David W. Zingg and Thomas A. Reist
Institute for Aerospace Studies
University of Toronto

Aerodynamic Shape Optimization: Theory to Practice

July 21, 2020

Springer Nature

Preface

This book is intended to provide both students and practitioners with a deep understanding of fundamental aspects of aerodynamic shape optimization based on computational fluid dynamics, whether they are developing new algorithms or applying existing algorithms. It can be used as a textbook for a course at the graduate level but will hopefully be equally useful for individuals seeking information on the development and selection of algorithms or guidance on the formulation of aerodynamic shape optimization problems. A basic level of understanding of both aerodynamics and computational fluid dynamics is assumed.

With these objectives in mind, the book is structured in the following manner. After a detailed description of problem formulations, the essential components of an aerodynamic shape optimization methodology are introduced, including geometry parameterization and control, mesh deformation, flow evaluation, gradient computation, and optimization. Various algorithms are discussed for each component, exposing the reader to the strengths and weaknesses of the different options. A particular methodology is then described in more detail. This is not intended to suggest that the particular algorithms described more comprehensively are best for all problems. Rather, the advantage of this approach is twofold. First, it provides the reader with enough information to code up the methodology, enabling a thorough understanding of the issues involved. Second, it facilitates an understanding of important algorithmic aspects which, although the details will be specific to a particular approach, can best be described in the context of a specific algorithm.

This is followed by a description of several examples of aerodynamic shape optimization problems that have been solved using the methodology described. These examples serve several purposes. For example, they enable presentation of various considerations in effective problem formulation for aerodynamic design. Moreover, if the reader chooses to code the algorithms presented, these examples can be used to confirm that the algorithms have been coded correctly. Alternatively, if the reader has developed or has access to an alternative methodology, the examples provide useful benchmarks for characterizing the performance of various approaches. The focus of the examples presented is on aircraft and wing design, but the principles illustrated apply equally well to other areas of aerodynamic design.

The reader's needs will determine the manner in which this book is used. It has been deliberately structured to be suitable as a textbook for a graduate course in that the material has a logical sequence. A reader who seeks guidance related to a particular topic or to the application of aerodynamic shape optimization, however, may prefer to skip to a particular chapter that provides the relevant information.

Aerodynamic shape optimization can be a powerful tool for aerodynamic design in a wide range of applications. It is hoped that this book will provide the reader with the foundation for an intelligent and creative approach to the development and application of algorithms for aerodynamic shape optimization.

Toronto,
April 2020

David W. Zingg
Thomas A. Reist

Contents

Part I
Fundamentals of Aerodynamic Shape Optimization

Chapter 1
Introduction

1.1 Background and Overview

The field of aerodynamic shape optimization combines computational fluid dynamics (CFD) with numerical optimization. CFD algorithms enable the numerical solution of the equations governing the flow over aerodynamic geometries, such as wings and aircraft. Initially CFD was seen as providing a *numerical wind tunnel* with the goal of providing the capability of predicting the flow around a geometry and the resulting forces and moments without the expense associated with wind tunnel testing. Implicit in this approach is the assumption that the design process will utilize the CFD solutions in much the same manner as wind tunnel measurements, where design improvements result largely from the designers' intuition and experience. However, it was recognized early on that coupling CFD with numerical optimization could lead to a more efficient approach to aerodynamic design. For example, Hicks et al.[5] developed a procedure for optimum design of nonlifting airfoils based on the numerical solution of the inviscid, transonic, small-disturbance equations. Although constrained by the computers and algorithms available at the time, their work demonstrated the potential of aerodynamic shape optimization to play an important role in aircraft design and motivated further research.

Simply put, the goal of aerodynamic shape optimization is to determine the geometry, for example of a wing, that minimizes an objective function subject to some constraints under a particular range of operating conditions. We will delve further into the complexities of both the methodology and the problem formulation, but first we make an important distinction between *design* and *optimization*. Aircraft design[1] is a complex task involving numerous considerations, from safety to cost, a wide array of disciplines, such as aerodynamics, structures, controls, and aeroacoustics, and an enormous range of operating conditions under which the aircraft must operate safely and efficiently. In principle it is possible to determine the aircraft that minimizes

[1] The discussion here concentrates on aircraft design, but the field of aerodynamic shape optimization can be applied more broadly, and an analogous discussion will apply to other applications where aerodynamic design is required.

a particular objective function, such as direct operating cost, while considering all of the relevant disciplines and operating conditions based on estimates of uncertain quantities such as the future price of fuel. However, this is a herculean computational task far beyond the capabilities of computers for the foreseeable future. Therefore, the process of design involves intelligent use of the available tools in combination with knowledge and experience to develop a near optimal design in an appropriate time frame. Moreover, while the computer can find the optimal aircraft once the objectives and constraints have been defined, these depend on the priorities of the designers and their development cannot be reduced to a strictly computational task. This is the key distinction between design and optimization: the former is a distinctly human task involving setting of priorities,[2] while the latter is a strictly numerical task that can be accomplished by a computer.

Having clarified the distinction between design and optimization, we are now in a position to state the primary role of optimization within the design process. Numerical optimization frees the designers from the task of finding the geometry that minimizes the specified objective, enabling them to focus on the tasks where human input is essential, namely determining the overall problem formulation, including the objectives and constraints. Furthermore, numerical optimization can facilitate rapid study of the impact of changes in priorities and provide a detailed quantitative understanding of trade-offs that is critical information for the designers.

One can categorize design into the following three categories: routine design, innovative design, and creative design [2]. In routine design a solution is sought within a known design space; in innovative design an extended design space is considered; in creative design an entirely new design space is examined. Since past design experience is of limited value when a new design space is explored, the value of optimization is highest for creative design. Moreover, conceptual design tools will not be well developed for the new design space, so physics-based modeling is essential. The design category also influences the manner in which the optimization is carried out. In creative design, it is important that the optimization problem be formulated such that the optimizer has as much freedom as possible, without being constrained by preconceived notions of the designers, which may be inapplicable in the new design space under consideration.

Aerodynamic shape optimization can be used alone or in conjunction with other disciplines in the context of *multidisciplinary optimization*. Given our focus here, we will concentrate on the former, but aerodynamic shape optimization is a critical component of a multidisciplinary optimization methodology and is used extensively in that context. Moreover, the reader is cautioned that when used alone aerodynamic shape optimization must be used judiciously such that other disciplines are not compromised. This involves the use of appropriate constraints as surrogates for missing disciplines as well as application to problems where aerodynamics is the critical discipline and the optimizer cannot exploit the absent disciplines.

The objective function to be minimized is typically a functional such as aerodynamic drag that depends on the flow field. The flow field depends on a number of

[2] *pace* the artificial intelligence community

parameters that define the problem, including the shape of the geometry and the operating conditions. In CFD, the flow solution is approximated by solving the governing partial differential equations (PDEs) on a mesh of finite resolution. In aerodynamic shape optimization, a set of parameters that we will call *design variables* is controlled by an optimization algorithm, which seeks to minimize the objective function subject to some constraints. The design variables may include some of the parameters defining the problem, or they may introduce additional variables that control some of the problem–defining parameters. Moreover, the design variables typically control those parameters that define the shape of geometry, often including the angle of incidence to the oncoming flow, but can also control other parameters, such as the Mach number, if the designer has the freedom to vary those parameters.

Most numerical optimization algorithms begin with an initial set of design variables and iteratively modify those variables until the desired minimum is found. In shape optimization, this means that an initial shape is progressively modified until the objective is minimized. The following steps are typically carried out:

1. With the design variables set to their initial values, i.e. for the initial geometry, solve the flow problem and compute the resulting objective function value.
2. If a gradient-based optimization algorithm is to be used, compute the gradient of the objective function with respect to the design variables. If a gradient-free algorithm is used, this step is not needed.
3. Provide the design variable values, the objective function value, and, if necessary, the gradient to the optimization algorithm, which will return a new set of design variables.
4. Determine the new geometry that is associated with the new set of design variables.
5. Deform the mesh such that it conforms to the modified geometry. In principle one can regenerate the mesh for each new geometry, but it is often more efficient to deform the original mesh instead.
6. Evaluate the flow field, compute the objective function, and, if necessary, the gradient for the new geometry.
7. Assess whether the optimization has converged sufficiently to exit. If not, return to step 3 and repeat.

Based on these steps, the following are the components of an aerodynamic shape optimization methodology:

- a geometry parameterization,
- a method for controlling the geometry through its parameterization,
- a method for deforming the mesh,
- a flow evaluation method,
- a method for computing the gradient (if needed), and
- an optimization algorithm.

The purpose of this book is to provide the reader with a basic understanding of aerodynamic shape optimization, including both its application, i.e. considerations in the formulation of the optimization problem, as well as key aspects of the development, implementation, and choice of algorithms for each of the above components.

1.2 Roadmap

Chapter 2 will discuss the formulation of an aerodynamic shape optimization problem, which involves the specification of operating conditions or a range of operating conditions, objectives, constraints, and design variables. Multipoint optimization is described for handling a range of operating conditions, and robust design is briefly introduced. The chapter concludes with a description of Pareto fronts, their use in understanding trade-offs, and how this can inform the problem formulation.

The relative merits of gradient-based and gradient-free optimization algorithms in the context of aerodynamic shape optimization are presented in Chapter 3. Two methods for gradient computation are then described: the direct method (also known as the flow sensitivity method) and the adjoint method. The latter is of particular importance in aerodynamic shape optimization and hence is presented in some detail.

Chapter 4 presents the four major components of an aerodynamic shape optimization methodology: i) geometry parameterization and control, ii) mesh deformation, iii) numerical solution of the flow equations, and iv) gradient computation via the adjoint method. Throughout this chapter the algorithms used in the Jetstream aerodynamic shape optimization methodology are used as examples of each of the various components. The algorithms in Jetstream are described in detail in Hicken and Zingg [4], Hicken and Zingg [3], Osusky and Zingg [7], Osusky et al. [6], and Gagnon and Zingg [1]. Emphasis is on steady flows throughout, but the algorithms described are generally applicable to unsteady flows as well. For example, application of the adjoint method to unsteady flows is described by Rumpfkeil and Zingg [8].

References

1. Gagnon, H., Zingg, D.: Two-level free-form and axial deformation for exploratory aerodynamic shape optimization. AIAA Journal **53** (2015)
2. Gero, J., Maher, M.: Modelling Creativity and Knowledge-Based Creative Design. Lawrence Erlbaum Associates (1993)
3. Hicken, J., Zingg, D.: Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement. AIAA Journal **48** (2010)
4. Hicken, J.E., Zingg, D.W.: A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms. AIAA Journal **46**(11), 2773–2786 (2008). DOI 10.2514/1.34810. URL http://dx.doi.org/10.2514/1.34810
5. Hicks, R., Murman, E.M., Vanderplaats, G.: An assessment of airfoil design by numerical optimization. TM X-3092, NASA (1974)
6. Osusky, L., Buckley, H., Reist, T., , Zingg, D.: Drag minimization based on the Navier-Stokes equations using a Newton-Krylov approach. AIAA Journal **53** (2015)
7. Osusky, M., Zingg, D.: Parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations. AIAA Journal **51** (2013)
8. Rumpfkeil, M., Zingg, D.: The optimal control of unsteady flows with a discrete adjoint method. Optimization and Engineering **11** (2010)

Chapter 2
Problem Formulation

This chapter introduces the concept of partial differential equation constrained optimization problems and discusses the formulation of aerodynamic shape optimization problems. An optimization methodology can be very good at exploiting a weakness in the problem formulation; hence a sound formulation is critical to achieving the desired outcome. Moreover, it can be necessary to modify the problem formulation based on the results obtained. For example, if analysis of an optimized geometry reveals inadequacies in its performance, the designer must revise the objective or constraints to address these limitations.

