

Article

On an Exact Step Length in Gradient-Based Aerodynamic Shape Optimization

Farzad Mohebbi *  and Ben Evans 

Zienkiewicz Centre for Computational Engineering, College of Engineering, Swansea University, Bay Campus, Fabian Way, Crymlyn Burrows, Swansea SA1 8EN, UK

* Correspondence: farzad.mohebbi@swansea.ac.uk

Received: 31 March 2020; Accepted: 9 May 2020; Published: 13 May 2020



Abstract: This study proposed a novel exact expression for step length (size) in gradient-based aerodynamic shape optimization for an airfoil in steady inviscid transonic flows. The airfoil surfaces were parameterized using Bezier curves. The Bezier curve control points were considered as design variables and the finite-difference method was used to compute the gradient of the objective function (drag-to-lift ratio) with respect to the design variables. An exact explicit expression was derived for the step length in gradient-based shape optimization problems. It was shown that the derived step length was independent of the method used for calculating the gradient (adjoint method, finite-difference method, etc.). The obtained results reveal the accuracy of the derived step length.

Keywords: step length; Bezier curve; aerodynamic shape optimization; transonic flows

1. Introduction

The combination of computational fluid dynamics with numerical optimization algorithms has become a numerical tool used to predict the optimal shape for aerodynamic configurations, such as airfoils. During the last few decades, aerodynamic shape optimization has been an important area of research and great advances have been made due to the rapid progress in the development of high-performance computers and computational algorithms. The traditional “trial and error” approach used to design aerodynamic configurations is based on wind-tunnel testing. Nowadays, the high cost associated with wind-tunnel testing of real aerodynamic models is significantly reduced using numerical shape optimization. Instead of costly experiments of a whole range of aerodynamic configurations, only the optimal shape predicted by the computational analysis is tested in the wind tunnel. When validated this way, the optimal design may be used in the development of the final aerodynamic configuration. To make this approach viable, shape optimization algorithms for aerodynamic configurations must be as efficient as possible.

Among the factors involved in aerodynamic shape optimization are the type of flow regime (subsonic, transonic, supersonic, hypersonic), the governing equations (Laplace, Euler, Navier–Stokes), the grid used to mesh the flow domain (structured, unstructured, if structured: O-type, C-type, H-type, or hybrid), the design variables and the angle of attack, the method of discretization of the governing equations (finite element method (FEM), finite difference method (FDM), finite volume method (FVM)), the optimization method (Newton and quasi-Newton methods, conjugate gradient, genetic algorithm, etc.), the shape sensitivity analysis (finite-difference method, adjoint method, automatic differentiation, etc.), the aerodynamic body to be optimized (airfoil, wing, fuselage, blades, etc.), the objective function definition, the body parameterization method, and the imposed constraints (geometrical, etc.).

In Hicks and coworkers [1,2], computational fluid dynamics was combined with numerical optimization to predict the optimal shape design for airfoils and wings. In Hicks et al. [1], the drag on

nonlifting symmetric transonic airfoils in an inviscid flow is minimized under geometric constraints. The numerical optimization method is based on feasible directions and the inviscid flow is governed by the transonic, small-disturbance equation. In Hicks et al. [2], the optimal shape design of a wing is studied. The problems of drag minimization and lift-to-drag ratio maximization are addressed by solving the three-dimensional full potential equation coupled with the conjugate gradient method. The first use of adjoint equations for optimal design problems in fluid dynamics is presented in Pironneau [3] and for aerodynamic shape optimization problems governed by the full potential and the Euler equations are presented in Jameson and coworkers [4–7]. The application of the adjoint method to aerodynamic shape optimization has been the subject of intensive research during the past few decades [8–12]. The optimal shape design in fluid mechanics and aerodynamics using different methods is thoroughly studied in Mohammadi and Pironneau [13] and an excellent study on the coupling of computational fluid dynamics and optimization methods is given in Thevenin and Janiga [14]. A thorough examination of various aerodynamic optimization methods is given in Skinner and Zare-Behtash [15]. An extensive study on the mathematical aspects of the discrete adjoint method is presented in Giles and coworkers [16,17]. In Elliott and Peraire [18], the discrete adjoint method is used to calculate the gradient of the objective function with respect to design variables in airfoil and wing inverse design problems governed by the Euler equations using unstructured meshes. It is shown that the computation of the gradient of the objective function using the adjoint method is independent of the number of design variables. In Nemec and Zingg [19], a Newton–Krylov algorithm is applied to the discrete adjoint and the discrete flow-sensitivity approaches to calculate the gradient of the objective function in some two-dimensional aerodynamic shape optimization problems governed by the Navier–Stokes equations. The design problems considered include an inverse design, maximization of the lift-to-drag ratio, lift-constrained drag minimization, and lift enhancement. In Anderson and Venkatakrishnan [20], the continuous adjoint method is formulated and applied for aerodynamic shape optimization problems governed by the Euler and Navier–Stokes equations using unstructured grids for different objective functions, such as drag minimization, lift maximization, and an inverse design (based on a pressure distribution).

By increasing the number of design variables in large-scale aerodynamic shape optimization problems, the most efficient method for calculating the involved gradients is to combine gradient-based optimization methods, such as steepest-descent, conjugate-gradient, or quasi-Newton method, with the adjoint method. In the gradient-based optimization methods, the iterative process used to update the solution is based on the calculation of two variables: (1) the direction of descent and (2) the step length (a positive scalar). The former, which may be obtained from the calculation of the gradient of the underlying objective function, specifies the direction along which the value of the objective function is reduced, while the latter determines the distance that the updated solution should move in the direction of descent to substantially minimize the objective function. The success of a gradient-based optimization method depends on effective choices of both the direction of descent and the step length [21]. In aerodynamic shape optimization problems, the gradient of the objective function may be determined using the finite-difference method or the adjoint method. The step length is calculated using line search or trust-region methods. If the objective function to be minimized can be expressed as a polynomial function, one can find an exact step length; otherwise, an inexact step length may be obtained using the line search or trust-region methods, and it is chosen in such a way that some conditions, such as the Wolfe conditions, are satisfied [21]. To the best of our knowledge, in aerodynamic shape optimization problems, there exists no exact expression for the step length. In this study, an exact step length is developed for the first time for unconstrained minimization problems in aerodynamic shape optimization, such as drag and drag-to-lift ratio minimization problems.

2. Governing Equations

The steady inviscid transonic flows considered here are governed by the Euler equations. In this study, these equations were solved using the ANSYS Fluent v19.2 software (ANSYS, Inc., Canonsburg,

PA, USA) to obtain the airfoil surface pressure distribution needed to calculate the sensitivity coefficients using the finitedifference method.

Airfoil Parameterization

In this study, Bezier curves (BezierAirfoil: A MATLAB code for generating an airfoil shape using Bezier curve-V2. DOI: 10.13140/RG.2.2.30002.56000) were used to parameterize the airfoil surfaces.

Bezier curve: The mathematical definition of a Bezier curve is as follows:

$$P(t) = \sum_{i=0}^n B_i J_{n,i}(t) \tag{1}$$

where

$$J_{n,i}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \tag{2}$$

is the Bernstein basis polynomial of degree n , $t \in [0, 1]$, and $P(t) = (x(t), y(t))$. $x(t)$ and $y(t)$ are the x - and y - coordinates of the predetermined data points on the airfoil surface. By convention: $0^0 \equiv 1$ and $0! \equiv 1$. The order of a Bezier curve is equal to $n + 1$ (the number of the control points).

The parametric Bezier curve of interest here was of order 11 (because it can successfully generate a large number of airfoil shapes with a high degree of accuracy) and is given as follows:

$$\begin{aligned} n = 10 \Rightarrow \text{number of control points} &= 11 \\ P(t) &= \sum_{i=0}^{10} B_i J_{10,i}(t) = B_0 J_{10,0}(t) + \dots + B_{10} J_{10,10}(t) \\ &= B_0 \frac{10!}{0!(10-0)!} t^0 (1-t)^{10-0} + \dots + B_{10} \frac{10!}{10!(10-10)!} t^{10} (1-t)^{10-10} \end{aligned} \tag{3}$$

In this study, the x -coordinates of airfoil nodes were considered constant during the optimization, $P(t) = y(t)$. Therefore, for the predetermined y -coordinate of the airfoil nodes, we could write:

$$\begin{aligned} y_{1,j}(t) &= B_0(1-t_{y_{1,j}})^{10} + B_1 10 t_{y_{1,j}} (1-t_{y_{1,j}})^9 + B_2 45 t_{y_{1,j}}^2 (1-t_{y_{1,j}})^8 + B_3 120 t_{y_{1,j}}^3 (1-t_{y_{1,j}})^7 \\ &+ B_4 210 t_{y_{1,j}}^4 (1-t_{y_{1,j}})^6 + B_5 252 t_{y_{1,j}}^5 (1-t_{y_{1,j}})^5 + B_6 210 t_{y_{1,j}}^6 (1-t_{y_{1,j}})^4 + B_7 120 t_{y_{1,j}}^7 (1-t_{y_{1,j}})^3 \\ &+ B_8 45 t_{y_{1,j}}^8 (1-t_{y_{1,j}})^2 + B_9 10 t_{y_{1,j}}^9 (1-t_{y_{1,j}})^1 + B_{10} t_{y_{1,j}}^{10} \end{aligned} \tag{4}$$

Therefore, the Bezier control points B_0, \dots, B_{10} could be obtained at the first iteration for the predetermined y -coordinate of the initial airfoil. Two Bezier curves were needed to construct an airfoil surface: one for the upper surface (11 Bezier control points) and one for the lower surface (11 Bezier control points). Hence, we needed $2 \times 11 = 22$ Bezier control points. $y_{1,j}$ are the y -coordinates of the airfoil nodes in which $j = 1, \dots, \frac{N+1}{2}$ (N is the number of nodes on the airfoil surface) for the upper surface, $j = \frac{N+1}{2}, \dots, N$ for the lower surface (Figure 1), and $t_{y_{1,j}}$ is the value of $t \in [0, 1]$ associated with the $y_{1,j}$ (Figure 2). The airfoil chord was divided into $\frac{N+1}{2} - 1$ intervals of equal length l .

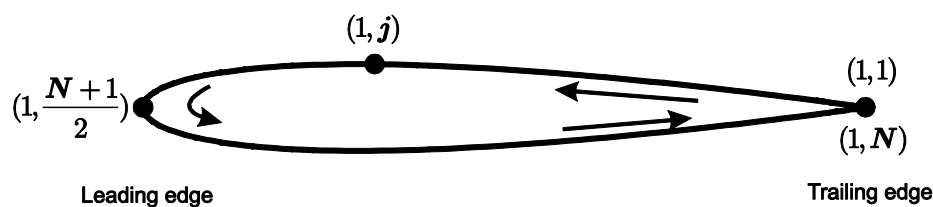


Figure 1. The node definition on the airfoil surface.

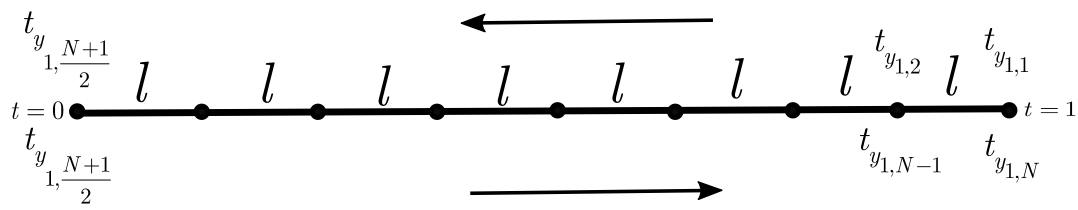


Figure 2. Definition of the parameter t used in the Bezier curve.

