This is a preprint of the following article, which is available from http://mdolab.engin.umich.edu
Xiaolong He, Jichao Li, Charles A. Mader, Anil Yildirim, Joaquim R. R. A. Martins. Robust aerodynamic shape optimization—from a circle to an airfoil. *Aerospace Science and Technology*, 87 (2019) 48–61. doi:10.1016/j.ast.2019.01.051
The published article may differ from this preprint, and is available by following the DOI above.

Robust aerodynamic shape optimization—from a circle to an airfoil

Xiaolong He, Jichao Li, Charles A. Mader,
Anil Yildirim, Joaquim R. R. A. Martins

Abstract

Aerodynamic design optimization currently lacks robustness with respect to the starting design and requires trial and error in the flow solver and optimization algorithm settings to get a converged optimal design. We address this need by developing ways to overcome robustness issues arising from shape parametrization, mesh deformation, and flow solver convergence. Our approach is demonstrated on the Aerodynamic Design Optimization Discussion Group (ADODG) airfoil optimization benchmarks to show the factors that dominate the robustness and efficiency. In the ADODG NACA 0012 benchmark, we address the additional issue of non-unique solutions. In the ADODG RAE 2822 case, we address solver failure due to shock waves, separation, and gradient accuracies due to the frozen turbulence model. Finally, we create a new, challenging aerodynamic shape optimization case that starts with a circle to test the robustness of our aerodynamic shape optimization framework. We use both fixed and adaptive parametrization methods to tackle this problem and show how we can exploit the advantages of adaptive parametrization methods to improve both robustness and efficiency. The combination of flow solver robustness, precision of gradient information, robust mesh deformation, and adaptive parametrization brings us closer to a "push-button" solution for airfoil design.

1 Introduction

Continuous advances in high-performance computing hardware and computational modeling have enabled the design optimization of aircraft with increasing fidelity. However, major challenges remain in engineering design optimization in general, including providing the numerical optimizer the flexibility to widely explore the design space and realizing the gains promised by novel designs. This flexibility in the design space is dependent on the design parametrization, which is usually fixed in an optimization loop. Adaptive parametrization offers the potential to improve the optimization process's efficiency and yield better optimal designs.

There are two main categories of numerical optimizers: gradient-free methods and gradient-based methods. For large-scale, high-fidelity aerodynamic optimization problems based on computational fluid dynamics (CFD), the combination of a gradient-based optimizer and the adjoint method for computing the required gradients has proven to be the most efficient method. In fact, this is the only feasible choice when the number of design variables exceeds 100 or so [1, 2].

The adjoint method is an efficient way of calculating gradients when the number of design variables is much larger than the number of objectives and constraints [3, 4]. The computational cost of solving the adjoint equations is comparable to one flow solution and yields derivatives with respect to all design variables, which is much more efficient than the finite-difference approximations or the complex-step method [5].

Jameson [6] first developed the adjoint method in CFD and applied it to aerodynamic optimization. Since then, various researchers have developed increasingly complex implementations of the adjoint method and have applied them to numerous aerodynamic design optimization problems [2, 7–11].

Despite this extensive work on aerodynamic shape optimization methods and applications, robustness issues with the optimization process have not been well documented. The Aerodynamic Design Optimization Discussion Group (ADODG) proposed a series of benchmark cases to compare and test aerodynamic optimization methods with a focus on the optimization results, but so far there has been little emphasis on robustness [12–20].

By "robustness", we mean the capability of the aerodynamic optimization process to converge to the actual optimal shape starting from a wide range of shapes and flow conditions. For a given aerodynamic design problem, a robust framework should be able to parametrize the shape, deform the mesh while maintaining a high mesh quality, and produce a well-converged CFD solution in addition to performing an efficient optimization. Such robustness requires less manual handling and parameter tuning, leading to a faster turnaround. Troubleshooting issues in any aspect of the optimization process is time-consuming and costly, so we need to identify the key features needed for a robust aerodynamic design optimization framework.

Our objective is to identify the key features needed for such a robust aerodynamic design optimization framework and develop the methods to achieve those features. To demonstrate the developed methodologies, we analyze the robustness of our optimization process for the 2-D ADODG benchmark cases and a new case we developed.

This paper is organized as follows. We briefly introduce the optimization tools used in this work in Section 2, with an emphasis on the new adaptive parametrization that we developed. Section 3 describes the key features needed for overall robustness. In Section 4, we show and discuss the optimization results for all cases: ADODG NACA 0012 and RAE 2822, as well as the new case that starts with a circle. We end with conclusions in Section 5.