2.1 Constrained Optimization Problems

A constrained optimization problem involves the minimization of a specified objective function of a set of variables called *design variables* subject to a set of constraints that can be equality or inequality constraints. This can be written as

$$\begin{aligned} &\min_X J(X) \\ &\text{subject to} \\ &c_{\text{eq}}(X) = 0 \\ &c_{\text{in}}(X) \leq 0\,, \end{aligned} \tag{2.1}$$

where J represents the objective function, X the design variables, and $c_{\text{eq}} = 0$ and $c_{\text{in}} \leq 0$ the equality and inequality constraints respectively.

2.2 Partial Differential Equation Constrained Optimization Problems

Aerodynamic shape optimization falls under the general subject of partial differential equation (PDE) constrained optimization. In PDE-constrained optimization the objective function depends not only on the design variables X but also on another set of variables Q that is the solution of a PDE, i.e. $J = J(Q, X)$. In CFD, Q represents the flow variables, and the PDE is solved numerically by discretizing the governing equations, e.g. the steady Navier-Stokes equations, in space and solving the resulting system of nonlinear algebraic equations. This nonlinear system of equations is represented by $R(Q, X) = 0$, where R represents the discrete residual. Therefore (2.1) becomes

$$\begin{aligned} &\min_X J(Q, X) \\ &\text{subject to} \\ &R(Q, X) = 0 \\ &c_{\text{eq}}(Q, X) = 0 \\ &c_{\text{in}}(Q, X) \leq 0 \,. \end{aligned} \tag{2.2}$$

Note the added PDE constraint and that the constraints now depend on Q as well as X. In aerodynamic shape optimization, the design variables X typically control the shape of the geometry. Some constraints, such as a thickness constraint, may depend solely on X, while others, such as a lift constraint, may depend on the flow field Q as well.

An important consideration in the selection and development of algorithms for (2.2) is that evaluation of the objective function, which requires the solution of $R(Q, X) = 0$, i.e. the solution of the CFD problem, is expensive. A second characteristic of aerodynamic shape optimization is that the number of design variables is typically much larger than the number of constraints that depend on the flow solution. We shall see in Chapter 3 that these two properties lead directly to the choice of gradient-based optimization algorithms where the gradient is computed using the adjoint method.

2.3 Design Variables, Objectives, Constraints, and Operating Conditions

The problem formulation involves specification of an objective, constraints, operating conditions, and design variables. The problem formulation is critical to a successful outcome. Some trial and error may be needed to reach the necessary formulation. For example, an unintended outcome of an initial optimization may reveal a shortcoming in the problem formulation that can be rectified through an additional constraint or

other change in the problem formulation. Moreover, the designers' priorities can be influenced by the knowledge gained from trade-off studies enabled by varying a parameter during optimization. This will be further discussed in Section 2.5.

The choice of the design variables, i.e. the geometry control, is determined by the geometric freedom the designers wish to permit. This will have a substantial effect on the resulting optimized geometry and the number of iterations of the optimization algorithm needed to converge. For example, if the designer seeks a planar wing with a straight leading edge, it will be counterproductive to introduce design variables that enable the optimizer to generate a nonplanar wing with a curved leading edge. It is possible to introduce constraints to ensure the desired properties, but it is usually more effective not to introduce undesired freedom in the first place. However, if the design falls into the creative design category with the expectation that the optimizer will explore a hitherto unknown design space, then the design variables should be chosen with as few preconceived restrictions as is feasible. It is important to ensure that the design variables are not redundant, i.e. that the same geometry cannot be achieved with two different sets of design variable values. It is also usually advisable to include the angle of attack as a design variable, as a fixed angle of attack can unnecessarily constrain the design.

Perhaps the most common objective function is drag (or drag coefficient if the projected area is constrained), but other objectives, such as maximum lift, lift to drag ratio, endurance factor, etc., may also be used depending on the nature of the design problem. Constraints can be divided into flow-dependent constraints and purely geometric constraints. In aircraft and wing design, the most common flow-dependent constraint is an equality constraint for lift, whereas an inequality constraint on the pitching moment may be imposed on a wing optimization, and trim or static margin constraints can be needed for optimization of full aircraft. Several other constraints can be required for stability and control purposes and off-design requirements.

Geometric constraints can be required for practical reasons or to ensure that a wing's structural integrity is not compromised. For example, a volume constraint can be enforced to ensure that a wing can store the required amount of fuel, and constraints can be imposed on a fuselage or blended wing-body to ensure specific cabin dimensions. An aircraft angle of attack constraint is usually needed to keep the cabin floor from being excessively inclined. Thickness constraints are often imposed during optimization of wings for structural reasons.

The basic operating conditions needed for a CFD computation are the Reynolds number and the Mach number. Together with the lift constraint, these can be determined from the speed, altitude, and aircraft weight. A great deal can be learned by performing an optimization at a single operating condition consisting of a specified Reynolds number, Mach number, and lift constraint. However, aircraft inevitably operate under a range of conditions. Even a cruise configuration must operate effectively over a range of speeds, altitudes, and aircraft weights, and hence a range of lift coefficients, Mach numbers, and Reynolds numbers. Moreover, there are off-design requirements related to low-speed operation, buffet, dive conditions, flutter, etc. There are also considerations related to take-off and landing performance when

high-lift systems are deployed. Nevertheless, it is unlikely to be feasible to include all relevant flight conditions, and the designers will be expected to reduce the number of conditions to be included in the optimization based on their knowledge of the problem. In order to perform aerodynamic shape optimization over a range of operating conditions, multipoint optimization can be performed where the objective function is a composite of objective functions at the various operating conditions included; multipoint optimization is further discussed in the next section.

We next present an example of a single-point lift-constrained drag minimization problem formulation. Sample results obtained applying Jetstream to this problem are provided in Section 4.5. The initial geometry is the NASA Common Research Model (CRM) Wing, and the geometric freedom allowed includes the section shape along the wing, the twist, and the angle of attack. With this limited freedom the projected surface area is constant and does not need to be constrained. Geometric constraints include the requirement that the wing volume be greater than or equal to the volume of the initial wing. In addition, the thickness must be no less than 85% of the thickness of the initial wing at all chordwise and spanwise locations. The objective function is the drag coefficient, which is effectively the drag, since the projected surface area is fixed. The following flow-dependent constraints are enforced:

$$\begin{aligned} C_L &= 0.50 \\ C_M &\geq -0.17\,, \end{aligned} \tag{2.3}$$

where C_L is the lift coefficient and C_M the pitching moment coefficient (defined positive nose upward). The Mach number is 0.85 and the Reynolds number is 5 million.

In assessing the results of an aerodynamic shape optimization problem, it can be important to establish which constraints are active once the optimization has converged and which are inactive. A constraint that is inactive at convergence will not have any impact on the final result but can be useful in aiding convergence by preventing geometries from arising that might cause difficulties, for example, for the mesh deformation or flow evaluation methods. A constraint that is active at convergence will affect the performance of the final geometry, and it is often valuable to conduct studies to establish the sensitivity of the objective function value to the constraint value. If there is considerable potential improvement available by relaxing the constraint, this is important information for the designer, who can then consider alternative ways of achieving the goal that the constraint is attempting to achieve that might not penalize the performance to the same extent. This is an example of how the results of an optimization can provide information that can enable the designer to improve the problem formulation.

2.4 Multipoint Optimization and Robust Design

As discussed in Section 2.3, an aerodynamic geometry must function over a range of operating conditions, necessitating multipoint optimization. Buckley and Zingg [1] present one approach to this subject. The range of on-design operating conditions is addressed through an integral formulation of the objective function, while off-design performance requirements are included as constraints. For example, consider an aircraft where the designer wishes to minimize a weighted average of the drag coefficient over the range of Mach numbers M_1 to M_2, the range of aircraft weights W_1 to W_2, and the range of altitudes h_1 to h_2 associated with the cruise conditions expected under normal operation, where the lift must equal the weight of the aircraft at any operating condition. This can be achieved through the following objective function:

$$\int_{h_1}^{h_2} \int_{W_1}^{W_2} \int_{M_1}^{M_2} C_{\mathrm{D}}(M, W, h) \mathcal{Z}(M, W, h) \mathrm{d}M \mathrm{d}W \mathrm{d}h \, . \tag{2.4}$$

The freestream density is dependent on the altitude; this affects the calculation of the lift, the speed of sound (which determines the relation between the speed and the Mach number), and the Reynolds number. Hence the drag coefficient is a function of Mach number, aircraft weight, and altitude, as shown in (2.4). The function $\mathcal{Z}(M, W, h)$ is a weighting function that can be used by the designer to prioritize particular regions of the operating space, for example based on knowledge of the relative time the aircraft is expected to spend flying under particular conditions.

The integrals in (2.4) can be approximated by a quadrature rule such as the trapezoidal rule to obtain

$$J = \sum_{i=1}^{N_h} \sum_{j=1}^{N_W} \sum_{k=1}^{N_M} \tau_{i,j,k} C_{\mathrm{D}}(M_k, W_j, h_i) \mathcal{Z}(M_k, W_j, h_i) \Delta M \Delta W \Delta h \, , \tag{2.5}$$

where $\tau_{i,j,k}$ are the weights associated with the quadrature rule, ΔM, ΔW, and Δh are the spacings between the quadrature points, and N_M, N_W, N_h are the associated numbers of quadrature points for M, W, and h, respectively. A sufficient number of quadrature points must be used such that the integral is well approximated. This formulation provides a flexible approach to optimization over a specified range of on-design operating conditions, such as cruise conditions of an aircraft. Additional considerations, such as buffet, flutter, stability and control, and performance requirements under low-speed and dive conditions, can be added as constraints.

The performance of an aerodynamic geometry is expected to be relatively insensitive to an appropriate degree of variability in parameters such as Mach number, lift, and even the geometry itself. Such robustness is inevitably associated with some reduction in the performance that can be achieved at a single operating condition. This trade-off is handled through techniques for *robust design* or *design under uncertainty*. Some knowledge of the expected uncertainties is needed, and including these uncertainties can be expensive; hence development of efficient approaches to robust aerodynamic design is currently an active area of research. The subject of

design under uncertainty will not be covered in this book, but it is worth noting that a design that results from a multi-point optimization over a range of values of a parameter such as Mach number will generally provide robust performance with respect to that parameter.

2.5 Pareto Fronts and the Study of Trade-offs

One important function of aerodynamic shape optimization, and optimization in general, is its ability to provide the designer with quantitative information about trade-offs. This information can be used to select appropriate priorities among competing objectives. A Pareto front provides a convenient means of studying such trade-offs. Consider two competing objectives J_1 and J_2.[1] In aircraft design, these could be the endurance and range of an unmanned aerial vehicle or the direct operating cost and an environmental impact metric of a transport aircraft. A sample Pareto front is depicted in Fig. 2.1. It represents the locus of designs that are non-dominated. This means that for all designs on the front, there exist no designs that are superior in terms of both objective functions; designs can exist that are superior with respect to one objective or the other, but not both. All designs above and to the right of the front are dominated by designs on the front; there are no designs below and to the left of the front. The design at the upper left end of the front (indicated by the circular symbol) arises if J_1 is minimized and J_2 is ignored. As one moves along the front from upper left to lower right, the emphasis on J_1 is decreasing and that on J_2 increasing. The lower right point on the Pareto front (indicated by the square symbol) is associated with minimization of J_2 alone.