3. Aerodynamic Shape Optimization

3.1. Design Variables

Here, in the aerodynamic shape optimization, the design variables were the Bezier control points B_{DV_i} ($i = 1, \dots, 16$), as shown in Figure 3. The first and last two Bezier control points on the upper and the lower surfaces of the airfoil (hollow circles in Figure 3) were fixed and not considered to be design variables. For the ease of coding, the design variables were numbered consistent with the node numbering on the airfoil surface (see Appendix C). In other words, the design variables were numbered from right to left on the upper surface and from left to right on the lower surface (see Figure 3). The reason for fixing two control points adjacent to the airfoil trailing edge (B_9^U and B_9^L) was that the values of these two control points were very small ($B_9^U = -1.35283 \times 10^{-5}$ and $B_9^L = 1.35283 \times 10^{-5}$ for the NACA0012 airfoil, as shown in Figure 4), especially given that they should be multiplied by another small variable (finite-difference step-size: see Equation (17)), and even the double-precision calculations of the flow solver (here ANSYS Fluent) may not be able to detect the change of the pressure of the perturbed nodes. Thus, the computation of the sensitivity coefficients using the finite difference method for these two control points would be problematic and would result in a zero sensitivity coefficient due to the subtraction of nearly equal terms. The complex-step derivative approximation is a remedy for this issue if one has access to the solver source code to substitute all real type variable declarations with complex declarations and define all functions and operators that are not defined for complex arguments [22]. However, in this study, the two control points mentioned above were fixed and are not considered as design variables.

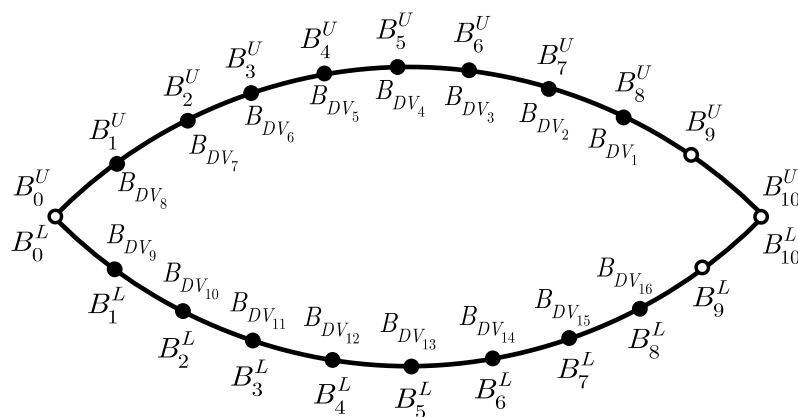


Figure 3. Definition of the design variables used for aerodynamic shape optimization. Here, the coordinates of the Bezier control points were not real and were plotted this way for the sake of simplicity in the definition.

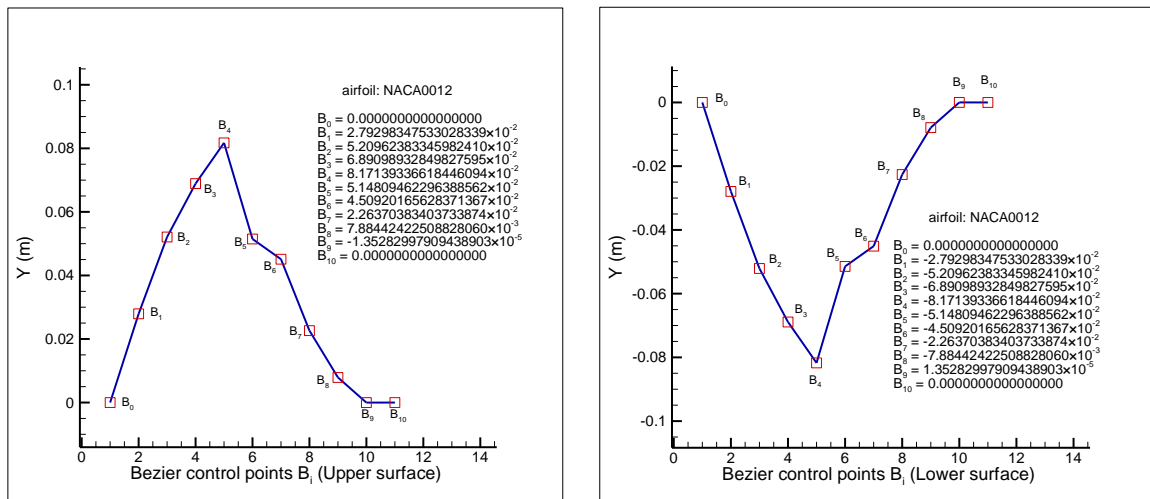


Figure 4. The values of the Bezier control points for the upper and lower surfaces of a NACA 0012 airfoil.

3.2. Objective Function

The aerodynamic shape optimization problem of interest here consisted of the minimization of the drag (D) to lift (L) ratio, $\frac{D}{L}$, at a fixed angle of attack α . This was an unconstrained optimization problem since no constraint was considered for it.

The objective function for this case is:

$$\mathcal{J} = \left(\frac{D}{L}\right)^2 \tag{5}$$

where

$$D = A \cos \alpha + N \sin \alpha \tag{6}$$

$$L = -A \sin \alpha + N \cos \alpha \tag{7}$$

and A and N are axial and normal forces, respectively (Figure 5) [23]. We can write the drag and lift forces as follows:

$$D = \sum_{j=1}^{N-1} (p_{1,j}(y_{1,j} - y_{1,j+1}) \cos \alpha + p_{1,j}(x_{1,j+1} - x_{1,j}) \sin \alpha) \tag{8}$$

$$L = \sum_{j=1}^{N-1} (-p_{1,j}(y_{1,j} - y_{1,j+1}) \sin \alpha + p_{1,j}(x_{1,j+1} - x_{1,j}) \cos \alpha) \tag{9}$$

Therefore, the objective function becomes:

$$\min \mathcal{J} = \left(\frac{D}{L}\right)^2 = \left(\frac{A \cos \alpha + N \sin \alpha}{-A \sin \alpha + N \cos \alpha}\right)^2 = \left(\frac{\sum_{j=1}^{N-1} (p_{1,j}(y_{1,j} - y_{1,j+1}) \cos \alpha + p_{1,j}(x_{1,j+1} - x_{1,j}) \sin \alpha)}{\sum_{j=1}^{N-1} (-p_{1,j}(y_{1,j} - y_{1,j+1}) \sin \alpha + p_{1,j}(x_{1,j+1} - x_{1,j}) \cos \alpha)}\right)^2 \tag{10}$$

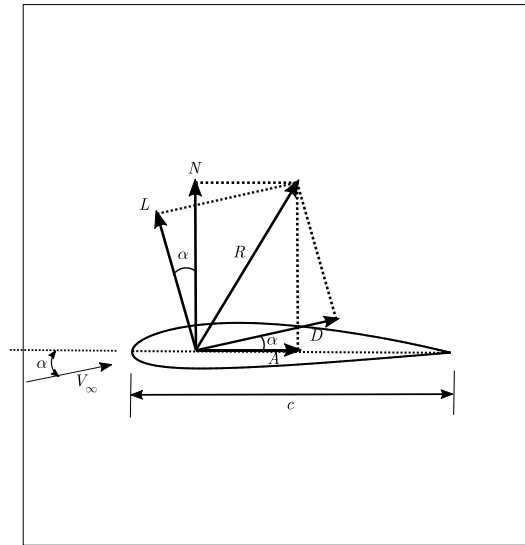


Figure 5. Resultant aerodynamic force R and the components into which it splits [23].

3.3. Sensitivity Analysis

The sensitivity analysis in gradient-based optimization is concerned with the computation of the derivative of the objective function with respect to the unknown variables. Suppose we wish to calculate the sensitivity of the objective function \mathcal{J} defined by Equation (10) to the Bezier control points B_{DV_i} ($i = 1, \dots, 16$). This can be mathematically expressed as:

$$\frac{\partial \mathcal{J}}{\partial B_{DV_i}} = 2\left(\frac{D}{L}\right) \frac{\partial}{\partial B_{DV_i}} \left(\frac{D}{L}\right) = 2\left(\frac{D}{L}\right) \frac{\frac{\partial D}{\partial B_{DV_i}} L - \frac{\partial L}{\partial B_{DV_i}} D}{L^2} = 2\left(\frac{D}{L^3}\right) \left(\frac{\partial D}{\partial B_{DV_i}} L - \frac{\partial L}{\partial B_{DV_i}} D\right) \quad (11)$$

The derivatives on the right-hand side of Equation (11) can be evaluated by taking the derivative of drag D , Equation (8), and lift L , Equation (9) with respect to the design variables B_{DV_i} , as follows:

$$\frac{\partial D}{\partial B_{DV_i}} = \sum_{j=1}^{N-1} \left[\left(\frac{\partial p_{1,j}}{\partial B_{DV_i}} (y_{1,j} - y_{1,j+1}) + p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \right) \cos \alpha + \left(\frac{\partial p_{1,j}}{\partial B_{DV_i}} (x_{1,j+1} - x_{1,j}) + p_{1,j} \underbrace{\frac{\partial (x_{1,j+1} - x_{1,j})}{\partial B_{DV_i}}}_{=0} \right) \sin \alpha \right] \quad (12)$$

Factoring $\frac{\partial p_{1,j}}{\partial B_{DV_i}}$, we obtain:

$$\frac{\partial D}{\partial B_{DV_i}} = \sum_{j=1}^{N-1} \left[\left(\frac{\partial p_{1,j}}{\partial B_{DV_i}} \left((y_{1,j} - y_{1,j+1}) \cos \alpha + (x_{1,j+1} - x_{1,j}) \sin \alpha \right) + p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \cos \alpha \right) \right] \quad (13)$$

In a similar fashion, we obtain:

$$\frac{\partial L}{\partial B_{DV_i}} = \sum_{j=1}^{N-1} \left[\left(-\frac{\partial p_{1,j}}{\partial B_{DV_i}} (y_{1,j} - y_{1,j+1}) - p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \right) \sin \alpha + \left(\frac{\partial p_{1,j}}{\partial B_{DV_i}} (x_{1,j+1} - x_{1,j}) + p_{1,j} \underbrace{\frac{\partial (x_{1,j+1} - x_{1,j})}{\partial B_{DV_i}}}_{=0} \right) \cos \alpha \right] \quad (14)$$

Factoring $\frac{\partial p_{1,j}}{\partial B_{DV_i}}$, we obtain:

$$\frac{\partial L}{\partial B_{DV_i}} = \sum_{j=1}^{N-1} \left[\left(\frac{\partial p_{1,j}}{\partial B_{DV_i}} \left(-(y_{1,j} - y_{1,j+1}) \sin \alpha + (x_{1,j+1} - x_{1,j}) \cos \alpha \right) - p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \sin \alpha \right) \right] \quad (15)$$

Thus, the derivative of the objective function, Equation (10) becomes:

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial B_{DV_i}} &= 2 \left(\frac{D}{L^3} \right) \left(\frac{\partial D}{\partial B_{DV_i}} L - \frac{\partial L}{\partial B_{DV_i}} D \right) = \\ &2 \left(\frac{D}{L^3} \right) \left[\sum_{j=1}^{N-1} \left[\left(\frac{\partial p_{1,j}}{\partial B_{DV_i}} \left((y_{1,j} - y_{1,j+1}) \cos \alpha + (x_{1,j+1} - x_{1,j}) \sin \alpha \right) + p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \cos \alpha \right) L - \right. \right. \\ &\left. \left. \sum_{j=1}^{N-1} \left[\left(\frac{\partial p_{1,j}}{\partial B_{DV_i}} \left(-(y_{1,j} - y_{1,j+1}) \sin \alpha + (x_{1,j+1} - x_{1,j}) \cos \alpha \right) - p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \sin \alpha \right) D \right] \right] = \quad (16) \\ &2 \left(\frac{D}{L^3} \right) \sum_{j=1}^{N-1} \left[\frac{\partial p_{1,j}}{\partial B_{DV_i}} (y_{1,j} - y_{1,j+1}) (L \cos \alpha + D \sin \alpha) + \frac{\partial p_{1,j}}{\partial B_{DV_i}} (x_{1,j+1} - x_{1,j}) (L \sin \alpha - D \cos \alpha) + \right. \\ &\left. p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} (L \cos \alpha + D \sin \alpha) \right] \end{aligned}$$

The sensitivity coefficients $\frac{\partial p_{1,j}}{\partial B_{DV_i}}$ may be computed using the finitedifference method, which is chosen for the simplicity of implementation. Suppose we wish to calculate the sensitivity of the pressure of nodes on the airfoil surface, $p_{1,j}$ ($j = 1, \dots, N - 1$), to the y -coordinate of the 16 Bezier control points (B_{DV_i} ($i = 1, \dots, 16$), 8 for each of the upper and lower surfaces) in this study. The sensitivity analysis can be performed by introducing a small perturbation to the y -coordinate of each of the Bezier control points. The grid generation (using perturbed values for the Bezier control points) and flow problem may be solved to obtain the new values for the pressure $p_{1,j}$. Using these values for the pressure, the dependency of the pressure $p_{1,j}$ on the perturbation of the y -position of Bezier control points can be evaluated. The central finitedifference method is used to compute these sensitivities as follows:

$$\frac{\partial p_{1,j}}{\partial B_{DV_i}} = \frac{p_{1,j}(B_{DV_i} + \varepsilon B_{DV_i}) - p_{1,j}(B_{DV_i} - \varepsilon B_{DV_i})}{2\varepsilon B_{DV_i}} \quad (17)$$

In this study, the finitedifference step-size was $\varepsilon = 0.005$. The term εB_{DV_i} is the perturbation in the y -position of the Bezier control point B_{DV_i} . Since the sensitivity of each pressure $p_{1,j}$ ($j = 1, \dots, N - 1$) to each y -position of the Bezier control point B_{DV_i} ($i = 1, \dots, 16$) is required, the computation of the sensitivity coefficients using this method requires $2 \times (\text{total number of control points} - \text{total number of fixed control points}) = 2 \times [2(11) - 2(3)] = 32$ additional solutions of the flow problem at each iteration.

