Journal of Computational Physics*, 229(7):2697–2718, 2010.

[133] T. E. Tezduyar, S. Sathe, R. Keedy, and K. Stein. Space-time finite element techniques for computation of fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 195(17):2002–2027, 2006.

[134] D. A. Tortorelli and P. Michaleris. Design sensitivity analysis: Overview and review. *Inverse Problems in Engineering*, 1(1): 71–105, 1994.

[135] F. Van Keulen, R. T. Haftka, and N. H. Kim. Review of options for structural design sensitivity analysis. Part 1: Linear systems. *Computer Methods in Applied Mechanics and Engineering*, 194 (30):3213–3243, 2005.

[136] A. H. van Zuijlen, S. Bosscher, and H. Bijl. Two level algorithms for partitioned fluid–structure interaction computations. *Computer Methods in Applied Mechanics and Engineering*, 196(8): 1458–1470, 2007.

[137] G. N. Vanderplaats. *Numerical optimization techniques for engineering design: with applications*, volume 1. McGraw-Hill New York, 1984.

[138] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12:620, 1998.

[139] N. G. Wright and D. M. Hargreaves. Unsteady cfd simulations for natural ventilation. *International Journal of Ventilation*, 5 (1):13–20, 2006.

[140] R. Wüchner and K.-U. Bletzinger. Stress-adapted numerical form finding of pre-stressed surfaces by the updated reference strategy. *International Journal For Numerical Methods in Engineering*, 64 (2):143–166, 2005.

[141] R. Wüchner, A. Kupzok, and K.-U. Bletzinger. A framework for stabilized partitioned analysis of thin membrane–wind interaction. *International Journal for Numerical Methods in Fluids*, 54(6-8): 945–963, 2007.

[142] P. Zhang, J. Peng, and N. Riedel. Finite sample error bound for Parzen windows. In *AAAI*, volume 5, pages 925–930, 2005.

[143] M. Zhou, Y. K. Shyy, and H. L. Thomas. Checkerboard and minimum member size control in topology optimization. *Structural and Multidisciplinary Optimization*, 21(2):152–158, 2001.

[144] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 189–192. ACM, 1996.

[145] A. S. Zymaris, D. I. Papadimitriou, K. C. Giannakoglou, and C. Othmer. Continuous adjoint approach to the Spalart–Allmaras turbulence model for incompressible flows. *Computers & Fluids*, 38(8):1528–1538, 2009.