Such Pareto fronts can be generated by defining a composite objective function

$$J = (1 - \lambda)J_1 + \lambda J_2 \, . \tag{2.6}$$

The composite objective function is then minimized for values of λ varying from 0 to 1. An alternative approach is as follows. First minimize J_1 and J_2 independently to obtain the two endpoints of the front. Then minimize J_1 with an equality constraint on J_2 where J_2 is successively constrained to values varying between its value at the two endpoints. These two approaches should generate identical Pareto fronts but the second approach may be preferred if the front has a concave (downward) portion. Other methods for generating Pareto fronts are presented in the literature [2].

While a Pareto front does not provide a definitive answer with respect to the weighting that should be assigned to the competing objectives, it provides quantitative information with respect to the trade-off between the two objectives. For example, if a portion of the front is nearly vertical, that means that J_2 can be reduced with a small penalty in J_1, so this trade should probably be made. Similarly, a nearly horizontal portion of the front indicates that the reduction in J_1 is much

[1] The Pareto front concept can be extended to an arbitrary number of dimensions but here we will consider only two competing objectives.

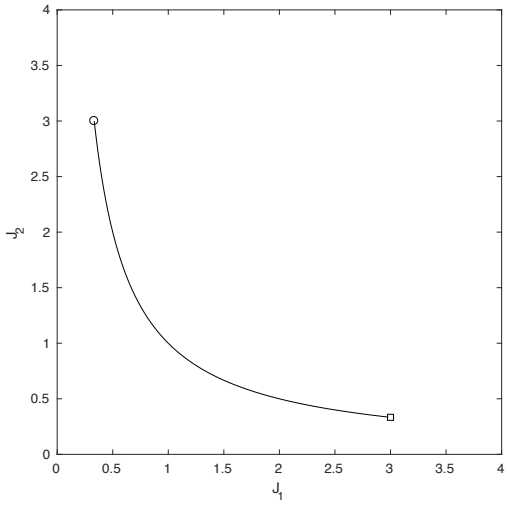


Fig. 2.1 Example of a Pareto front

larger than the increase in J_2, again suggesting that this trade is likely worthwhile. If the designer has an a priori basis for quantitatively weighting the two objectives, then it is not necessary to generate a Pareto front. In many design studies, however, the quantitative information on trade-offs provided by a Pareto front can be highly useful in enabling the designer to establish priorities among competing objectives.

References

1. Buckley, H., Zingg, D.: Approach to aerodynamic design through numerical optimization. AIAA Journal **51** (2013)
2. Das, D., Dennis, J.: Normal–boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM Journal on Optimization **8** (1998)

Chapter 3
Gradient Computation

This chapter begins by outlining the motivation for choosing a gradient-based optimization algorithm. This is followed by an introduction to the adjoint method, which is typically used to compute the gradient in aerodynamic shape optimization. The advantage of the adjoint method is seen in problems where the number of design variables exceeds the number of functions for which the gradient is required, including the objective function and any constraints that depend on the flow variables, a property of the vast majority of aerodynamic shape optimization problems. In order to enable a thorough understanding, the adjoint method is presented from a number of different perspectives. Computation of the gradient via the solution of the adjoint equation is further discussed in Chapter 4.

3.1 Gradient-Based vs. Gradient-Free Optimization Algorithms

There exist a wide variety of gradient-free optimization algorithms, exemplified by genetic (or evolutionary) algorithms [4]. These typically search the design space through a combination of exploitation of the positive attributes of known solutions with exploration of new areas of the design space. Such algorithms have a number of useful properties. For example, they are generally nonintrusive, i.e. they can be combined with a flow algorithm with little extra effort. Moreover, they can potentially find the global minimum in a problem with multiple local minima. Finally, they avoid the cost of computing the gradient. In contrast, a gradient-based algorithm will generally find a local minimum and, especially if the adjoint method is used, can require close coupling with the flow solution algorithm. The primary advantage of the gradient-based approach is that it converges much faster than the gradient-free approach. Many gradient-based optimization methods have the potential for superlinear convergence with respect to the number of design variables, and linear scaling is seen in practice [6]. In contrast, convergence of, for example, genetic algorithms is much slower, and the cost increases rapidly with the number of design

variables [6]. Hence gradient-free algorithms are better suited to problems where the function evaluations are inexpensive. For aerodynamic shape optimization, which is characterized by very expensive function evaluations and a large number of design variables, the gradient-based approach is highly advantageous [6]. For problems with a large number of design variables, adjoint methods contribute greatly to the efficiency of gradient-based methods. The key property of the adjoint method is that it renders the gradient computation essentially independent of the number of design variables, enabling efficient gradient computations when the number of design variables is large.

For gradient-based methods, the issue of convergence to a local minimum cannot be ignored. One cannot know beforehand whether the design space is convex and has a single minimum. However, experience has shown that some aerodynamic shape optimization problems have convex design spaces [1]. Moreover, global optimization methods that are based on gradient-based algorithms can be used efficiently in the solution of problems with multiple local minima [1].

The gradient-based optimization package used by Jetstream, which is quite commonly used in aerodynamic shape optimization, is known as SNOPT [3]. It is based on sequential quadratic programming and is intended for constrained optimization problems with many design variables. Gradients of the objective function and any flow-dependent constraints must be provided to SNOPT. These are calculated using the adjoint method described in the next section.

3.2 The Adjoint Method

The adjoint method will be presented in several different ways, beginning with the approach of Jameson et al. [5]. The goal is to minimize an objective function J that depends on the flow variables $\mathcal{Q}$ and a function representing the shape of the geometry $\mathcal{X}$:

$$J = J(\mathcal{Q}, \mathcal{X}) \; . \tag{3.1}$$

Here we use a calligraphic font to emphasize that here $\mathcal{Q}$ and $\mathcal{X}$ represent functions, not a discrete solution or a vector of design variables, respectively. The partial differential equation governing the flow is given by

$$\mathcal{R}(\mathcal{Q}, \mathcal{X}) = 0 \; . \tag{3.2}$$

A change in $\mathcal{X}$ produces a change in $\mathcal{Q}$ such that $\mathcal{R}$ remains zero. This can be represented by

$$\delta\mathcal{R} = \frac{\partial \mathcal{R}}{\partial \mathcal{Q}} \delta\mathcal{Q} + \frac{\partial \mathcal{R}}{\partial \mathcal{X}} \delta\mathcal{X} = 0 \; . \tag{3.3}$$

The resulting change in the objective function is

$$\delta J = \frac{\partial J}{\partial \mathcal{Q}} \delta\mathcal{Q} + \frac{\partial J}{\partial \mathcal{X}} \delta\mathcal{X} \; . \tag{3.4}$$

Now introduce the Lagrange multiplier ψ in the following manner

$$\delta J = \frac{\partial J}{\partial Q}\delta Q + \frac{\partial J}{\partial \mathcal{X}}\delta \mathcal{X} - \psi^T \left[\frac{\partial \mathcal{R}}{\partial Q}\delta Q + \frac{\partial \mathcal{R}}{\partial \mathcal{X}}\delta \mathcal{X}\right] , \tag{3.5}$$

where we note that the quantity in square brackets is equal to zero from (3.3). This equation can be rearranged to obtain

$$\delta J = \left[\frac{\partial J}{\partial Q} - \psi^T \frac{\partial \mathcal{R}}{\partial Q}\right]\delta Q + \left[\frac{\partial J}{\partial \mathcal{X}} - \psi^T \frac{\partial \mathcal{R}}{\partial \mathcal{X}}\right]\delta \mathcal{X} . \tag{3.6}$$

If we choose ψ such that the first term in square brackets is zero, then we obtain

$$\delta J = \left[\frac{\partial J}{\partial \mathcal{X}} - \psi^T \frac{\partial \mathcal{R}}{\partial \mathcal{X}}\right]\delta \mathcal{X} , \tag{3.7}$$

and hence the gradient of the objective function is given by

$$\mathcal{G} = \frac{\partial J}{\partial \mathcal{X}} - \psi^T \frac{\partial \mathcal{R}}{\partial \mathcal{X}} . \tag{3.8}$$

The key point is that δQ has been eliminated; as a result the gradient can be found without additional flow evaluations.

The first term in square brackets in (3.6) is zero if ψ satisfies

$$\psi^T \frac{\partial \mathcal{R}}{\partial Q} = \frac{\partial J}{\partial Q} , \tag{3.9}$$

or

$$\left(\frac{\partial \mathcal{R}}{\partial Q}\right)^T \psi = \left(\frac{\partial J}{\partial Q}\right)^T . \tag{3.10}$$

This is a linear partial differential equation for the Lagrange multiplier, which requires suitable boundary conditions.

Consider now the same analysis but replacing Q with a discrete vector of flow variables Q, $\mathcal{X}$ with a vector of design variables representing the geometry through a parameterization X, and $\mathcal{R}(Q, \mathcal{X})$ with a discrete residual vector resulting from the spatial discretization of the partial differential equations governing the flow $R(Q, X)$. This results in the following equation for the Lagrange multiplier:

$$\left(\frac{\partial R}{\partial Q}\right)^T \psi = \left(\frac{\partial J}{\partial Q}\right)^T . \tag{3.11}$$

This is a linear system of equations where the matrix on the left-hand side is the transpose of the Jacobian of the discrete residual, while the vector on the right-hand side is the partial derivative of the objective function with respect to the discrete solution variables. The gradient is then given by

$$G = \frac{\partial J}{\partial X} - \psi^T \frac{\partial R}{\partial X} \,. \tag{3.12}$$

These two approaches are classified as the *continuous* and *discrete* adjoint methods, respectively. In the continuous approach one first differentiates the continuous equations to obtain the equation for the Lagrange multipliers (3.10) and then discretizes the resulting equations in order to solve for the Lagrange multipliers. In the discrete approach, one first discretizes the governing flow equations and then differentiates to obtain the equation for the Lagrange multipliers (3.11). While there may be contexts where the continuous approach is preferred, the discrete approach has two important advantages. First, there is less work involved in determining suitable boundary conditions, as these follow directly from the discrete boundary conditions in the discrete case. Second, the gradient determined using the discrete approach is the actual gradient of the objective function computed by solving the discretized equations. In contrast, the continuous approach can produce a gradient that is inconsistent with the discrete gradient. For these reasons we will henceforth concentrate the discrete adjoint method.