Moreover, the terms $\frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} = \frac{\partial y_{1,j}}{\partial B_{DV_i}} - \frac{\partial y_{1,j+1}}{\partial B_{DV_i}}$ can be calculated using Equation (4). By having the value of the pressure at each node from the solution of flow equations (here, the Euler equations are used), the drag D and the lift L can be calculated using Equations (8) and (9), respectively. The angle of attack α is given a priori and is constant. The airfoil surface coordinates, $x_{1,j}$ and $y_{1,j}$,

are also known from the Bezier curve equation (here, the $x_{1,j}$ are constant and do not change in the optimization process).

4. Optimization Method

In this study, the unconstrained nonlinear minimization problem was solved using Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (a quasi-Newton method). The quasi-Newton method is a gradient-based optimization method that uses an approximation of the Hessian matrix or the inverse of a Hessian matrix [24]. In the BFGS method, an approximate inverse of the Hessian matrix, $\mathcal{B}^{(k)}$, is updated iteratively until an optimal solution is found. The steps of the aerodynamic shape optimization using the BFGS method as implemented in this study are as follows.

- (1) Specify the physical domain; the boundary conditions; the problem conditions, such as the free stream Mach number; and the angle of attack.
- (2) Generate structured or unstructured grids over the domain. The airfoil surface is generated using a Bezier curve of degree n .
- (3) Solve the flow problem regarding finding the pressure values at the grid points of the physical domain and hence the airfoil surface.
- (4) Using Equations (8) and (9), compute the objective function $\mathcal{J}^{(k)}$ Equation (10).
- (5) Compute the sensitivity coefficients using Equation (17).
- (6) Compute the gradient direction $\nabla \mathcal{J}^{(k)}$ using Equation (16).
- (7) The initial approximate inverse of Hessian matrix, $\mathcal{B}^{(1)}$, is taken as the identity matrix, namely $\mathcal{B}^{(1)} = \mathbf{I}$.
- (8) Compute the direction of descent $\mathbf{S}^{(k)}$ using the following equation:

$$\mathbf{S}^{(k)} = -\mathcal{B}^{(k)} \nabla \mathcal{J}^{(k)} \tag{18}$$

- (9) Compute the search step length $\beta^{(k)}$. An explicit expression for computing the search step length will be presented later.
- (10) Evaluate the new values of the design variables (Bezier curve control points) using the following equation:

$$\mathbf{B}_{DV}^{(k+1)} = \mathbf{B}_{DV}^{(k)} + \beta^{(k)} \mathbf{S}^{(k)} = \mathbf{B}_{DV}^{(k)} - \beta^{(k)} \mathcal{B}^{(k)} \nabla \mathcal{J}^{(k)} \tag{19}$$

- (11) Set the next iteration ($k = k + 1$) and return to step 2. Update the approximate inverse of the Hessian matrix as [24]:

$$\mathcal{B}^{(k+1)} = \mathcal{B}^{(k)} + \left(1 + \frac{(\mathbf{g}^{(k)})^T \mathcal{B}^{(k)} \mathbf{g}^{(k)}}{(\mathbf{d}^{(k)})^T \mathbf{g}^{(k)}}\right) \frac{\mathbf{d}^{(k)} (\mathbf{d}^{(k)})^T}{(\mathbf{d}^{(k)})^T \mathbf{g}^{(k)}} - \frac{\mathbf{d}^{(k)} (\mathbf{g}^{(k)})^T \mathcal{B}^{(k)}}{(\mathbf{d}^{(k)})^T \mathbf{g}^{(k)}} - \frac{\mathcal{B}^{(k)} \mathbf{g}^{(k)} (\mathbf{d}^{(k)})^T}{(\mathbf{d}^{(k)})^T \mathbf{g}^{(k)}} \tag{20}$$

where:

$$\mathbf{d}^{(k)} = \mathbf{B}_{DV}^{(k+1)} - \mathbf{B}_{DV}^{(k)} = \beta^{(k)} \mathbf{S}^{(k)} \tag{21}$$

$$\mathbf{g}^{(k)} = \nabla \mathcal{J}^{(k+1)} - \nabla \mathcal{J}^{(k)} \tag{22}$$

There exists no exact search step length for highly nonlinear aerodynamic shape optimization problems. Therefore, an inexact step length using line search methods, such as backtracking, is determined and it is chosen in such a way that some conditions, such as Wolfe conditions, are satisfied [13,21,25]. Here, we present a novel method to obtain an exact step length in aerodynamic shape optimization problems involving the unconstrained minimization of common objective function definitions, such as drag and the drag-to lift-ratio. The formulation given here is concerned with the Bezier curve only but it can be extended to other airfoil parameterization methods.

Exact Step Length

The exact step length can be obtained as:

$$\beta_{\text{exact}}^{(k)} = \underset{\beta \geq 0}{\operatorname{argmin}} \mathcal{J}(\mathbf{B}_{DV}^{(k+1)}) = \underset{\beta \geq 0}{\operatorname{argmin}} \mathcal{J}\left(\mathbf{B}_{DV}^{(k)} - \beta^{(k)} \mathcal{B}^{(k)} \nabla \mathcal{J}(\mathbf{B}_{DV}^{(k)})\right) \tag{23}$$

For simplicity, let $B_{DV_i}^{(k)} = a_i^{(k)}$ and $\mathcal{B}^{(k)} \nabla \mathcal{J}^{(k)} = b_i^{(k)}$ ($i = 1, \dots, 2n - 2$), and suppose, for generality, that the Bezier control points associated with the trailing and leading edges are considered fixed (they are not considered as design variables). Thus, we can write:

$$B_{DV_i}^{(k+1)} = a_i^{(k)} - \beta^{(k)} b_i^{(k)} \tag{24}$$

where $a_i^{(k)}$ is divided into $a_i^{U(k)}$ ($i = 1, \dots, n - 1$) for the upper surface and $a_i^{L(k)}$ ($i = n, \dots, 2n - 2$) for the lower surface. To obtain the exact step length $\beta_{\text{exact}}^{(k)}$, we minimize $\mathcal{J}(B_{DV_i}^{(k+1)}) = \left(\frac{D(B_{DV_i}^{(k+1)})}{L(B_{DV_i}^{(k+1)})}\right)^2$ with respect to $\beta^{(k)}$ as follows:

$$\frac{\partial \mathcal{J}}{\partial \beta} = 2 \frac{D}{L} \frac{\partial}{\partial \beta} \left(\frac{D}{L}\right) = 0 \Rightarrow D = 0 \tag{25}$$

$$D = \sum_{j=1}^{N-1} \left(y_{1,j} (p_{1,j} - p_{1,j-1}) \cos \alpha + N \sin \alpha \right) = \left(\sum_{j=2}^{\frac{N+1}{2}-1} y_{1,j} (p_{1,j} - p_{1,j-1}) + \sum_{j=\frac{N+1}{2}+1}^{N-1} y_{1,j} (p_{1,j} - p_{1,j-1}) \right) \cos \alpha + N \sin \alpha = 0 \tag{26}$$

Since $y_{1,1} = y_{1,N} = y_{1,\frac{N+1}{2}} = 0$, the indices in Equation (26) do not include $j = 1, \frac{N+1}{2}, N$. Equation (26) is the same as Equation (8) with the rearrangement of terms being with respect to $y_{1,j}$ rather than $p_{1,j}$. From the definition of the Bezier curve, Equations (1) and (2)), we can write the drag, Equation (26) as:

$$D = \left\{ \sum_{j=2}^{\frac{N+1}{2}-1} \left(\sum_{i=1}^{n-1} B_{DV_i}^{U(k+1)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) + \sum_{j=\frac{N+1}{2}+1}^{N-1} \left(\sum_{i=1}^{n-1} B_{DV_i}^{L(k+1)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) \right\} \cos \alpha + N \sin \alpha = 0 \tag{27}$$

Using $B_{DV_i}^{(k+1)} = a_i^{(k)} - \beta^{(k)} b_i^{(k)}$ (Equation (24)), we can write:

$$D = \left\{ \sum_{j=2}^{\frac{N+1}{2}-1} \left(\sum_{i=1}^{n-1} (a_i^{U(k)} - \beta^{(k)} b_i^{U(k)}) \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) + \sum_{j=\frac{N+1}{2}+1}^{N-1} \left(\sum_{i=1}^{n-1} (a_i^{L(k)} - \beta^{(k)} b_i^{L(k)}) \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) \right\} \cos \alpha + N \sin \alpha = 0 \tag{28}$$

After expanding, we obtain:

$$D = \left\{ \sum_{j=2}^{\frac{N+1}{2}-1} \left(\sum_{i=1}^{n-1} a_i^{U(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} - \beta^{(k)} \sum_{i=1}^{n-1} b_i^{U(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) + \sum_{j=\frac{N+1}{2}+1}^{N-1} \left(\sum_{i=1}^{n-1} a_i^{L(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} - \beta^{(k)} \sum_{i=1}^{n-1} b_i^{L(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1 - t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) \right\} \cos \alpha + N \sin \alpha = 0 \tag{29}$$

Using the identity $\sum_{i=k_0}^{k_1} \sum_{j=l_0}^{l_1} a_{ij} = \sum_{j=l_0}^{l_1} \sum_{i=k_0}^{k_1} a_{ij}$, we can write the above expression as:

$$D = \left\{ \sum_{i=1}^{n-1} \left(\sum_{j=2}^{\frac{N+1}{2}-1} a_i^{U^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} - \beta^{(k)} \sum_{j=2}^{\frac{N+1}{2}-1} b_i^{U^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) + \right. \tag{30}$$

$$\left. \sum_{i=1}^{n-1} \left(\sum_{j=\frac{N+1}{2}+1}^{N-1} a_i^{L^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} - \beta^{(k)} \sum_{j=\frac{N+1}{2}+1}^{N-1} b_i^{L^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} \right) (p_{1,j} - p_{1,j-1}) \right\} \cos \alpha + N \sin \alpha = 0$$

Therefore, we obtain

$$\beta_{\text{exact}}^{(k)} = \frac{\left[\sum_{i=1}^{n-1} \sum_{j=2}^{\frac{N+1}{2}-1} a_i^{U^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) + \sum_{i=1}^{n-1} \sum_{j=\frac{N+1}{2}+1}^{N-1} a_i^{L^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) \right] \cos \alpha + N \sin \alpha}{\left[\sum_{i=1}^{n-1} \sum_{j=2}^{\frac{N+1}{2}-1} b_i^{U^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) + \sum_{i=1}^{n-1} \sum_{j=\frac{N+1}{2}+1}^{N-1} b_i^{L^{(k)}} \frac{m!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) \right] \cos \alpha} \tag{31}$$

which is the exact step length. As can be seen from Equation (31), the value of the exact step length depends on the gradient of the objective function and it is independent of the method used for calculating the gradient. In other words, one can calculate the gradient with any available method, such as the adjoint method, the finitedifference method, etc., and then use Equation (31) to calculate the exact step length. It can be concluded that an inaccuracy in computing the gradient of the objective function or the approximate inverse of Hessian matrix $\mathcal{B}^{(k)}$ can give rise to an inaccuracy in calculating the exact step length.

For the sake of simplicity, the derivation of the above general step length expression is presented for the Bezier curve of degree 10 and is given in Appendix A and a Fortran code to calculate it is given in Appendix D. The derivation of the exact step length for other forms of the objective function, such as drag minimization and lift-constrained drag minimization, is given in Appendix B.

5. Results

As mentioned previously, the objective of the aerodynamic shape optimization problem considered in this study was to minimize the drag-to-lift ratio (which is equivalent to the maximization of the lift-to-drag ratio) for an airfoil moving in a steady inviscid transonic flow through the implementation of the exact step length in a gradient-based optimization algorithm. The following test case is presented to investigate the accuracy and performance of the derived step length.

Test Case: A NACA 0012 airfoil operating at a fixed angle of attack of 2° and a free-stream Mach number of 0.75 was considered for this test case as the initial shape and flow conditions. The free stream pressure was $p_\infty = 101,325.0$ Pa. There was a strong shock at half of the airfoil chord length on the upper surface. Since there was shock-induced pressure drag on the airfoil in these operating conditions, we aimed to optimize the airfoil to eliminate or weaken the shock as much as possible. The initial drag and lift forces were $D = 515.760$ N and $L = 17,090.0$ N, respectively (this means that for the initial airfoil, NACA 0012, in these flow conditions, $\frac{L}{D} = \frac{17,090.0 \text{ N}}{515.760 \text{ N}} = 33.1$). The initial drag and lift coefficients were $c_d = 1.294 \times 10^{-2}$ and $c_l = 0.42867$, respectively, which are in excellent agreement with $c_d = 1.25 \times 10^{-2}$ and $c_l = 0.4225$ in Carpentieri et al. [26]. As shown in Table 1, the drag and lift forces of the optimal shape were $D = 36.142$ N and $L = 15,086.0$ N, respectively (this means that for the optimal airfoil, $\frac{L}{D} = \frac{15,086.0 \text{ N}}{36.142 \text{ N}} = 417.4$, which was an approximately 11.6 times increase in the value of $(\frac{L}{D})$). A substantial decrease in the drag (from 515.760 N to 36.142 N) using only 16 design variables was achieved. Moreover, the drag and lift coefficients of the optimal shape were $c_d = 9.065 \times 10^{-4}$ and $c_l = 0.37839$, respectively. A structured O-grid (see Figure 6) with $N = 601$ nodes on the airfoil surface and 109 nodes in the normal direction to the airfoil surface was generated. The inviscid transonic flow equations (Euler equations) were solved using the ANSYS Fluent software to compute the pressure at each node on the airfoil surface (for unperturbed and $2 \times 16 = 32$ perturbed shapes for each iteration). The computed pressures were then substituted into Equation (17) to calculate the

sensitivity coefficients needed to compute the gradient of the objective function. A comparison of the initial shape (NACA 0012 airfoil) and the optimal one, as well as the corresponding surface pressure coefficients, is shown in Figure 7a,b, respectively. As can be seen, the shock on the initial shape was eliminated in the optimal shape. The static pressure distributions of the initial and optimal shapes are depicted in Figure 8a,b, respectively. A comparison of the initial and updated shape at each iteration is shown in Figure 9. As can be seen in Figure 9, the shock was eliminated or significantly weakened at each iteration (it is apparent from the value of the drag force given in Table 1, as well as from the thinner leading edge in the updated airfoils). The optimal shape is also in good agreement with the optimal shape in References [26,27]. A comparison of the initial shape (NACA 0012) and the optimal shapes obtained in Chen et al. [27] and from our study is shown in Figure 10. As can be seen, the upper surface shapes in both optimal shapes were shock-free, which resulted in the minimum drag in both shapes (as can be seen in Table 1, the majority of the reduction in the objective function $(\frac{D}{L})^2$ in the test case was due to the decrease in the drag value). The history of the objective function is shown in Figure 11. The value of the objective function was substantially reduced by 99.4% and most of the reduction was obtained in the first iteration. The optimization process was continued for 15 iterations. The value of the exact step length used in the BFGS optimization method (using Equation (31) or Equation (A5)) is plotted in Figure 12. At iteration 6, the negative value for the step length was obtained ($\beta^{(6)} = -1.2468$). The minimum value for the objective function was obtained at iteration 6; however, as this value was obtained by a negative step length, the optimization process was continued further. The sensitivity coefficients at some arbitrary nodal points (node numbers 20, 170, and 270 on the upper surface and node numbers 350, 500, and 580 on the lower surface; there are $\frac{N+1}{2} = \frac{601+1}{2} = 301$ nodes on each surface of the airfoil) on the airfoil surface using three different finite-difference step-sizes $\epsilon = 10^{-2}$, $\epsilon = 10^{-3}$, and $\epsilon = 5 \times 10^{-3}$ are depicted in Figure 13. As shown, the sensitivity coefficients at each node using the three step-sizes were in good agreement. However, in this study, the finite-difference step-size $\epsilon = 5 \times 10^{-3}$ was used to calculate the sensitivity coefficients using Equation (17). The gradients of the objective function $\mathcal{J} = (\frac{D}{L})^2$ with respect to the design variables B_{DV_i} ($i = 1, \dots, 16$) for the first and last iterations (iterations 1 and 15) are shown in Figure 14a,b, respectively, representing a significant decrease in the norm of the gradient (from 0.0697 to 4.151×10^{-4} ; a 99.4% reduction) to infer the optimal shape.

Table 1. Results for the test case.

Iteration Number (k)	$J = (\frac{D}{L})^2$	Step Length $\beta(m^2)$	D(N)	L(N)	c_d	c_l
0 (initial shape)	9.10775×10^{-4}	-	515.760	17,090.000	1.294×10^{-2}	0.42867
1	1.97841×10^{-5}	0.5534	34.032	7651.200	8.536×10^{-4}	0.19191
2	1.34569×10^{-5}	1.6872	35.644	9716.600	8.940×10^{-4}	0.24372
3	1.06817×10^{-5}	1.1510	38.239	11,700.000	9.591×10^{-4}	0.29348
4	9.92128×10^{-6}	2.4779	40.664	12,910.000	1.020×10^{-3}	0.32381
5	8.40409×10^{-6}	0.0188	40.980	14,136.000	1.028×10^{-3}	0.35456
6	4.89131×10^{-6}	-1.2468	35.521	16,061.000	8.910×10^{-4}	0.40284
7	1.50193×10^{-5}	47.6144	72.886	18,807.000	1.828×10^{-3}	0.47174
8	2.14748×10^{-5}	2.8478	49.214	10,620.000	1.234×10^{-3}	0.26638
9	1.66016×10^{-5}	0.3450	48.401	11,879.000	1.214×10^{-3}	0.29795
10	1.24568×10^{-5}	0.2160	45.653	12,935.000	1.145×10^{-3}	0.32445
11	7.33858×10^{-6}	0.2599	38.305	14,140.000	9.608×10^{-4}	0.35466
12	6.68139×10^{-6}	0.0009	40.202	15,553.000	1.008×10^{-3}	0.39011
13	8.65189×10^{-6}	11.8508	40.162	13,654.000	1.007×10^{-3}	0.34247
14	1.26244×10^{-5}	1.8642	59.411	16,721.000	1.490×10^{-3}	0.41941
15	5.73953×10^{-6}	0.9977	36.142	15,086.000	9.065×10^{-4}	0.37839
-	99.4% reduction	-	93.0% reduction	11.7% reduction	-	-

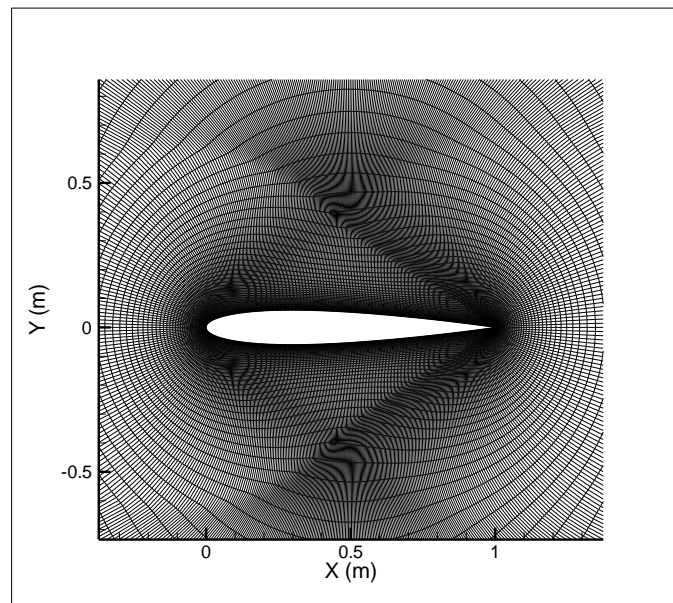


Figure 6. Structured mesh (hyperbolic O-grid) used for solving the transonic flow equations. The size of the mesh was 109×601 .

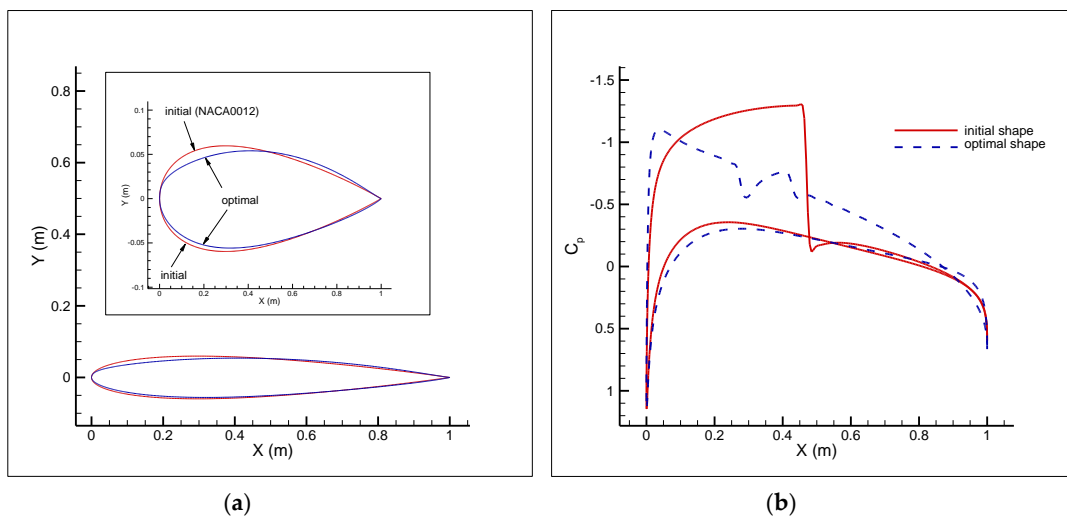


Figure 7. Comparison of initial and optimal airfoils used in the transonic flow shape optimization (a) and the corresponding airfoil surface pressure coefficients (b).

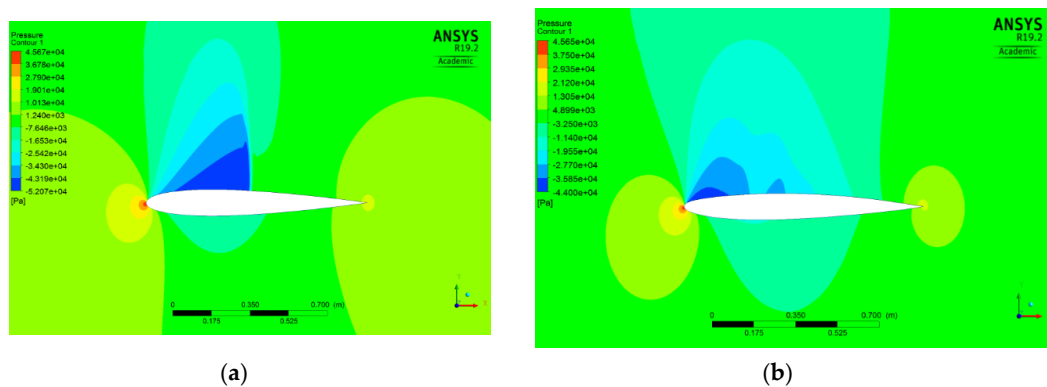


Figure 8. Static pressure distribution for the initial shape (NACA 0012) (a) and the optimal shape (at iteration 15) (b).