We will next present a different perspective that is more general and makes clear why the Lagrange multipliers are called *adjoint variables* [2]. The gradient can be written in the form

$$\frac{\mathrm{d}J}{\mathrm{d}X} = \frac{\partial J}{\partial Q}\frac{\mathrm{d}Q}{\mathrm{d}X} + \frac{\partial J}{\partial X} \,, \tag{3.13}$$

where we assume a single design variable in order to simplify the notation. The primary computational challenge in determining the gradient is related to evaluating the product on the right-hand side of (3.13), which we write as

$$\frac{\partial J}{\partial Q}\frac{\mathrm{d}Q}{\mathrm{d}X} = g^T u \,, \tag{3.14}$$

where we have introduced the variables g and u as shown. The term $\mathrm{d}Q/\mathrm{d}X = u$, is the solution of a linear system of equations resulting from a linearization of the discretized governing equations, which we write as

$$Au = f \,. \tag{3.15}$$

We seek to evaluate $g^T u$, where u is the solution to the linear system (3.15); this is called the *primal* problem. The *dual* or *adjoint* form is to evaluate $v^T f$, where v is the solution to the linear system

$$A^T v = g \,. \tag{3.16}$$

The equivalence of the primal and dual forms follows from

$$v^T f = v^T A u = (A^T v)^T u = g^T u \,. \tag{3.17}$$

Now we are in a position to relate this to the Lagrange multiplier approach described previously. Let

$$\begin{aligned} g^T &= \frac{\partial J}{\partial Q} \\ u &= \frac{\mathrm{d}Q}{\mathrm{d}X} \\ A &= \frac{\partial R}{\partial Q} \\ v &= \psi \\ f &= -\frac{\partial R}{\partial X}\,, \end{aligned} \tag{3.18}$$

where we continue to assume a single design variable. We then obtain for the gradient

$$\begin{aligned} \frac{\mathrm{d}J}{\mathrm{d}X} &= g^T u + \frac{\partial J}{\partial X} \\ &= v^T f + \frac{\partial J}{\partial X} \end{aligned} \tag{3.19}$$

as a result of the equivalence of the primal and dual forms. Hence one can obtain the gradient either by solving the primal form (3.15) for u or by solving the dual problem (3.16) for v. The difference is that the primal problem must be solved for each design variable independent of the number of functions for which the gradient is required, while the dual problem must be solved for each function independent of the number of design variables. Hence the dual, or adjoint, approach is much more efficient when the number of design variables exceeds the number of functions, which is typically the case in aerodynamic shape optimization. Note that the functions for which the gradient is needed include the objective function as well as any constraints that depend on the flow solution, such as a lift constraint.

The primal approach is often called the direct or flow sensitivity method. With the residual of the discretized flow equations given by $R(Q, X) = 0$, the derivative with respect to a design variable is

$$\frac{\mathrm{d}R}{\mathrm{d}X} = \frac{\partial R}{\partial Q}\frac{\mathrm{d}Q}{\mathrm{d}X} + \frac{\partial R}{\partial X} = 0\,. \tag{3.20}$$

This derivative is equal to zero because $R(Q, X) = 0$ for arbitrary X, i.e. when X is modified, the discrete flow equations are solved to find a new Q such that $R(Q, X) = 0$. Hence Q is an implicit function of X and $\mathrm{d}Q/\mathrm{d}X$ can be found by solving the linear system

$$\frac{\partial R}{\partial Q}\frac{\mathrm{d}Q}{\mathrm{d}X} = -\frac{\partial R}{\partial X}\,, \tag{3.21}$$

which is (3.15) above with the substitutions given in (3.18). This linear system must be solved once for each design variable. However, once $\mathrm{d}Q/\mathrm{d}X$ is known for a given design variable, the gradient of any function can be found by substituting it into (3.13). As described above, this approach is efficient if the number of functions for which gradients are required exceeds the number of design variables, which is rarely the case in aerodynamic shape optimization.

Equations (3.13) and (3.20) provide another way to understand the adjoint method. From (3.20) we have

$$\frac{\mathrm{d}Q}{\mathrm{d}X} = -\left(\frac{\partial R}{\partial Q}\right)^{-1} \frac{\partial R}{\partial X} \,. \tag{3.22}$$

Substituting this into (3.13) gives

$$\frac{\mathrm{d}J}{\mathrm{d}X} = -\underbrace{\frac{\partial J}{\partial Q}\left(\frac{\partial R}{\partial Q}\right)^{-1}}_{\psi^T} \frac{\partial R}{\partial X} + \frac{\partial J}{\partial X} \,. \tag{3.23}$$

If the product indicated is defined as ψ^T, (3.11) and (3.12) are obtained, the former as follows:

$$\psi = \left[\frac{\partial J}{\partial Q}\left(\frac{\partial R}{\partial Q}\right)^{-1}\right]^T = \left(\frac{\partial R}{\partial Q}\right)^{-T}\left(\frac{\partial J}{\partial Q}\right)^T \,, \tag{3.24}$$

or

$$\left(\frac{\partial R}{\partial Q}\right)^T \psi = \left(\frac{\partial J}{\partial Q}\right)^T \,, \tag{3.25}$$

which is the dual or adjoint form (3.16).

With these different perspectives as background, we now present a final approach to the adjoint method, concentrating on the discrete adjoint method and assuming a single design variable for notational clarity. Consider the following Lagrangian function:

$$L(Q, X, \psi) = J(Q, X) - \psi^T R(Q, X) \,. \tag{3.26}$$

Setting the partial derivatives of L with respect to ψ, Q, and X, respectively, to zero gives the first-order optimality conditions for the PDE-constrained optimization problem defined by (2.2):

$$\begin{aligned}
\frac{\partial L}{\partial \psi} &= 0 = R(Q, X) && \text{the discrete flow problem} \\
\frac{\partial L}{\partial Q} &= 0 = \frac{\partial J}{\partial Q} - \psi^T \frac{\partial R}{\partial Q} && \text{the adjoint problem} \\
\frac{\partial L}{\partial X} &= 0 = \frac{\partial J}{\partial X} - \psi^T \frac{\partial R}{\partial X} && \text{the gradient} \,.
\end{aligned} \tag{3.27}$$

These three conditions constitute a coupled system and can be solved as such in what is sometimes known as a *one-shot* method. However, it is more common to solve them sequentially in conjunction with an optimization algorithm. Given a shape represented by specific values in the vector X, the discrete flow problem can be solved for Q, which enables J to be computed. With X and Q known, the discrete adjoint problem can be solved for ψ. Finally, with X, Q, and ψ known, the gradient can be calculated and, along with the objective function value, provided to the optimization algorithm, which will update the design variables X.

This section has provided the reader with an introduction to the adjoint method for computing the gradient. One aspect not yet discussed relates to the mesh sensitivities, which must be properly treated in order to obtain an accurate gradient. These are discussed in Section 4.4, along with additional aspects of computing the gradient using the discrete adjoint method.

References

1. Chernukhin, O., Zingg, D.: Multimodality and global optimization in aerodynamic design. AIAA Journal **51** (2013)
2. Giles, M., Pierce, N.: An introduction to the adjoint approach to design. Flow, Turbulence and Combustion **65** (2000)
3. Gill, P., Murray, W., Saunders, M.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Journal on Optimization **12** (2002)
4. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, Mass. (1989)
5. Jameson, A., Martinelli, L., Pierce, N.: Optimum aerodynamic design using the navier-stokes equations. Theoretical and Computational Fluid Dynamics **10** (1998)
6. Zingg, D., Nemec, M., Pulliam, T.: A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. Revue Europeene de Mecanique Numerique **17** (2008)

Chapter 4
Components of an Aerodynamic Shape Optimization Methodology

In this chapter, the basic components of an aerodynamic shape optimization methodology are presented and discussed. The components of the Jetstream methodology are used as an example of each. In Jetstream, geometry parameterization is accomplished using a number of B-spline surfaces. The parameters that define the analytical representation of the surfaces are the coordinates of the B-spline surface control points, which we designate X_p. Geometry control is achieved through the free-form deformation (FFD) approach; the coordinates of the control points of the FFD volume are designated X_c. The design variables X are some subset of X_c, often along with the angle of attack as an additional design variable. Through the B-spline formulation, X_p is an explicit function of X_c, which in turn is an explicit function of X; hence we can write $X_\text{p} = X_\text{p}(X_\text{c}(X))$.

Mesh deformation in Jetstream is accomplished through an efficient approach that is integrated with the geometry parameterization. The coordinates of the grid nodes G are controlled by a number of B-spline volumes some of which have the B-spline surfaces defining the geometry as one of their faces. The coordinates of the B-spline volume control points C for a given set of design variables X are found by solving a discrete form of the linear elasticity equations, which we write as

$$M(C, X) = 0 \, . \tag{4.1}$$

We choose this form rather than $M(C, X_\text{p}) = 0$ for clarity in Section 4.4, but the reader should be aware that

$$\frac{\partial M}{\partial X} = \frac{\partial M}{\partial X_\text{p}} \frac{\partial X_\text{p}}{\partial X_c} \frac{\partial X_\text{c}}{\partial X} \tag{4.2}$$

via the chain rule. The coordinates of the grid nodes are then found from the explicit function $G(C)$. In order to include the influence of the mesh on the discrete flow residual, we will henceforth write $R = R(Q, C, X)$. However the reader should keep in mind that the residual explicitly depends on the coordinates of the grid nodes G rather than the coordinates of the B-spline volume control points C, i.e. $R = R(Q, G(C), X)$.

The chapter concludes with results obtained using Jetstream for the example aerodynamic shape optimization problem presented in Section 2.3.

4.1 Geometry Parameterization and Control

We distinguish between parameterization and control of the geometry. The geometry parameterization provides a representation of the geometry that is described by a finite number of parameters. If these parameters are used directly as design variables, then there is no distinction between parametrization and control. On the other hand, if a separate set of design variables controls the parameters that represent the geometry, this decouples parameterization and control, which can be advantageous.

In some ways the most natural way to parameterize a geometry is through a computer aided design (CAD) package, where the underlying parameterization is often based on non-uniform rational B-splines (NURBS). A CAD-based geometry parameterization has numerous advantages in that it provides a common geometry with designers from other areas of specialization. However, it may also provide unneeded capability and require too much expertise. Therefore, CAD-free parameterizations are also popular, especially in the context of creative design, and this is the approach taken in Jetstream.

A wide variety of CAD-free geometry parameterizations have been used. Ideally, one would like to have flexibility in the number of design variables and the nature of the geometric freedom available to the optimizer. Moreover, it is desirable that the parametrization be convergent to an arbitrary geometry as the number of parameters is increased, a property of B-splines, for example. The coordinates of the grid nodes on the surface of the geometry are sometimes used as the geometry representation. This has the disadvantage that no analytical representation of the geometry exists, which can make it difficult to adapt or refine the grid, or to export the geometry. Another popular approach to geometry parameterization is the class function/shape function transformation method of Kulfan [5], which defines unique geometric shapes within several fundamental classes of geometry, such as airfoils, axisymmetric bodies, and axisymmetric nacelles. The bump functions originally introduced by Hicks and Henne [4] have also received widespread use.

In Jetstream the geometry is parameterized by B-spline surfaces defined by the coordinates of B-spline control points. Fig. 4.1 shows the B-spline surface control points for a parameterization of the CRM wing. The initial geometry can be imported by fitting it, usually represented by a surface mesh, to B-spline surfaces. This will lead to a slight modification of the geometry, but the change can be made arbitrarily small by increasing the number of B-spline control points. Alternatively the geometry can be created in analytical form compatible with a B-spline representation. The geometry generation package developed by Gagnon and Zingg enables the generation of a wide range of geometries that are defined analytically through watertight networks of topologically four-sided NURBS surfaces [2].