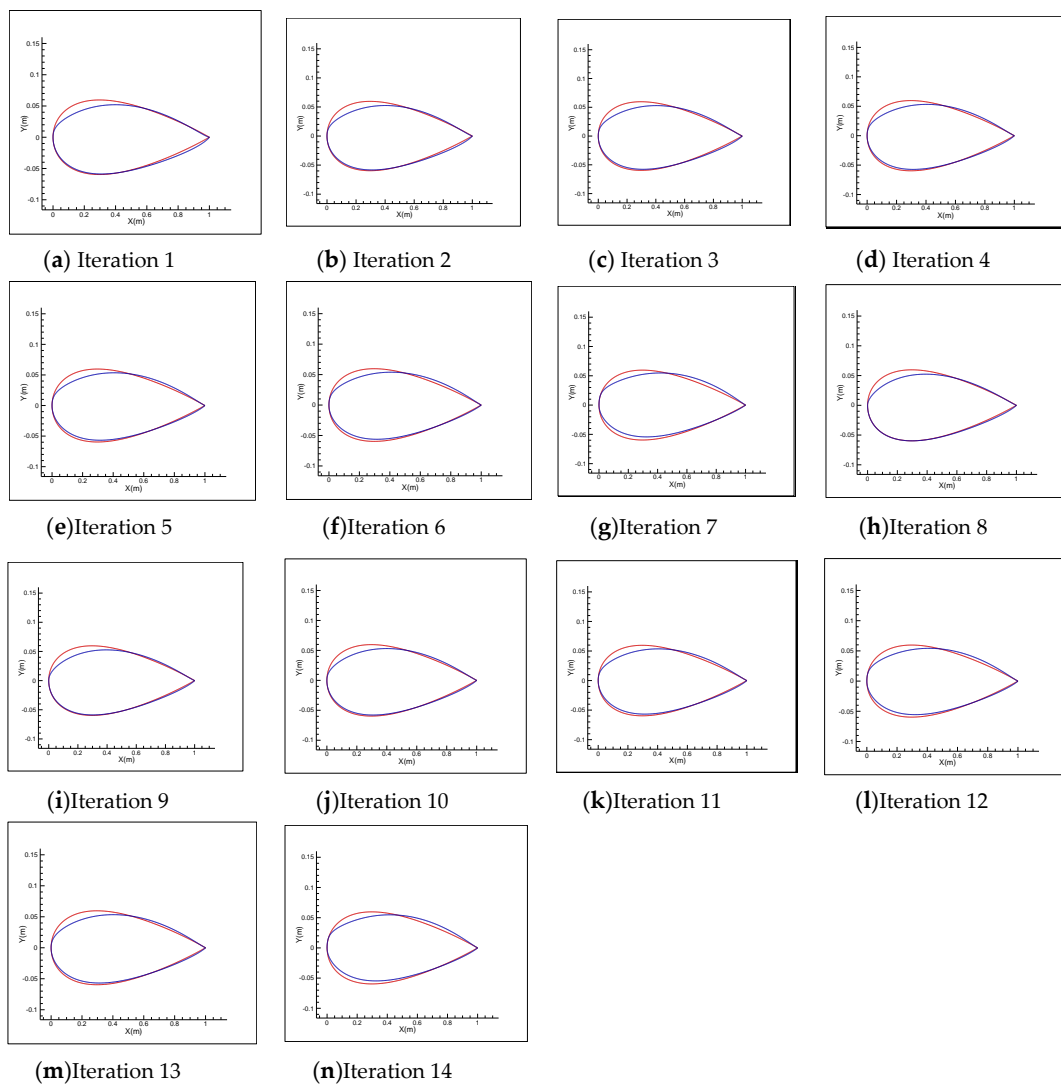


Figure 9. Comparison of the initial shape (NACA 0012) and the updated shape at (a–n) iteration 1 to iteration 14.

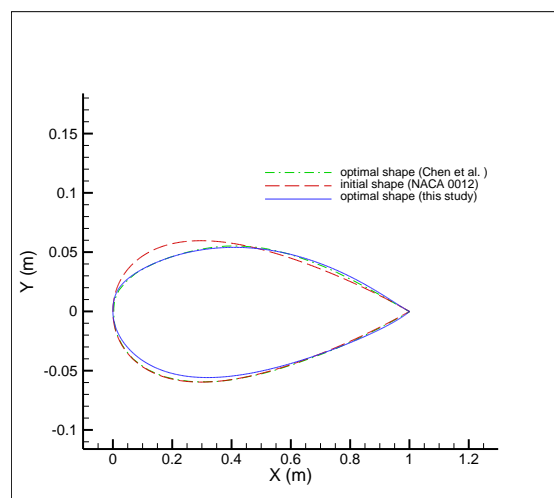


Figure 10. Comparison of the initial shape (NACA 0012), the optimal shape from our study, and the optimal shape from Chen et al. [27]; it shows a good agreement between the optimal shapes and an excellent agreement between the optimal upper surfaces where the shock was eliminated.

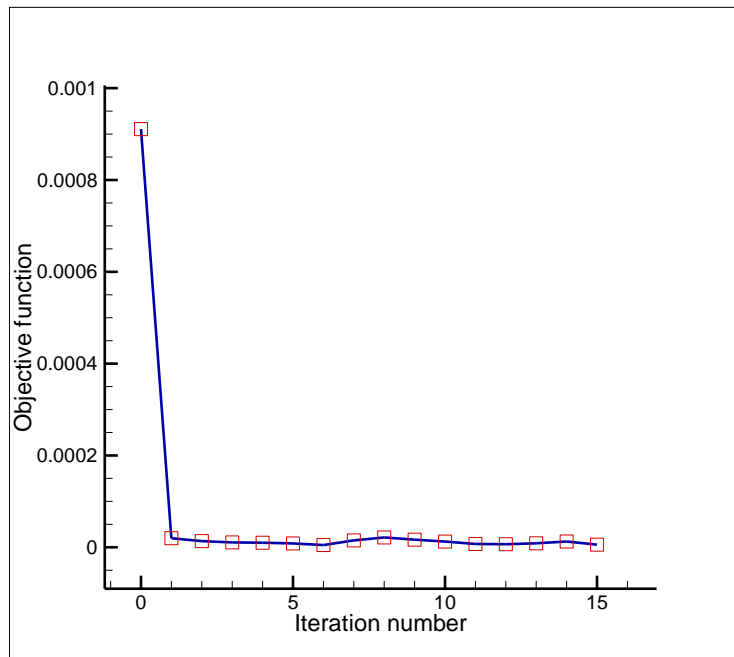


Figure 11. History of the objective function, $\mathcal{J} = (\frac{D}{L})^2$, during the shape optimization process.

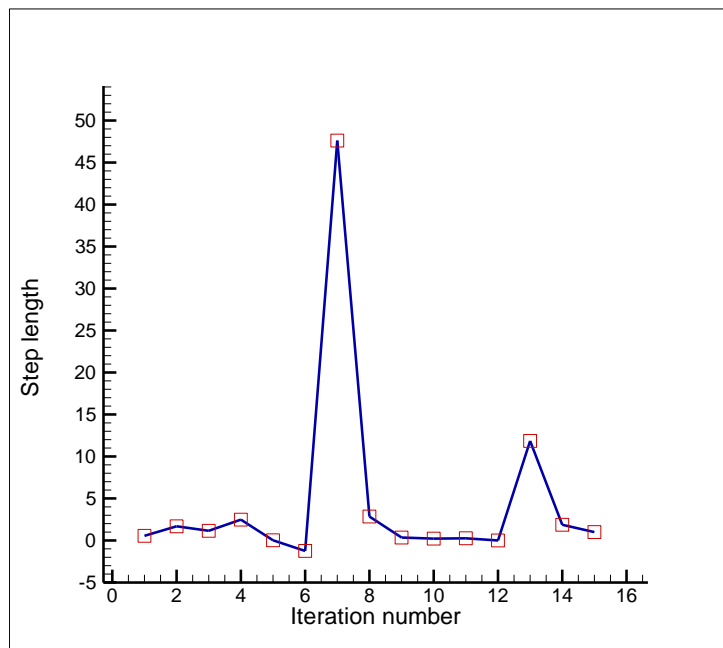
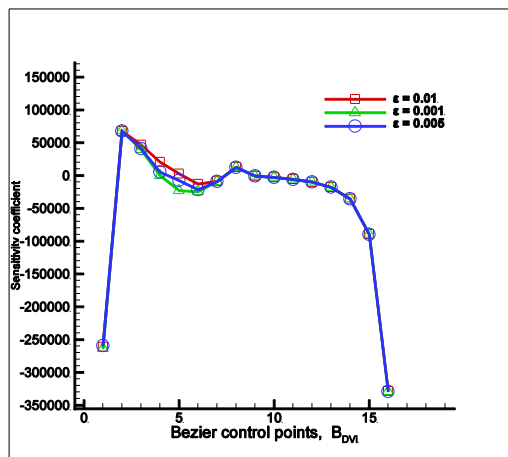
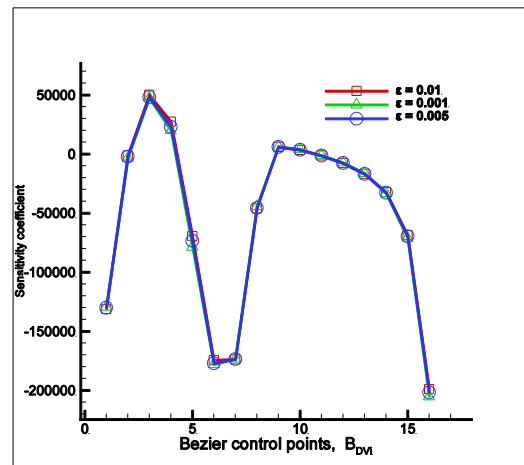


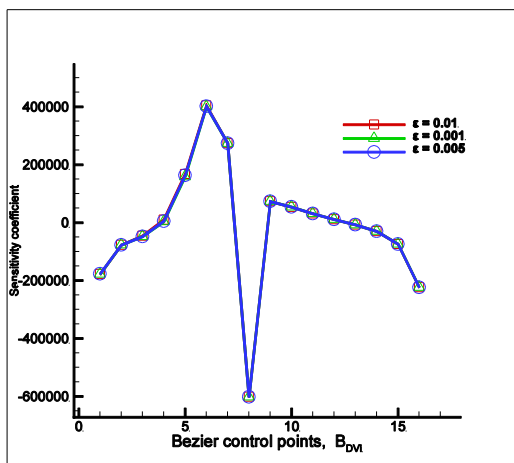
Figure 12. Exact step length value during the shape optimization process.



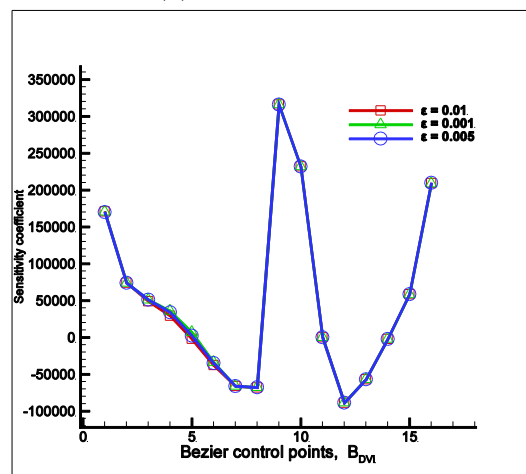
(a) Node number 20



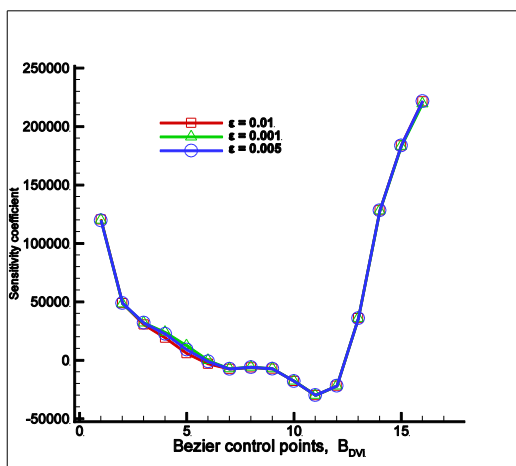
(b) Node number 170



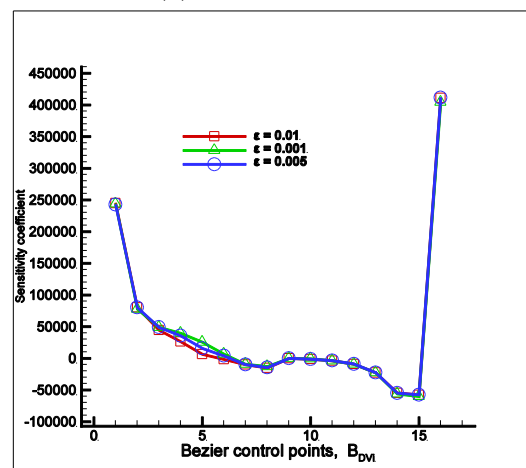
(c) Node number 270



(d) Node number 350



(e) Node number 500



(f) Node number 580

Figure 13. Sensitivity coefficients at some arbitrary nodes using three different finite-difference stepsizes.

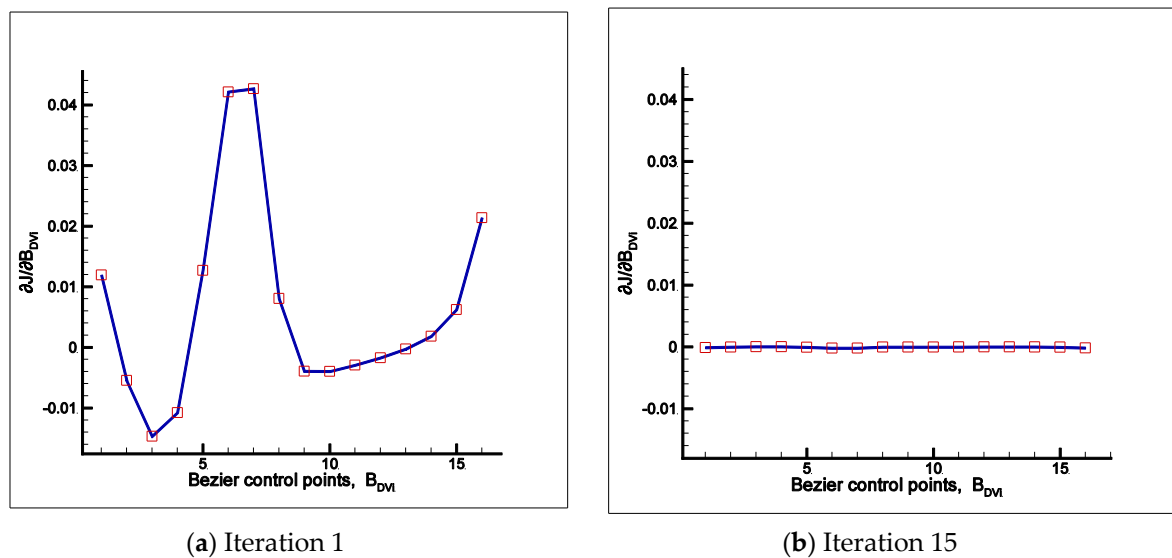


Figure 14. The gradient of the objective function with respect to the design variables at iteration 1 (a) and iteration 15 (b).

It should be emphasized that the gradient of the objective function with respect to the design variables was computed using the finitedifference method. More accurate results are expected if the gradient computation is accomplished with more accurate methods, such as the adjoint method or the complex-step method.

6. Conclusions

This study aimed to develop a novel and exact step length for gradient-based aerodynamic shape optimization problems. The steady inviscid transonic flow governed by the Euler equations was considered and the Euler equations were solved using the finite element solver ANSYS Fluent. Minimization of the drag-to-lift ratio (which is equivalent to the maximization of the lift-to-drag ratio) was performed. The quasi-Newton (BFGS) method was employed to minimize the defined objective function. The airfoil (NACA 0012) surface was parameterized using a Bezier curve and an explicit exact step length was derived for the unconstrained minimization problem. We showed that the value of the exact step length depended on both the values of the gradient of the objective function and the Bezier control points. A test case was presented to investigate the effect of the exact step length on the shape optimization process. Using the exact step length, the optimal shape was obtained within a few iterations with a substantial reduction in the objective function value.

Author Contributions: Conceptualization, F.M. and B.E.; Data curation, F.M.; Formal analysis, F.M.; Funding acquisition, F.M. and B.E.; Investigation, F.M.; Methodology, F.M. and B.E.; Resources, F.M. Software, F.M.; Supervision, B.E.; Validation, F.M.; Visualization, F.M.; Writing—original draft, F.M.; Writing—review & editing, F.M. and B.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement no. 663830.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

- B** Bezier curve control point
- S^(k)** direction of descent at iteration k
- J** objective function
- J** Bernstein basis polynomial
- n** degree of the Bernstein basis polynomial in Bezier curve
- p** pressure
- P** x and/or y in the Bezier curve expression
- t** parameter used in the Bezier curve, $t \in [0, 1]$
- x, y** Cartesian coordinates in the physical domain
- β^(k)** search step length at iteration k
- DV** design variable
- k** iteration number

Appendix A

$$\begin{aligned}
 D(B_{DV_i}^{(k+1)}) = & \left[\underbrace{\left[\overbrace{0}^{B_0^U} \right] (p_{1,1} - p_{1,N-1}) \cos \alpha + \left[\overbrace{0}^{B_0^U} + B_{DV_9}^{(k+1)} 10t_{1,2}(1-t_{1,2})^9 + B_{DV_8}^{(k+1)} 45t_{1,2}^2(1-t_{1,2})^8 + \dots + B_{DV_1}^{(k+1)} 10t_{1,2}^9(1-t_{1,2}) + \overbrace{0}^{B_{10}^U} \right]}^{y_{1,2}} (p_{1,2} - p_{1,1}) \cos \alpha + \right. \\
 & \left. \underbrace{\left[\overbrace{0}^{B_0^U} + B_{DV_9}^{(k+1)} 10t_{1,3}(1-t_{1,3})^9 + B_{DV_8}^{(k+1)} 45t_{1,3}^2(1-t_{1,3})^8 + \dots + B_{DV_1}^{(k+1)} 10t_{1,3}^9(1-t_{1,3}) + \overbrace{0}^{B_{10}^U} \right]}^{y_{1,3}} (p_{1,3} - p_{1,2}) \cos \alpha + \right. \\
 & \vdots \\
 & \left. \underbrace{\left[\overbrace{0}^{B_0^U} + B_{DV_9}^{(k+1)} 10t_{1,\frac{N+1}{2}-1}(1-t_{1,\frac{N+1}{2}-1})^9 + B_{DV_8}^{(k+1)} 45t_{1,\frac{N+1}{2}-1}^2(1-t_{1,\frac{N+1}{2}-1})^8 + \dots + B_{DV_1}^{(k+1)} 10t_{1,\frac{N+1}{2}-1}^9(1-t_{1,\frac{N+1}{2}-1}) + \overbrace{0}^{B_{10}^U} \right]}^{y_{1,\frac{N+1}{2}-1}} (p_{1,\frac{N+1}{2}-1} - p_{1,\frac{N+1}{2}-2}) \cos \alpha + \right. \\
 & \left. \underbrace{\left[\overbrace{0}^{B_0^U} + B_{DV_{10}}^{(k+1)} 10t_{1,\frac{N+1}{2}+1}(1-t_{1,\frac{N+1}{2}+1})^9 + B_{DV_{11}}^{(k+1)} 45t_{1,\frac{N+1}{2}+1}^2(1-t_{1,\frac{N+1}{2}+1})^8 + \dots + B_{DV_{18}}^{(k+1)} 10t_{1,\frac{N+1}{2}+1}^9(1-t_{1,\frac{N+1}{2}+1}) + \overbrace{0}^{B_{10}^U} \right]}^{y_{1,\frac{N+1}{2}+1}} (p_{1,\frac{N+1}{2}+1} - p_{1,\frac{N+1}{2}}) \cos \alpha + \right. \\
 & \left. \underbrace{\left[\overbrace{0}^{B_0^U} + B_{DV_{10}}^{(k+1)} 10t_{1,\frac{N+1}{2}+2}(1-t_{1,\frac{N+1}{2}+2})^9 + B_{DV_{11}}^{(k+1)} 45t_{1,\frac{N+1}{2}+2}^2(1-t_{1,\frac{N+1}{2}+2})^8 + \dots + B_{DV_{18}}^{(k+1)} 10t_{1,\frac{N+1}{2}+2}^9(1-t_{1,\frac{N+1}{2}+2}) + \overbrace{0}^{B_{10}^U} \right]}^{y_{1,\frac{N+1}{2}+2}} (p_{1,\frac{N+1}{2}+2} - p_{1,\frac{N+1}{2}+1}) \cos \alpha + \right. \\
 & \vdots \\
 & \left. \underbrace{\left[\overbrace{0}^{B_0^U} + B_{DV_{10}}^{(k+1)} 10t_{1,N-1}(1-t_{1,N-1})^9 + B_{DV_{11}}^{(k+1)} 45t_{1,N-1}^2(1-t_{1,N-1})^8 + \dots + B_{DV_{18}}^{(k+1)} 10t_{1,N-1}^9(1-t_{1,N-1}) + \overbrace{0}^{B_{10}^U} \right]}^{y_{1,N-1}} (p_{1,N-1} - p_{1,N-2}) \cos \alpha + N \sin \alpha \right] = 0
 \end{aligned}
 \tag{A1}$$

Using $B_{DV_i}^{(k+1)} = a_i^{(k)} - \beta^{(k)} b_i^{(k)}$, we can write Equation (A1) as:

$$\begin{aligned}
 D(B_{DV_i}^{(k+1)}) = & \left[\overbrace{[(a_9^{(k)} - \beta^{(k)} b_9^{(k)}) 10t_{1,2}(1-t_{1,2})^9 + (a_8^{(k)} - \beta^{(k)} b_8^{(k)}) 45t_{1,2}^2(1-t_{1,2})^8 + \dots + (a_1^{(k)} - \beta^{(k)} b_1^{(k)}) 10t_{1,2}^9(1-t_{1,2})]}^{y_{1,2}} \right] (p_{1,2} - p_{1,1}) \cos \alpha + \\
 & \overbrace{[(a_9^{(k)} - \beta^{(k)} b_9^{(k)}) 10t_{1,3}(1-t_{1,3})^9 + (a_8^{(k)} - \beta^{(k)} b_8^{(k)}) 45t_{1,3}^2(1-t_{1,3})^8 + \dots + (a_1^{(k)} - \beta^{(k)} b_1^{(k)}) 10t_{1,3}^9(1-t_{1,3})]}^{y_{1,3}} \right] (p_{1,3} - p_{1,2}) \cos \alpha + \\
 & \vdots + \\
 & \overbrace{[(a_9^{(k)} - \beta^{(k)} b_9^{(k)}) 10t_{1, \frac{N+1}{2}-1}(1-t_{1, \frac{N+1}{2}-1})^9 + (a_8^{(k)} - \beta^{(k)} b_8^{(k)}) 45t_{1, \frac{N+1}{2}-1}^2(1-t_{1, \frac{N+1}{2}-1})^8 + \dots + (a_1^{(k)} - \beta^{(k)} b_1^{(k)}) 10t_{1, \frac{N+1}{2}-1}^9(1-t_{1, \frac{N+1}{2}-1})]}^{y_{1, \frac{N+1}{2}-1}} \right] (p_{1, \frac{N+1}{2}-1} - p_{1, \frac{N+1}{2}-2}) \cos \alpha + \\
 & \overbrace{[(a_9^{(k)} - \beta^{(k)} b_9^{(k)}) 10t_{1, \frac{N+1}{2}+1}(1-t_{1, \frac{N+1}{2}+1})^9 + (a_{11}^{(k)} - \beta^{(k)} b_{11}^{(k)}) 45t_{1, \frac{N+1}{2}+1}^2(1-t_{1, \frac{N+1}{2}+1})^8 + \dots + (a_{18}^{(k)} - \beta^{(k)} b_{18}^{(k)}) 10t_{1, \frac{N+1}{2}+1}^9(1-t_{1, \frac{N+1}{2}+1})]}^{y_{1, \frac{N+1}{2}+1}} \right] (p_{1, \frac{N+1}{2}+1} - p_{1, \frac{N+1}{2}}) \cos \alpha + \\
 & \overbrace{[(a_{10}^{(k)} - \beta^{(k)} b_{10}^{(k)}) 10t_{1, \frac{N+1}{2}+2}(1-t_{1, \frac{N+1}{2}+2})^9 + (a_{11}^{(k)} - \beta^{(k)} b_{11}^{(k)}) 45t_{1, \frac{N+1}{2}+2}^2(1-t_{1, \frac{N+1}{2}+2})^8 + \dots + (a_{18}^{(k)} - \beta^{(k)} b_{18}^{(k)}) 10t_{1, \frac{N+1}{2}+2}^9(1-t_{1, \frac{N+1}{2}+2})]}^{y_{1, \frac{N+1}{2}+2}} \right] (p_{1, \frac{N+1}{2}+2} - p_{1, \frac{N+1}{2}+1}) \cos \alpha + \\
 & \vdots + \\
 & \overbrace{[(a_{10}^{(k)} - \beta^{(k)} b_{10}^{(k)}) 10t_{1, N-1}(1-t_{1, N-1})^9 + (a_{11}^{(k)} - \beta^{(k)} b_{11}^{(k)}) 45t_{1, N-1}^2(1-t_{1, N-1})^8 + \dots + (a_{18}^{(k)} - \beta^{(k)} b_{18}^{(k)}) 10t_{1, N-1}^9(1-t_{1, N-1})]}^{y_{1, N-1}} \right] (p_{1, N-1} - p_{1, N-2}) \cos \alpha + N \sin \alpha \Big] = 0
 \end{aligned} \tag{A2}$$