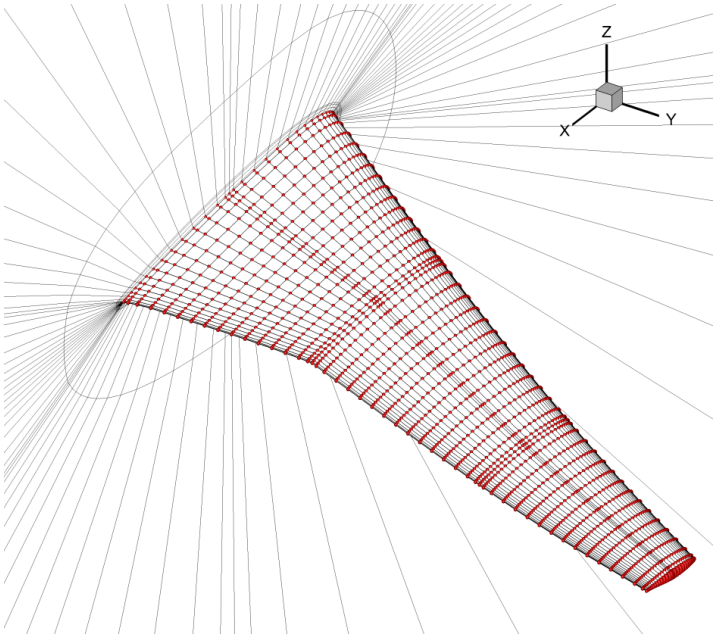
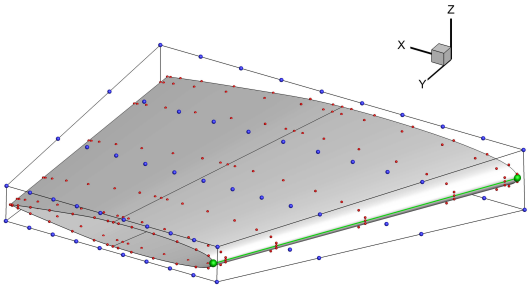
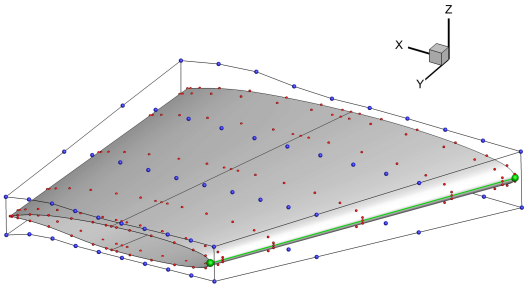


Fig. 4.1 Surface B-spline control points (in red) and control grid on the NASA CRM wing

One can use the parameters defining the geometry directly as design variables to control the geometry. If the surface grid nodes are used to control the geometry, this leads to a large number of design variables and typically necessitates smoothing of the gradient. Instead the surface mesh nodes can be controlled by a free-form deformation volume or via radial basis functions [7]. This reduces the number of design variables but retains the disadvantage that no analytical representation of the geometry exists. With a B-spline geometry parameterization, the B-spline surface control point coordinates can be used as design variables. This works well and remains available in Jetstream as an option. However, coupling the parameterization with the control has the following disadvantage. In order to generate or fit an initial geometry with B-spline surfaces, B-spline control points generally need to be clustered in regions of high curvature in order to achieve an accurate representation of the desired geometry, for example at a wing tip. However, when the B-spline control points are used as the design variables this can give the optimizer the ability to create shapes that may not be desired by the designer, for example because they cannot be manufactured. One might expect the optimizer to produce smooth shapes eventually, but during the early iterations highly complex geometries can lead to difficulties with mesh deformation and flow solution. Since the requirements of the parameterization, which are based on the need to generate or fit an initial geometry, are distinct from those of the control, which are related to the desired degree of geometric freedom, it is preferable to decouple control from parameterization.



(a) initial FFD volume



(b) modified FFD volume

Fig. 4.2 Initial and modified FFD volumes showing effect on wing geometry; FFD control points are shown in blue, while the B-spline control points parameterizing the geometry are shown in red

In order to address this, Jetstream uses a two-level approach where the B-spline control points that parameterize the surface are controlled via free-form and axial deformation. The B-spline control points are embedded in an FFD volume, which is a B-spline volume, as shown in Fig. 4.2(a). The FFD volume is controlled by its surface control points, the coordinates of which are used as design variables. Through their parametric coordinates, the B-spline surface control points parameterizing the geometry move when the FFD volume is modified; hence the geometry is also modified, as can be seen in Fig. 4.2(b).

In Jetstream the free-form deformation is combined with axial deformation [6] by adding an axial curve, as displayed in Fig. 4.3. The axial curve controls high-level deformations such as sweep, span, and dihedral changes, while the FFD controls the twist, taper, and sectional geometry. This approach reduces the number of constraints needed for the high-level deformations.

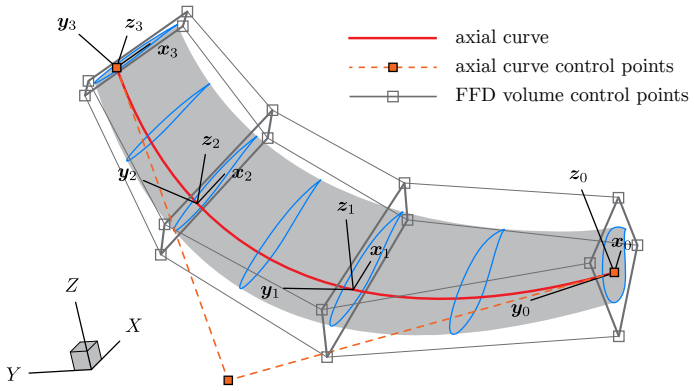


Fig. 4.3 FFD combined with axial curve for high-level geometry control

4.2 Mesh Deformation

The common practice in aerodynamic shape optimization is to deform an existing mesh rather than generating a new mesh for each new geometry. The goal of the mesh deformation is to maintain mesh quality comparable to that of the initial mesh. If only small geometric changes are of interest, then algebraic mesh deformation can be sufficient, but in general a more robust strategy is needed in order to maintain sufficient mesh quality. Moreover, with some large geometric changes, mesh deformation can be close to impossible, and a new mesh must be generated. This can arise, for example, with multi-element high-lift geometries or if a wing can move to a different location on a fuselage. For practical mesh deformation, current strategies can be divided into two types. One can either tightly couple the mesh deformation with the geometry parametrization, as in Jetstream, or keep the two separate such that the mesh deformation approach can be used with an arbitrary geometry parameterization. In the latter category, the use of radial basis functions to deform the mesh can be general, robust, efficient, and effective [1] [8].

In Jetstream's integrated approach, the B-spline surface geometry parameterization is leveraged by creating B-spline volumes to control the mesh. The control points of the B-spline volumes, which are distinct from the FFD volumes used for geometry control, form a control mesh that is much coarser than the actual mesh . The mesh nodes are associated with specific parametric coordinates in the B-spline volumes. When the control mesh is deformed, the nodal coordinates of the deformed mesh are thus readily found. An example of a control grid is shown in Fig. 4.1 with the corresponding grid seen in Fig. 4.4. The largest difference between the control grid and the actual grid is in the wall normal direction, where the control grid is much coarser.

Deformation of the control grid in response to a change in the surface control points is achieved by modeling the control grid as an elastic solid subject to the equations of linear elasticity. Nonlinearity is introduced by performing the defor-

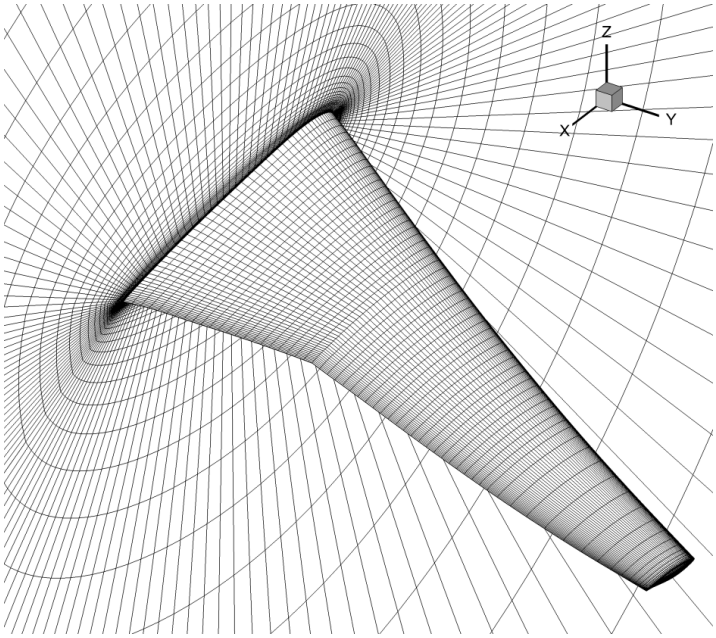


Fig. 4.4 Grid on the NASA CRM wing

mation in increments with the material properties changed at each increment, as described by Truong et al. [9]. The discretized equations of linear elasticity $M(C, X)$ must be solved for C at each increment. In Jetstream this is a linear system with $3n_c$ unknowns, where n_c is the number of B-spline volume control points, which is much smaller than the number of nodes in the mesh. The linear system is symmetric and can be solved using the preconditioned conjugate gradient method. In order to obtain a consistent design space, it is necessary to solve for the mesh deformation from the initial mesh as opposed to the mesh from the previous optimization iteration. For the same reason it is also important to have a consistent and relatively deep convergence criterion when solving the mesh deformation equation.

4.3 Numerical Solution of the Flow Equations

The flow evaluation is is the most computationally expensive component of an aerodynamic shape optimization methodology. Hence efficiency is important, and, given that during an optimization the flow may need to be evaluated for a wide range of possibly unusual shapes, the flow solver must also be extremely robust. For many aerodynamic shape optimization problems the appropriate governing equations are the steady compressible Reynolds-averaged Navier-Stokes (RANS) equations. Although the Euler equations governing compressible inviscid flows are sometimes

solved, they have some important limitations. With respect to drag minimization, the Euler equations can predict induced and wave drag but not viscous drag. Hence they can be useful in studying trade-offs associated with induced drag reduction. However, if shocks arise, the Euler equations can be inaccurate because the shock-boundary-layer interaction is not captured. Moreover, if the optimizer has the freedom to increase the surface area of the geometry, the Euler equations will not predict the associated increase in friction drag. Finally, even in induced drag studies where the Euler equations can be useful, only the relative induced drag values and optimized wing planforms are valid. The sectional geometries resulting from optimization based on the Euler equations will typically display high aft camber and excessive adverse pressure gradients because turbulent boundary-layer separation is not included in the model.

The grid density must be chosen to provide sufficient accuracy for the optimized geometry to remain optimal when analyzed using a finer mesh on which the solution is grid converged. For example, in an optimization where there is a trade off between friction and pressure drag, the grid resolution must be sufficient that both are accurately evaluated. It is always a good practice to perform a grid refinement study on a geometry that has been optimized on a grid with moderate resolution in order to ensure that the desired performance is achieved. For example, shock waves that have been eliminated during optimization can reappear when the resulting geometry is analyzed on a fine grid. This may mean that the grid used for the optimization was insufficiently refined.

The discrete flow residual $R(Q, C, X)$ is obtained by spatially discretizing the governing equations through a finite-difference, finite-volume, or finite-element method. Subsequently, the solution Q that satisfies $R(Q, C, X) = 0$ can be found by a variety of different approaches, as described by Witherden et al. [10]. Regardless of whether boundary conditions are applied explicitly or implicitly, the discrete flow Jacobian $\partial R/\partial Q$ must be based on a residual that includes all boundary conditions, and the vector Q must include the flow solution values at all grid nodes, including boundary nodes. The Jacobian is found by differentiating the residual with respect to the flow variables, a task that can be done by hand or through the use of automatic differentiation.

In Jetstream, spatial derivatives are approximated through summation-by-parts finite-difference operators applied on multi-block structured grids. Boundary and block interface conditions are enforced through simultaneous approximation terms. With this formulation, Q and R naturally include values at all nodes, including boundary nodes. Convergence to steady-state is achieved through a Jacobian-free inexact-Newton method with all equations, including the Spalart-Allmaras turbulence model equation, fully coupled. A pseudo-transient continuation method is used to bring the solution into the radius of convergence of the Newton method. The linear system arising at each Newton iteration is solved using a Krylov method, the generalized minimal residual method (GMRES), preconditioned through an approximate-Schur approach that enables excellent parallel scaling on thousands of processors. Although a Jacobian-free approach is used, an approximate Jacobian is formed based on nearest neighbor contributions and used in the formation of the

preconditioner. The residual norm is normal reduced by twelve orders of magnitude during optimization. Lesser convergence can lead to a noisy design space and prevent deep convergence.

4.4 Gradient Computation via the Adjoint Equations

Concentrating on the discrete adjoint method, the notation of Section 3.2 can be extended to include the additional partial differential equation constraint associated with the mesh deformation equation (4.1).[1] Using the Jetstream methodology as an example, the Lagrangian function defined in (3.26) becomes

$$L(Q, C, X, \psi_\mathrm{f}, \psi_\mathrm{m}) = J(Q, C, X) - \psi_\mathrm{f}^T R(Q, C, X) - \psi_\mathrm{m}^T M(C, X)\,, \tag{4.3}$$

where ψ_f and ψ_m represent flow adjoint variables and mesh adjoint variables, respectively. Continuing as in Section 3.2, the partial derivatives of L with respect to ψ_m, ψ_f, Q, C, and X, respectively, can be set to zero to obtain the first-order optimality conditions:

$$\begin{aligned}
\frac{\partial L}{\partial \psi_\mathrm{m}} &= 0 = M(C, X) && \text{the mesh deformation problem} \\
\frac{\partial L}{\partial \psi_\mathrm{f}} &= 0 = R(Q, C, X) && \text{the discrete flow problem} \\
\frac{\partial L}{\partial Q} &= 0 = \frac{\partial J}{\partial Q} - \psi_\mathrm{f}^T \frac{\partial R}{\partial Q} && \text{the flow adjoint problem} \\
\frac{\partial L}{\partial C} &= 0 = \frac{\partial J}{\partial C} - \psi_\mathrm{f}^T \frac{\partial R}{\partial C} - \psi_\mathrm{m}^T \frac{\partial M}{\partial C} && \text{the mesh adjoint problem} \\
\frac{\partial L}{\partial X} &= 0 = \frac{\partial J}{\partial X} - \psi_\mathrm{f}^T \frac{\partial R}{\partial X} - \psi_\mathrm{m}^T \frac{\partial M}{\partial X} && \text{the gradient .}
\end{aligned} \tag{4.4}$$

In the sequential approach these equations are solved in the order indicated. Given a set of design variables X that is an update from the previous set of design variables, the first step is to solve the mesh equation $M(C, X)$ for new values of the coordinates of the B-spline volume control points C, as described in Section 4.2. Given C, new grid coordinates can be determined from $G(C)$, and the flow equations $R(Q, C, X)$ can be solved for the flow field Q, as discussed in Section 4.3. Once Q and C are known, the flow adjoint equation can be solved for ψ_f:

$$\left(\frac{\partial R}{\partial Q}\right)^T \psi_\mathrm{f} = \left(\frac{\partial J}{\partial Q}\right)^T . \tag{4.5}$$

Solution of the flow adjoint equation requires the flow Jacobian $\frac{\partial R}{\partial Q}$. In Jetstream this is found primarily through hand linearization with some entries found using

[1] for simplicity we consider only one mesh deformation increment

the complex step method. Other approaches, such as automatic differentiation, are also possible. Equation 3.2 is a nonsymmetric linear system that can in principle be solved using preconditioned GMRES as in the flow algorithm. In order to obtain accurate gradients, however, the flow adjoint equation must be converged much further than is necessary for the linear system solution during the Newton iterations in the flow evaluation. Hence the use of GMRES can become quite memory intensive and a truncated strategy is preferable. In Jetstream, GCROT(m, k), a simplified and flexible version of GCROT, is used to solve the flow adjoint system [3]. The flow adjoint residual norm is normally reduced by at least eight orders of magnitude in order to achieve an accurate gradient.

Once the flow adjoint problem has been solved for ψ_{f}, the mesh adjoint problem can be solved for ψ_{m}. This is again a symmetric linear system that can be solved using the preconditioned conjugate gradient method. As in the case of the mesh deformation problem, this system has $3n_{\mathrm{c}}$ unknowns; hence the cost of its solution is much less than the solution of the flow adjoint problem. Applying the linear elasticity approach to the coarse control grid, rather than the grid itself, thus greatly reduces the computational cost of solving both the mesh deformation and mesh adjoint problems. If the mesh deformation is performed in increments, one mesh adjoint system must be solved for each increment.

After the mesh adjoint problem has been solved, the gradient can be computed and provided to a gradient-based optimization algorithm such as SNOPT. With the efficient mesh deformation strategy used in Jetstream, the primary computational costs are associated with the flow evaluation and the solution of the flow adjoint problem.

4.5 An Example of an Aerodynamic Shape Optimization Problem

We return now to the wing optimization example presented in Section 2.3. This is a single-point lift-constrained drag minimization at transonic speed with the freedom to modify the section shape and twist distribution. The wing planform is not free to change. This example is intended to illustrate the behaviour of the methodology and does not represent a practical design problem. The governing equations are the RANS equations with the Spalart-Allmaras turbulence model.

Figure 4.5 shows the convergence history for this problem solved using Jetstream.[2] The *optimality* reflects the gradient of the objective function and the constraints, excluding the PDE constraints associated with the flow and mesh equations, and hence is a good measure of the convergence of the optimization. Here it has been reduced by roughly three orders of magnitude in 280 iterations of the optimization algorithm SNOPT. The *feasibility* is a measure of how well the constraints are satisfied, again excluding the PDE constraints. Here the feasibility has been reduced to roughly

[2] Thanks to Howard Buckley for generating the results presented in this section.

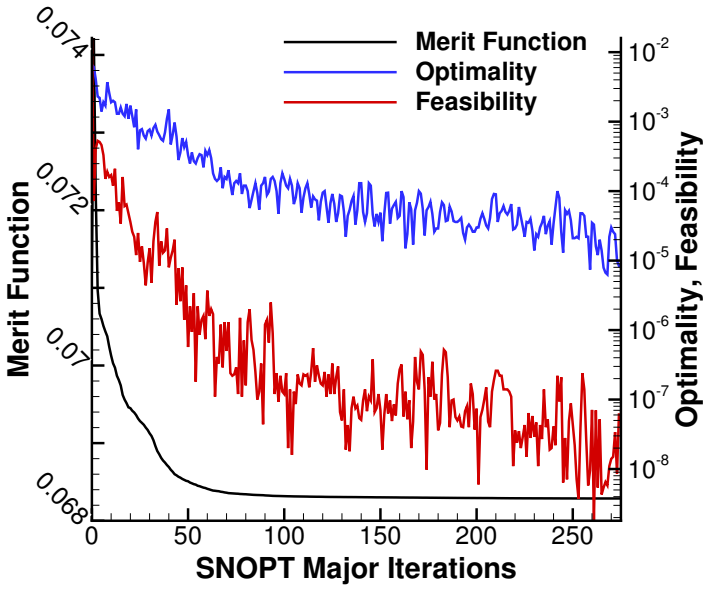


Fig. 4.5 Convergence of sample optimization problem

1×10^{-8}, indicating that the constraints are very well satisfied. The *merit function* includes the objective function plus the (non-PDE) constraint values; hence once the feasibility is small, the merit function effectively represents the objective function. Here the objective function is $C_D S$, where S is the nondimensional projected area of the wing, which is not free to vary during the optimization. Hence it effectively displays the reduction in the drag coefficient as the optimization progresses. It is clear that the majority of the drag reduction occurs during the first 50 optimization iterations, and virtually no improvement is seen beyond 100 iterations. Correlating this to the reduction in the optimality, we see that most of the reduction in the objective function is achieved by the time the optimality is reduced by two orders of magnitude.

The initial wing has an angle of attack of 2.4 degrees, $C_L = 0.5$, $C_M = -0.1709$, and $C_D = 0.0218$, while the optimized wing has an angle of attack of 3.6 degrees, $C_L = 0.5000$, $C_M = -0.1700$, and $C_D = 0.0200$, with both solutions computed on the relatively coarse grid used for the optimization. The reduction in the drag coefficient is over 8%.

Figure 4.6 shows contours of pressure coefficient on the upper surface of the initial and optimized wings. The primary difference is that the pressure recovery is more gradual on the optimized wing, especially at the inboard stations. This is seen more clearly in Figs. 4.7 and 4.8, which display the section shapes and pressure distributions at various spanwise positions along the wing. The pressure distributions show that the shocks present on the initial wing have been eliminated

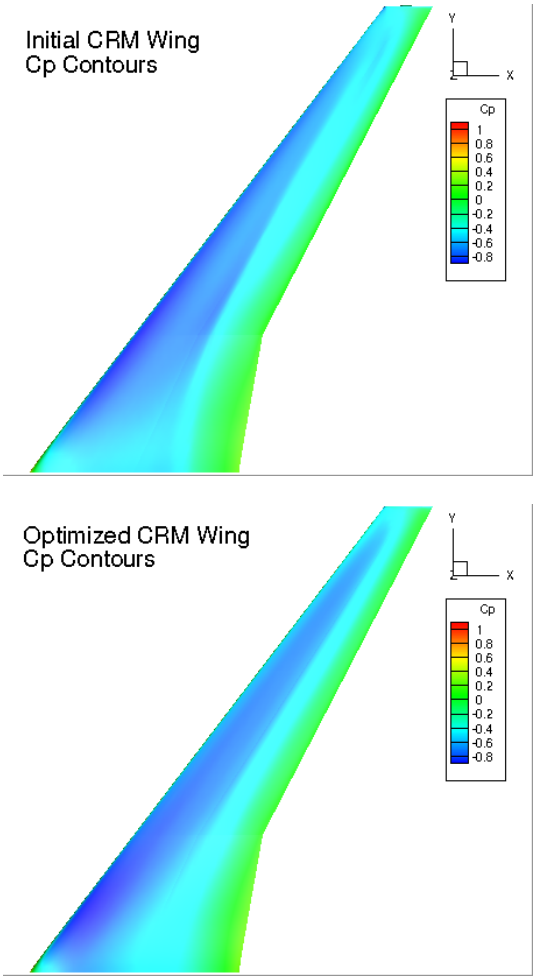
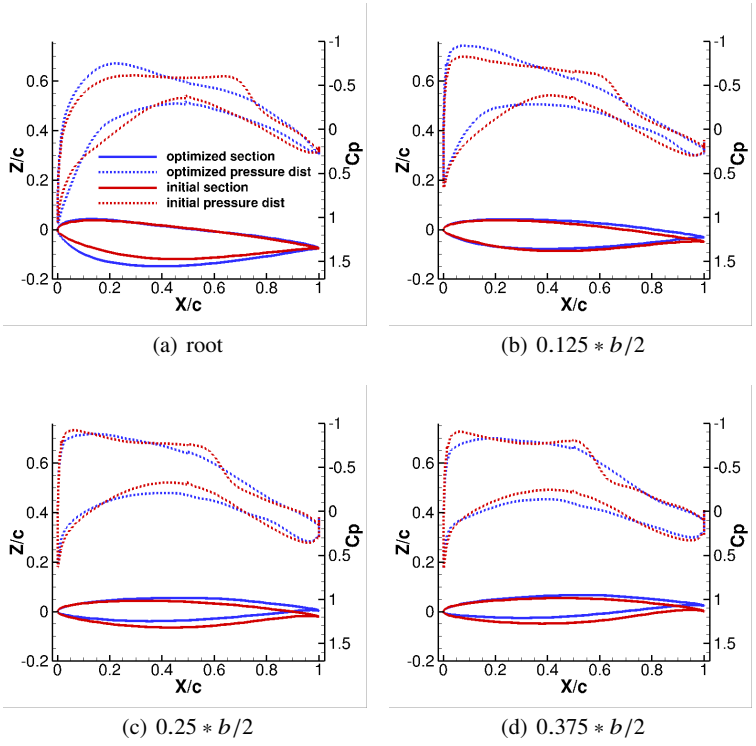


Fig. 4.6 Contours of pressure coefficient on upper surface of the initial and optimized wings

on the optimized wing. The wing has become thicker at the root and thinner toward the tip. Of course, this may not be practical from an aerostructural perspective; this is one reason that this is only a demonstration problem as opposed to a practical wing design.