After factoring, we obtain:

$$\begin{aligned}
 D(B_{DV_i}^{(k+1)}) = & \left[(a_9^{(k)} - \beta^{(k)} b_9^{(k)}) \left[\overbrace{10t_{1,2}(1-t_{1,2})^9(p_{1,2} - p_{1,1}) + 10t_{1,3}(1-t_{1,3})^9(p_{1,3} - p_{1,2}) + \dots + 10t_{1, \frac{N+1}{2}-1}(1-t_{1, \frac{N+1}{2}-1})^9(p_{1, \frac{N+1}{2}-1} - p_{1, \frac{N+1}{2}-2})}^{A_9}} \right] \cos \alpha + \right. \\
 & (a_8^{(k)} - \beta^{(k)} b_8^{(k)}) \left[\overbrace{45t_{1,2}^2(1-t_{1,2})^8(p_{1,2} - p_{1,1}) + 45t_{1,3}^2(1-t_{1,3})^8(p_{1,3} - p_{1,2}) + \dots + 45t_{1, \frac{N+1}{2}-1}^2(1-t_{1, \frac{N+1}{2}-1})^8(p_{1, \frac{N+1}{2}-1} - p_{1, \frac{N+1}{2}-2})}^{A_8}} \right] \cos \alpha + \\
 & \vdots + \\
 & (a_1^{(k)} - \beta^{(k)} b_1^{(k)}) \left[\overbrace{10t_{1,2}^9(1-t_{1,2})(p_{1,2} - p_{1,1}) + 10t_{1,3}^9(1-t_{1,3})(p_{1,3} - p_{1,2}) + \dots + 10t_{1, \frac{N+1}{2}-1}^9(1-t_{1, \frac{N+1}{2}-1})(p_{1, \frac{N+1}{2}-1} - p_{1, \frac{N+1}{2}-2})}^{A_1}} \right] \cos \alpha + \\
 & (a_{10}^{(k)} - \beta^{(k)} b_{10}^{(k)}) \left[\overbrace{10t_{1, \frac{N+1}{2}+1}(1-t_{1, \frac{N+1}{2}+1})^9(p_{1, \frac{N+1}{2}+1} - p_{1, \frac{N+1}{2}}) + \dots + 10t_{1, N-1}(1-t_{1, N-1})^9(p_{1, N-1} - p_{1, N-2})}^{A_{10}} \right] \cos \alpha + \\
 & (a_{11}^{(k)} - \beta^{(k)} b_{11}^{(k)}) \left[\overbrace{45t_{1, \frac{N+1}{2}+1}^2(1-t_{1, \frac{N+1}{2}+1})^8(p_{1, \frac{N+1}{2}+1} - p_{1, \frac{N+1}{2}}) + \dots + 45t_{1, N-1}^2(1-t_{1, N-1})^8(p_{1, N-1} - p_{1, N-2})}^{A_{11}} \right] \cos \alpha + \\
 & \vdots + \\
 & (a_{18}^{(k)} - \beta^{(k)} b_{18}^{(k)}) \left[\overbrace{10t_{1, \frac{N+1}{2}+1}^9(1-t_{1, \frac{N+1}{2}+1})(p_{1, \frac{N+1}{2}+1} - p_{1, \frac{N+1}{2}}) + \dots + 10t_{1, N-1}^9(1-t_{1, N-1})(p_{1, N-1} - p_{1, N-2})}^{A_{18}} \right] \cos \alpha + N \sin \alpha \Big] = 0
 \end{aligned} \tag{A3}$$

which will be:

$$\begin{aligned}
 D(B_{DV_i}^{(k+1)}) = & \left[\underbrace{(a_9^{(k)} - \beta^{(k)} b_9^{(k)}) A_9 + (a_8^{(k)} - \beta^{(k)} b_8^{(k)}) A_8 + \dots + (a_1^{(k)} - \beta^{(k)} b_1^{(k)}) A_1}_{\text{Upper surface}} + \right. \\
 & \left. \underbrace{(a_{10}^{(k)} - \beta^{(k)} b_{10}^{(k)}) A_{10} + (a_{11}^{(k)} - \beta^{(k)} b_{11}^{(k)}) A_{11} + \dots + (a_{18}^{(k)} - \beta^{(k)} b_{18}^{(k)}) A_{18}}_{\text{Lower surface}} \right] \cos \alpha + N \sin \alpha = 0
 \end{aligned} \tag{A4}$$

Thus, the exact step length is obtained as being:

$$\beta^{(k)} = \frac{\left[a_9^{(k)} A_9 + a_8^{(k)} A_8 + \dots + a_1^{(k)} A_1 + a_{10}^{(k)} A_{10} + a_{11}^{(k)} A_{11} + \dots + a_{18}^{(k)} A_{18} \right] \cos \alpha + N \sin \alpha}{\left[b_9^{(k)} A_9 + b_8^{(k)} A_8 + \dots + b_1^{(k)} A_1 + b_{10}^{(k)} A_{10} + b_{11}^{(k)} A_{11} + \dots + b_{18}^{(k)} A_{18} \right] \cos \alpha} \tag{A5}$$

The Fortran 77 code used for calculation of the exact step length given by Equation (A5) is presented in Appendix D.

Appendix B

Appendix B.1. Other Forms of the Objective Function

Appendix B.1.1. Drag Minimization

It is worth mentioning at this stage that the above expression for the step length is equally valid for drag minimization problems as well. In such a case, we can express the objective function as:

$$\mathcal{J} = D^2 \tag{A6}$$

Thus, we can write

$$\frac{\partial \mathcal{J}}{\partial B_{DV_i}} = 2D \frac{\partial D}{\partial B_{DV_i}} \tag{A7}$$

where the term $\frac{\partial D}{\partial B_{DV_i}}$ can be computed using Equation (13). In a similar fashion to Equation (25), to derive an exact expression for the step length, we can write:

$$\frac{\partial \mathcal{J}}{\partial \beta} = 2D \frac{\partial D}{\partial \beta} = 0 \Rightarrow D = 0 \tag{A8}$$

which results in the same expression for the step length given in Equation (31) (or Equation (A5)).

Appendix B.1.2. Lift-Constrained Drag Minimization

In such a case, we can express the objective function as:

$$\mathcal{J} = D^2 \tag{A9}$$

$$\text{subject to } L = \text{constant} = C \tag{A10}$$

The equality constraint, $L = \text{constant} = C$, can be explicitly incorporated into the objective function, $\mathcal{J} = D^2$, to convert the constrained optimization problem to an equivalent unconstrained one. If

$$L = C \tag{A11}$$

Then, from Equation (7), we have:

$$L = -A \sin \alpha + N \cos \alpha = C \tag{A12}$$

Therefore, we can write:

$$N = \frac{C + A \sin \alpha}{\cos \alpha} \tag{A13}$$

By substituting N into the drag force expression (Equation (6)), we have:

$$D = A \cos \alpha + N \sin \alpha = A \cos \alpha + \frac{C + A \sin \alpha}{\cos \alpha} \sin \alpha = \frac{A \cos^2 \alpha + C \sin \alpha + A \sin^2 \alpha}{\cos \alpha}$$

Using $\cos^2 \alpha + \sin^2 \alpha = 1$, the drag force expression simplifies to:

$$D = \frac{A + C \sin \alpha}{\cos \alpha} \tag{A14}$$

or

$$D = \frac{\sum_{j=1}^{N-1} p_{1,j}(y_{1,j} - y_{1,j+1}) + C \sin \alpha}{\cos \alpha} \tag{A15}$$

Now the objective function expression to be minimized is:

$$\mathcal{J} = D^2 = \left(\frac{\sum_{j=1}^{N-1} p_{1,j}(y_{1,j} - y_{1,j+1}) + C \sin \alpha}{\cos \alpha} \right)^2 \tag{A16}$$

The gradient expression is:

$$\frac{\partial \mathcal{J}}{\partial B_{DV_i}} = 2D \frac{\partial D}{\partial B_{DV_i}}$$

$$\frac{\partial \mathcal{J}}{\partial B_{DV_i}} = 2 \left(\frac{\sum_{j=1}^{N-1} p_{1,j}(y_{1,j} - y_{1,j+1}) + C \sin \alpha}{\cos \alpha} \right) \left(\frac{\sum_{j=1}^{N-1} \left[\frac{\partial p_{1,j}}{\partial B_{DV_i}} (y_{1,j} - y_{1,j+1}) + p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \right]}{\cos \alpha} \right)$$

After simplifying (because the lift force is constant, $L = C$, we have $\frac{\partial C}{\partial B_i} = 0$), we obtain:

$$\frac{\partial \mathcal{J}}{\partial B_{DV_i}} = \frac{2}{\cos^2 \alpha} \left(\sum_{j=1}^{N-1} p_{1,j}(y_{1,j} - y_{1,j+1}) + C \sin \alpha \right) \left(\sum_{j=1}^{N-1} \left[\frac{\partial p_{1,j}}{\partial B_{DV_i}} (y_{1,j} - y_{1,j+1}) + p_{1,j} \frac{\partial (y_{1,j} - y_{1,j+1})}{\partial B_{DV_i}} \right] \right) \tag{A17}$$

Equation (A17) is the gradient expression for the equivalent unconstrained optimization form of the lift-constrained drag minimization problem.

To obtain an exact step length for this case, it is necessary to have $D = 0$. From Equation (A16), after a rearrangement of terms with respect to $y_{1,j}$ rather than $p_{1,j}$, we can write:

$$D = \sum_{j=1}^{N-1} p_{1,j}(y_{1,j} - y_{1,j+1}) + C \sin \alpha = \sum_{j=1}^{N-1} y_{1,j}(p_{1,j} - p_{1,j-1}) + C \sin \alpha = 0 \tag{A18}$$

In a similar procedure to the previous cases, we can obtain the exact step length as being:

$$\beta_{\text{exact}}^{(k)} = \frac{\left[\sum_{i=1}^{n-1} \sum_{j=2}^{\frac{N+1}{2}-1} a_i^{U(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) + \sum_{i=1}^{n-1} \sum_{j=\frac{N+1}{2}+1}^{N-1} a_i^{L(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) \right] + C \sin \alpha}{\left[\sum_{i=1}^{n-1} \sum_{j=2}^{\frac{N+1}{2}-1} b_i^{U(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) + \sum_{i=1}^{n-1} \sum_{j=\frac{N+1}{2}+1}^{N-1} b_i^{L(k)} \frac{n!}{i!(n-i)!} t_{y_{1,j}}^i (1-t_{y_{1,j}})^{n-i} (p_{1,j} - p_{1,j-1}) \right]} \tag{A19}$$

or for the Bezier curve of degree 10 used in this study as being:

$$\beta^{(k)} = \frac{\left[a_9^{(k)} A_9 + a_8^{(k)} A_8 + \dots + a_1^{(k)} A_1 + a_{10}^{(k)} A_{10} + a_{11}^{(k)} A_{11} + \dots + a_{18}^{(k)} A_{18} \right] + C \sin \alpha}{\left[b_9^{(k)} A_9 + b_8^{(k)} A_8 + \dots + b_1^{(k)} A_1 + b_{10}^{(k)} A_{10} + b_{11}^{(k)} A_{11} + \dots + b_{18}^{(k)} A_{18} \right]} \tag{A20}$$

Note that in this equation, there is no $\cos \alpha \cdot C$ term as the initial lift force, which should be constant during the optimization process.