Given that fully turbulent flow has been assumed and the surface area is highly constrained, the optimizer has little ability to reduce the friction drag in this case, and the drag reduction is primarily associated with wave and induced drag. Consequently, one might assume that an optimization based on the Euler equations would be sufficient for this optimization problem. The difficulty with optimization based on inviscid flow, however, is that the pressure recovery is not limited by separation of



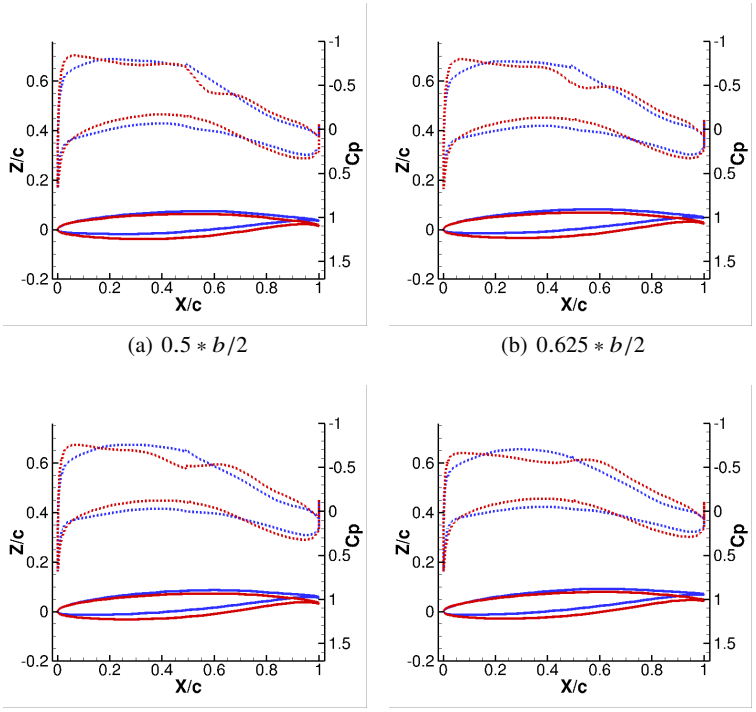
(a) root (b) $0.125 * b/2$

(c) $0.25 * b/2$ (d) $0.375 * b/2$

Fig. 4.7 Section shapes and pressure distributions for the initial and optimized wings ($b/2$ is the semispan of the wing)

the turbulent boundary layer. Therefore optimization based on the Euler equations can lead to excessive adverse pressure gradients and unrealistic aft loading.

The results presented are evaluated on the optimization mesh, which is quite coarse, with roughly one million nodes. After the optimization is completed, it is a good practice to re-evaluate the flow over the optimized geometry on a finer mesh. On the finer mesh, shocks that are not present when the flow is evaluated on the coarse mesh can reappear. Moreover, constraints that were satisfied based on the coarse mesh solution can be violated. Based on this information, the designer must decide whether the optimization must be repeated on a finer mesh. If so, the initial geometry for this second optimization can be the geometry optimized on the coarse mesh, so the number of optimization iterations needed to refine the geometry can be expected to be reasonably small.



(a) $0.5 * b/2$ (b) $0.625 * b/2$

(c) $0.75 * b/2$ (d) $0.875 * b/2$

Fig. 4.8 Section shapes and pressure distributions for the initial and optimized wings ($b/2$ is the semispan of the wing)

References

1. de Boer, A., van der Schoot, M., Bijl, H.: Mesh deformation based on radial basis function interpolation. Computers and Structures **85** (2007)
2. Gagnon, H., Zingg, D.: Generation of complex unconventional aircraft with application to high-fidelity aerodynamic shape optimization. Tech. Rep. 2013-2850, AIAA (2013)
3. Hicken, J., Zingg, D.: A simplified and flexible variant of GCROT for solving non symmetric linear systems. SIAM Journal on Scientific Computing **32** (2010)
4. Hicks, R., Henne, P.: Wing design by numerical optimization. Journal of Aircraft **15** (1978)
5. Kulfan, B.: Universal parametric geometry representation method. Journal of Aircraft **45** (2008)
6. Lazarus, F., Coquillart, S., Jancène, P.: Axial deformations: an intuitive deformation technique. Computer-aided Design **26** (1994)
7. Morris, A., Allen, C., Rendall, T.: CFD-based optimization of aerofoils using radial basis functions for domain element parameterization and mesh deformation. International Journal for Numerical Methods in Fluids **58** (2008)
8. Rendall, T., Allen, C.: Efficient mesh motion using radial basis functions with data reduction algorithms. Journal of Computational Physics **228** (2009)
9. Truong, A., Oldfield, C., Zingg, D.: Mesh movement for a discrete adjoint Newton-Krylov algorithm for aerodynamic optimization. AIAA Journal **46** (2008)

10. Witherden, F., Jameson, A., Zingg, D.: The design of steady state schemes for computational aerodynamics. In: Handbook of Numerical Methods for Hyperbolic Problems: Applied and Modern Issues, *Handbook of Numerical Analysis*, vol. 18, chap. 11. North Holland (2017)

Part II
An Aerodynamic Shape Optimization Methodology

Chapter 5
B-Spline Geometry Parameterization

5.1 B-Spline Curves

B-spline curves are of sufficient importance for geometry parameterization and control that we will discuss them in some detail. There are alternatives, but B-splines offer several nice properties that make them very attractive for use in geometry parameterization. The more general non-uniform rational B-splines, or NURBS, are commonly used in computer-aided design, but we focus here on B-splines as they are sufficient for our purposes.

The reader may be aware of cubic spline interpolation where a piecewise cubic curve is used to interpolate a function whose value is known at a finite set of points. In order to retain C^2 continuity at the breakpoints between distinct cubic segments, the problem must be solved globally. If one function value is altered, the interpolating function changes everywhere. This is not a desirable property for a shape parameterization, where we would like to be able to control the shape locally without altering the shape far away from the local change. Moreover, with a parametrization we do not need the ability to interpolate. We wish to find the shape that minimizes our objective function; in principle our initial shape does not need to be a specific shape. Relative to cubic spline interpolation, B-spline curves give up the ability to interpolate in return for being local in nature while maintaining a desired level of continuity. In practice, we may wish to begin with a specific shape, for example to see how much this shape can be improved upon or because portions of the shape are intended to remain unaltered with respect to a given shape. Fortunately, a B-splines curve can be found that lies arbitrarily close to a specified curve – we will return to this subject later.

A B-spline curve is defined as a weighted sum of basis functions $N_{i,k}$ as follows:

$$\boldsymbol{x}(t) = \sum_{i=1}^{n} \boldsymbol{X}_i N_{i,k}(t) \,, \quad t_{\min} \leq t \leq t_{\max} \,, \tag{5.1}$$

where n is the number of basis functions, $\boldsymbol{x}$ represents the coordinates of the curve as a function of the parameter t, $\boldsymbol{X}_i$ are the coordinates of the de Boor control points, and the $N_{i,k}(t)$ are polynomial basis functions of order k (degree $k-1$). The basis functions are defined with respect to a knot vector given by $\{t_1, t_2, \ldots, t_{k+n}\}$, where the t_i must be nondecreasing, the role of which will be explained shortly. With a given order of the basis functions and a specified knot vector, the B-spline curve is controlled by the coordinates of the control points $\boldsymbol{X}_i$.

The basis functions are defined recursively based on the following relations:

$$N_{i,1}(t) = \begin{cases} 1 \text{ if } \; t_i \le t < t_{i+1} \\ 0 \text{ otherwise} \end{cases} \tag{5.2}$$

$$N_{i,k}(t) = \left(\frac{t - t_i}{t_{i+p-1} - t_i} \right) N_{i,k-1}(t) + \left(\frac{t_{i+k} - t}{t_{i+p} - t_{i+1}} \right) N_{i+1,k-1}(t) \; . \tag{5.3}$$

For the remainder of this discussion we will concentrate on $k = 4$, i.e. cubic polynomials, which are a nice balance in terms of their continuity and their local support, as we will soon see. From (5.2) the piecewise constant basis functions span each of the knot intervals $t_i \le t \le t_{i+1}$ such that the number of such basis functions is equal to the number of knots minus one. Similarly, from (5.3) the piecewise linear functions will span two knot intervals, and the number of such basis functions is equal to the number of knots minus two. Continuing in this manner, we see that piecewise cubic basis functions span four knot intervals, and the number of piecewise cubic basis functions is equal to the number of knots minus four. Hence in order to have n piecewise cubic ($k = 4$) basis functions, the number of knots required is $n + 4$ (or $n + k$ in general).

The continuity of a B-spline curve at a knot is given by $k - m - 1$, where m is the *multiplicity* of the knot, i.e. the number of times the same value of t is repeated. Therefore C^2 continuity is achieved with a cubic B-spline at a knot of multiplicity one. In general, the B-spline curve does not pass through the control points; as discussed previously, it is not an interpolant. However, the curve can be forced to pass through a particular control point by repeating the knot; however, continuity is reduced. This is usually done in order to ensure that the endpoints of the curve are clamped. In the case of cubic B-splines, this requires that the first and last entries in the knot vector, $t_{\min}$ and $t_{\max}$ be repeated four times.