Appendix C

The Fortran code for converting the Bezier control points to the design variables is as follows:

```

C n is the degree of Bezier curve
C CPN is the Control Points Number (which is n+1=10+1=11 here)
C B_Y_T and B_Y_B is Bezier control points associated with the top and
C bottom surfaces, respectively
C B_Y is the Bezier design variable (see Figure 3: Two Bezier control
C points adjacent to trailing edge are not considered as design C
C variables in this code;therefore, there are eight design variables on top and
C surface and eight design variables on the bottom surface)

```

```

n=10
CPN=n+1
DO I=1,CPN-3
B_Y(I,1)=B_Y_T(CPN-I-1,1)
ENDDO

DO I=CPN-3+1,2*CPN-6
B_Y(I,1)=B_Y_B(I-CPN+3+1,1)
ENDDO

```

Appendix D

```

C STS stands for step-size (length) calculation
C t_STS is associated with each node on the airfoil surface,
C as defined in Figure 1
C t_T is t for the top surface; t_B is t for the bottom surface
C N is the number of nodes on airfoil surface

```

```

DELTA1_TT=(1.0-0.0)/((N+1)/2-1)
DELTA1_TB=(1.0-0.0)/((N+1)/2-1)
t_T(1,1)=0.0
t_T((N+1)/2,1)=1.0
t_B(1,1)=0.0
t_B((N+1)/2,1)=1.0
DO I=2,(N+1)/2-1
t_T(I,1)=t_T(I-1,1)+DELTA1_TT
ENDDO
DO I=2,(N+1)/2-1
t_B(I,1)=t_B(I-1,1)+DELTA1_TB
ENDDO
t_STS(1,1)=1.0
t_STS(1,(N+1)/2)=0.0
t_STS(1,N)=1.0

DO J=2,(N+1)/2-1
t_STS(1,J)=t_T((N+1)/2+1-J,1)
ENDDO

DO J=(N+1)/2+1,N-1
t_STS(1,J)=t_B(J-(N+1)/2+1,1)
ENDDO

```

```

C for the upper surface

```

```

A(1)=0.0
DO J=2,(N+1)/2-1
A(1)=A(1)+10.0*(t_STS(1,J)**9)*((1.0-t_STS(1,J))**1)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(2)=0.0

```

```

DO J=2,(N+1)/2-1
A(2)=A(2)+45.0*(t_STS(1,J)**8)*((1.0-t_STS(1,J))**2)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(3)=0.0
DO J=2,(N+1)/2-1
A(3)=A(3)+120.0*(t_STS(1,J)**7)*((1.0-t_STS(1,J))**3)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(4)=0.0
DO J=2,(N+1)/2-1
A(4)=A(4)+210.0*(t_STS(1,J)**6)*((1.0-t_STS(1,J))**4)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(5)=0.0
DO J=2,(N+1)/2-1
A(5)=A(5)+252.0*(t_STS(1,J)**5)*((1.0-t_STS(1,J))**5)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(6)=0.0
DO J=2,(N+1)/2-1
A(6)=A(6)+210.0*(t_STS(1,J)**4)*((1.0-t_STS(1,J))**6)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(7)=0.0
DO J=2,(N+1)/2-1
A(7)=A(7)+120.0*(t_STS(1,J)**3)*((1.0-t_STS(1,J))**7)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(8)=0.0
DO J=2,(N+1)/2-1
A(8)=A(8)+45.0*(t_STS(1,J)**2)*((1.0-t_STS(1,J))**8)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(9)=0.0
DO J=2,(N+1)/2-1
A(9)=A(9)+10.0*(t_STS(1,J)**1)*((1.0-t_STS(1,J))**9)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

C for the lower surface

```

A(10)=0.0
DO J=(N+1)/2+1,N-1
A(10)=A(10)+10.0*(t_STS(1,J)**1)*((1.0-t_STS(1,J))**9)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(11)=0.0
DO J=(N+1)/2+1,N-1
A(11)=A(11)+45.0*(t_STS(1,J)**2)*((1.0-t_STS(1,J))**8)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

A(12)=0.0

```

```

DO J=(N+1)/2+1,N-1
A(12)=A(12)+120.0*(t_STS(1,J)**3)*((1.0-t_STS(1,J))**7)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

```

A(13)=0.0
DO J=(N+1)/2+1,N-1
A(13)=A(13)+210.0*(t_STS(1,J)**4)*((1.0-t_STS(1,J))**6)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

```

A(14)=0.0
DO J=(N+1)/2+1,N-1
A(14)=A(14)+252.0*(t_STS(1,J)**5)*((1.0-t_STS(1,J))**5)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

```

A(15)=0.0
DO J=(N+1)/2+1,N-1
A(15)=A(15)+210.0*(t_STS(1,J)**6)*((1.0-t_STS(1,J))**4)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

```

A(16)=0.0
DO J=(N+1)/2+1,N-1
A(16)=A(16)+120.0*(t_STS(1,J)**7)*((1.0-t_STS(1,J))**3)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

```

A(17)=0.0
DO J=(N+1)/2+1,N-1
A(17)=A(17)+45.0*(t_STS(1,J)**8)*((1.0-t_STS(1,J))**2)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

```

A(18)=0.0
DO J=(N+1)/2+1,N-1
A(18)=A(18)+10.0*(t_STS(1,J)**9)*((1.0-t_STS(1,J))**1)*
+(PRESSURE(1,J)-PRESSURE(1,J-1))
ENDDO

```

C terms in Equation (24):

```

DO I=1,2*CPN-4
b_STS(I,1)=0.0
DO L=1,2*CPN-4
b_STS(I,1)=b_STS(I,1)+BY_BFGS(I,L)*DELTA_SY(L,1)
ENDDO
ENDDO

```

```

DO I=1,2*CPN-4
a_STS(I,1)=B_Y(I,1)
ENDDO

```

C calculation of beta (exact step-size):

C terms in numerator of Equation (A5)

```

BETA1_KY=0.0
DO I=1,2*CPN-4
BETA1_KY=BETA1_KY+a_STS(I,1)*A(I)
ENDDO

```

C terms in the denominator of Equation (A5)

```
BETA2_KY=0.0
DO I=1,2*CPN-4
  BETA2_KY=BETA2_KY+b_STS(I,1)*A(I)
ENDDO
```

C Equation (A5):

$$\text{BETA_KY} = \frac{\cos(\text{PI} \cdot \text{AOA} / 180.0) \cdot \text{BETA1_KY} + \text{NORMAL_FORCE}}{\cos(\text{PI} \cdot \text{AOA} / 180.0) \cdot \text{BETA2_KY} + \text{SIN}(\text{PI} \cdot \text{AOA} / 180.0)}$$

References

- Hicks, R.M.; Murman, E.M.; Vanderplaats, G.N. *An Assessment of Airfoil Design by Numerical Optimization*; National Technical Information Service: Springfield, VA, USA, 1974.
- Hicks, R.M.; Henne, P.A. Wing design by numerical optimization. *J. Aircr.* **1978**, *15*, 407–412. [[CrossRef](#)]
- Pironneau, O. On optimum design in fluid mechanics. *J. Fluid Mech.* **1974**, *64*, 97–110. [[CrossRef](#)]
- Jameson, A. Aerodynamic design via control theory. *J. Sci. Comput.* **1988**, *3*, 233–260. [[CrossRef](#)]
- Jameson, A. Optimum aerodynamic design using CFD and control theory. In Proceedings of the 12th Computational Fluid Dynamics Conference, San Diego, CA, USA, 19–22 June 1995; p. 1729.
- Jameson, A. Automatic design of transonic airfoils to reduce the shock induced pressure drag. In Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics, Tel Aviv, Israel, 21–22 February 1990; pp. 5–17.
- Jameson, A.; Reuther, J. Control theory based airfoil design using the Euler equations. In Proceedings of the 5th Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, USA, 7–9 September 1994; p. 4272.
- Lyu, Z.; Kenway, G.K.; Martins, J.R. Aerodynamic shape optimization investigations of the common research model wing benchmark. *AIAA J.* **2014**, *53*, 968–985. [[CrossRef](#)]
- Papadimitriou, D.I.; Papadimitriou, C. Aerodynamic shape optimization for minimum robust drag and lift reliability constraint. *Aerosp. Sci. Technol.* **2016**, *55*, 24–33. [[CrossRef](#)]
- Srinath, D.N.; Mittal, S. An adjoint method for shape optimization in unsteady viscous flows. *J. Comput. Phys.* **2010**, *229*, 1994–2008. [[CrossRef](#)]
- Economou, T.D.; Palacios, F.; Copeland, S.R.; Lukaczyk, T.W.; Alonso, J.J. SU2: An open-source suite for multiphysics simulation and design. *AIAA J.* **2015**, *54*, 828–846. [[CrossRef](#)]
- He, X.; Li, J.; Mader, C.A.; Yildirim, A.; Martins, J.R. Robust aerodynamic shape optimization—From a circle to an airfoil. *Aerosp. Sci. Technol.* **2019**, *87*, 48–61. [[CrossRef](#)]
- Mohammadi, B.; Pironneau, O. *Applied Shape Optimization for Fluids*; Oxford University Press: Oxford, UK, 2009.
- Thevenin, D.; Janiga, G. *Optimization and Computational Fluid Dynamics*; Springer Science & Business Media: Berlin, Germany, 2008.
- Skinner, S.N.; Zare-Behtash, H. State-of-the-art in aerodynamic shape optimisation methods. *Appl. Soft Comput.* **2018**, *62*, 933–962. [[CrossRef](#)]
- Giles, M.B.; Pierce, N.A. An introduction to the adjoint approach to design. *Flow Turbul. Combust.* **2000**, *65*, 393–415. [[CrossRef](#)]
- Giles, M.B.; Duta, M.C.; M-uacute, J.D.; Pierce, N.A. Algorithm developments for discrete adjoint methods. *AIAA J.* **2003**, *41*, 198–205. [[CrossRef](#)]
- Elliott, J.; Peraire, J. Aerodynamic design using unstructured meshes. In Proceedings of the Fluid Dynamics Conference, New Orleans, LA, USA, 17–20 June 1996; p. 1941.
- Nemec, M.; Zingg, D.W. Newton-Krylov algorithm for aerodynamic design using the Navier-Stokes equations. *AIAA J.* **2002**, *40*, 1146–1154. [[CrossRef](#)]
- Anderson, W.K.; Venkatakrishnan, V. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Comput. Fluids* **1999**, *28*, 443–480. [[CrossRef](#)]
- Nocedal, J.; Wright, S. *Numerical Optimization*; Springer: New York, NY, USA, 2006.
- Martins, J.R.; Sturdza, P.; Alonso, J.J. The complex-step derivative approximation. *ACM Trans. Math. Softw.* **2003**, *29*, 245–262. [[CrossRef](#)]

23. Anderson, J.D. *Fundamentals of Aerodynamics*; McGraw-Hill Education: New York, NY, USA, 2016.
24. Rao, S.S. *Engineering Optimization: Theory and Practice*; Wiley: Hoboken, NJ, USA, 2009.
25. Butenko, S.; Pardalos, P.M. *Numerical Methods and Optimization: An Introduction*; Taylor & Francis: Abingdon-on-Thames, UK, 2014.
26. Carpentieri, G.; Koren, B.; van Tooren, M.J.L. Adjoint-based aerodynamic shape optimization on unstructured meshes. *J. Comput. Phys.* **2007**, *224*, 267–287. [[CrossRef](#)]
27. Chen, W.; Zhang, W.; Liu, Y.; Kou, J. Accelerating the convergence of steady adjoint equations by dynamic mode decomposition. *Struct. Multidiscip. Optim.* **2020**. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).