Consider the following knot vector: {0 0 0 0 0.25 0.5 0.75 1 1 1 1} spanning $0 \le t \le 1$ with both ends clamped and the remaining knots equally spaced. This knot vector has 11 entries, so with cubic polynomials this is compatible with 7 basis functions, which are shown in Figure 5.1. It can be seen that the first and last basis functions span only one knot interval (as a result of the repeated knots), the second and sixth span two, the third and fifth span three, and the fourth spans four. Any further increase in the number of knots would lead to further basis functions spanning four knot intervals. This illustrates the local support property of B-splines. A B-spline curve represented by these basis functions will be affected by the location of the first control point only in the interval $0 \le t \le 0.25$. The location of the fourth control point actually has a global influence in this simple example, as it spans

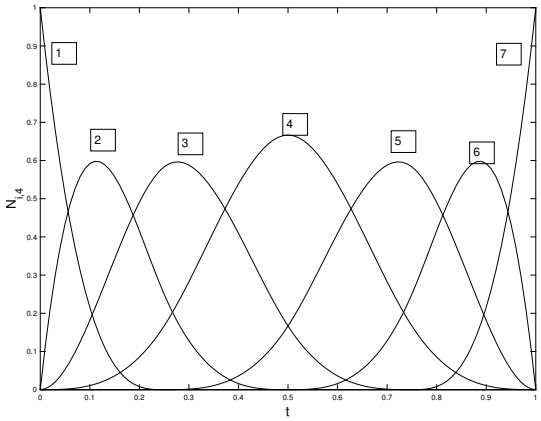


Fig. 5.1 Basis functions for example with clamped ends and uniform knot vector

$0 \leq t \leq 1$, but this arises only because there are only four knot intervals in this simple example. With cubic basis functions the maximum number of knot intervals spanned is four, so that as the number of knot intervals is increased the influence of any cubic basis function will remain capped at four knot intervals.

The effect of the repeated knot intervals at both ends can be seen in the nature of the basis functions at the end points. For example, at $t = 0$, the value of the first basis function $N_{1,4}$ is unity, while the value of all others is zero. Hence the coordinates of the B-spline curve at the first end point coincide exactly with the coordinates of the first control point, and the curve is clamped. The same occurs at the other end point. At any other point, the coordinate of the B-spline curve will result from a weighted sum of the different basis functions, and hence the curve will not pass through the control points. For example, at $t = 0.5$, $N_{3,4} = 1/6$, $N_{4,4} = 2/3$, and $N_{5,4} = 1/6$, while the other basis functions are equal to zero. Note that the basis functions sum to unity – this holds for all t. The coordinates of the B-spline at $t = 0.5$ are thus equal to 1/6 of the coordinates of the third control point plus 2/3 of the coordinates of the fourth control point plus 1/6 of the coordinates of the fifth control point. While the curve will be pulled toward the fourth control point, it will not pass through it.

Figure 5.2 shows a curve drawn using the seven basis functions shown in Figure 5.1 with the following coordinates for the seven control points:

$$(1,0),\ (0.5,-0.05),\ (0.25,-0.04),\ (0,0),\ (0.25,0.08),\ (0.5,0.10),\ (1,0)\,, \quad (5.4)$$

which produces quite a nice looking airfoil. The control points are also shown, as well as the convex hull they form. It is a property of B-splines that they lie inside such a convex hull (if the control points form a convex hull). The first and last points are clamped at $(1,0)$, ensuring that the airfoil trailing edge is closed and lies at this specific point. As a result we have a parameterized family of airfoils, where the parameters are the coordinates of the remaining five control points. One can fix the fourth control point at the origin and also fix the x coordinate of the remaining

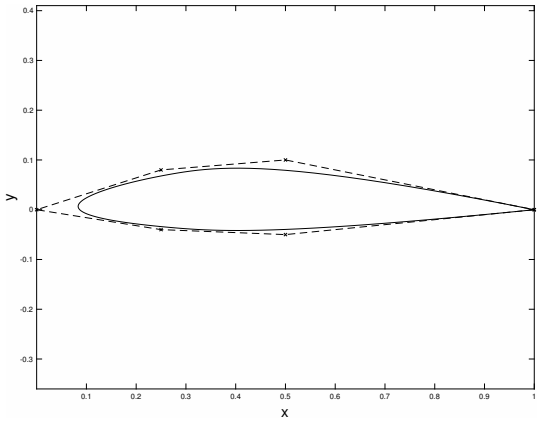


Fig. 5.2 A B-spline curve drawn using the basis functions shown in Figure 5.1 (solid line). The control points are shown (x) and connected by the dashed lines.

four control points to obtain a family of airfoils based on only four parameters, the y coordinates of the second, third, fifth, and sixth control points. This family does not provide sufficient geometric freedom to address many aerodynamic shape optimization problems, but it suffices for our purpose here, which is to show how a B-spline curve provides an excellent means of producing parameterized geometries in two dimensions. Before we explore this further, we will consider the role of the knot vector.

We have seen that having four repeated knot locations at the ends of a B-spline curve clamps the curve to the first and last control points. In our example above, the remaining knots are equally spaced. Next we examine the impact of nonuniformly spaced knots. For example, consider the following knot vector: {0 0 0 0 0.55 0.6 0.65 1 1 1 1}, where we now have two much smaller knot intervals near the middle of the parameter range. This nonuniform knot vector produces the basis functions displayed in Figure 5.3, showing that the fourth basis function has a much higher maximum value, while the second, third, fifth, and sixth basis functions have lower maximum values. Therefore, the curve will be much more strongly pulled toward the fourth control point and not as strongly toward the second, third, fifth, and sixth control points.

This is illustrated in Figure 5.4, which shows the curve produced using this nonuniform knot vector together with the same control point locations used with the uniform knot vector to produce the B-spline curve shown in Figure 5.2. One can see clearly the impact of the modified knot vector, which is consistent with the basis functions shown in Figure 5.3, where, relative to the original curve associated with the uniform knot vector, the modified curve lies much closer to the fourth control point and further from the second, third, fifth, and sixth control points. Hence the modified knot vector produces a parameterized family of airfoils that is distinct from that associated with the uniform knot vector. This example demonstrates how the knot vector affects the resulting B-spline curve. Although knot vector locations are

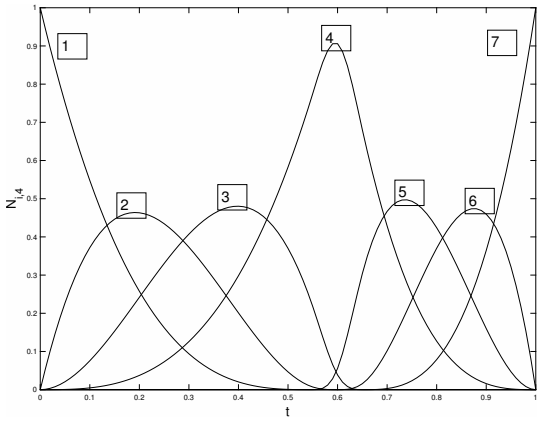


Fig. 5.3 Basis functions with nonuniform knot vector given by {0 0 0 0 0.55 0.6 0.65 1 1 1 1}.

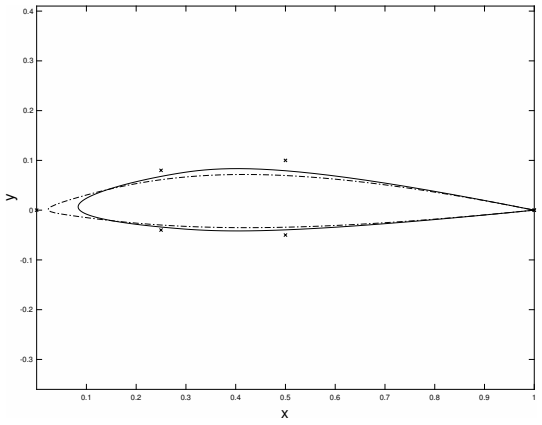


Fig. 5.4 A B-spline curve drawn using the modified knot vector given in (5.1) (dash-dot line) compared to the original curve shown in Figure 5.1 (solid line), along with the control points (x).

seldom used as design parameters, their judicious selection can impact the nature of the parameterized geometries. For example, clustering the knot vector in a given region gives more precise control in that region of the geometry. If three consecutive knots are assigned the same location, the curve will go through the associated control point, but continuity will be reduced to C^0. Furthermore, the knot locations come into play when fitting a B-spline curve to an existing geometry, a topic we will address later in this chapter.

Next consider a modification to the y coordinate of the second control point from -0.05 to +0.02, as shown in Figure 5.5. With this change, the control points no longer produce a convex hull. The resulting airfoil has substantially increased camber compared to the original airfoil. This simple example demonstrates how the parameters, for example the y coordinates of the second, third, fifth, and sixth control

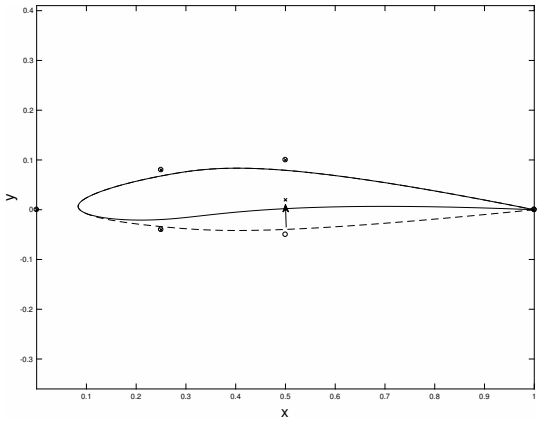


Fig. 5.5 Airfoil resulting from change in y coordinate of the second control point (solid line) compared to original airfoil (dashed line) both obtained using the uniform knot vector. Modified (x) and original (o) control points are shown.

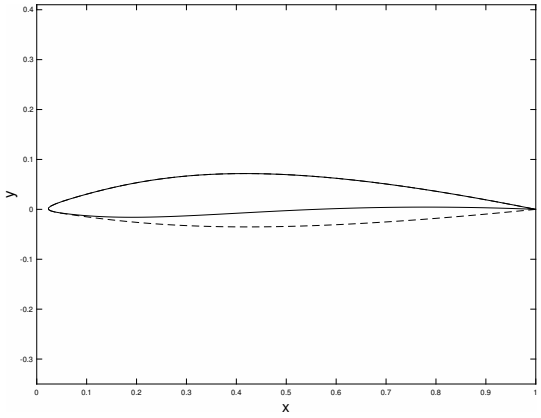


Fig. 5.6 Airfoil resulting from change in y coordinate of the second control point (solid line) compared to original airfoil (dashed line) both obtained using the nonuniform knot vector.

points, provide a parameterization from which a large and diverse family of airfoils with varying thickness and camber distributions can be obtained. This example also demonstrates the local support property of B-spline curves. The second basis function $N_{2,4}$ shown in Figure 5.1, which is the only basis function whose contribution is affected when the second control point is moved, is zero for $t \geq 0.6$. Hence the upper surface of the airfoil is unaltered as a result of moving the second control point. Finally, Figure 5.6 shows the airfoil that results from the same control point movement but with the nonuniform knot distribution discussed above, reinforcing the idea that the family of airfoils obtained can be influenced by the nature of the knot vector.

Discuss fitting next

Chapter 6
Free-Form and Axial Deformation for Geometry Control

Chapter 7
Mesh Deformation Based on the Equations of Linear Elasticity

Chapter 8
An Efficient Approach to Mesh Deformation

Chapter 9
Newton-Krylov-Schur Parallel Implicit Flow Solver for the Reynolds-Averaged Navier-Stokes Equations

Chapter 10
Implementation of the Discrete Adjoint Method

Chapter 11
The Complete Methodology

Part III
Case Studies

Chapter 12
Some Simple Examples

Chapter 13
Aspects of Problem Formulation and Influence of Various Parameters

Chapter 14
Aerodynamic Shape Optimization of an Aircraft Configuration