2 Methodology

In this section, we provide an overview of our aerodynamic shape optimization framework. We start by introducing the adaptive parametrization that we developed. We then describe the mesh deformation, the CFD solver, and the numerical optimization algorithm.

2.1 Free Form Deformation (FFD) and Adaptive Parametrization

Among the various parametrization methods, we choose the free-form deformation (FFD) method [21–23], which is a versatile method for both local and global shape changes. The main idea of FFD is to embed the object in a flexible solid called the control frame and construct a mathematical mapping from physical space to parametric space. The parametric coordinates of the baseline surface points are solved using the Newton method. By deforming the frame points, changes propagate throughout the interior of the frame so that the object within the frame is smoothly deformed. The advantages of FFD include the ability to start with and handle any baseline geometry, as well as the ability to control the parameterization's level of complexity, the shape at both the global and local levels, and the speed of execution.

Conventional parametrization methods like FFD rely on users to choose the number of design variables. In theory, shape optimization is an infinite-dimensional problem, but in practice, we parametrize shapes with a finite set of design variables. An issue here is that there is no theoretical indicator of whether a given parametrization is good enough for a specific optimization problem. Usually, we set up parametrization methods based on experience from similar applications, but this may not be an option for unconventional design problems. A geometry recovery test may help [17], but this test is limited for known shapes and does not apply to the optimal shapes that are to be obtained. Alternatively, we can run multiple optimizations, using trial and error to find the appropriate way to use parametrization methods, but this incurs substantial designer and computational time. Increasing the number of design variables improves the geometry representation at the cost of increasing the optimization time, so users need to balance between these two aspects. Another issue is that increasing the number of design variables tends to cause more difficulties for the flow solver because of high-frequency shape variations [24]. We explain these issues in more detail in the following sections and propose dealing with them using an adaptive FFD parametrization approach.

In addition to the number of design variables, shape parametrization involves a decision on how to distribute the control points. For example, the position of the FFD control points along the airfoil chord is particularly important. These decisions are usually based on the experience of designers and some trial and error. Therefore, we would like to automate this process as much as possible.

The objectives of this adaptive FFD parametrization are to (1) adaptively increase the number of design variables and (2) make automatic decisions on the parametrization, such as the chordwise distribution of control points on an airfoil.

The advantages of the developed adaptive parametrization method are a reduction

in computation time, as well as increased automation and improved robustness. For a gradient-based optimizer, the number of design variables still has a significant effect on the optimization efficiency, despite the use of the adjoint method for computing the gradient. As such, a smaller number of design variables is desirable in the initial stages of optimization because it saves time.

Another benefit is improved robustness because starting with a smaller number of design variables helps reduce the dimensionality and stabilize the optimization. There are some studies on adaptive FFD methods that only focus on redistributing the control points of the FFD frame without adding more control points [25]. Other adaptive parametrization methods add new design variables in the adaptation step [24, 26, 27].

2.1.1 B-spline FFD

The original FFD method is based on the Bernstein polynomial basis. However, we use B-splines in our implementation for their local modification properties, which we have shown to be advantageous in numerous applications [11, 28–31]. The B-spline basis function in recurrence form can be written as [32, Ch. 2]

$$\begin{aligned} \mathcal{B}_{i,0}(u) &= \begin{cases} 1 & t_i \leq i < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ \mathcal{B}_{i,p}(u) &= \frac{u - t_i}{t_{i+p} - t_i}\mathcal{B}_{i,p-1}(u) + \frac{t_{i+p+1} - u}{t_{i+p+1} - t_{i+1}}\mathcal{B}_{i+1,p-1}(u), \end{aligned} \tag{1}$$

where we can use j and k for the other two directions in the three-dimensional parametric coordinate system, and p represents the degree of the basis function.

Assume that for a given FFD frame there are l, m, n control points in each direction, respectively, and that p, q, and r represent the degree of the basis function in each direction. The length of the knot vector is $l+p+1$, $m+q+1$, $n+r+1$. The knot vector is constructed in the non-decreasing sequence form of $T = \big(0, 0, 0, 0, t_4, t_5, \ldots, t_{N-1}, 1, 1, 1, 1\big)$ if $p = 3$ and the number of control points is N. The equation governing the FFD method using a B-spline volume can be written as

$$X(u, v, w) = \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} \mathcal{B}_{i,p}(u)\mathcal{B}_{j,q}(v)\mathcal{B}_{k,r}(w)P_{i,j,k}, \tag{2}$$

where $X = (x, y, z)$ is a point in the global coordinate system, (u, v, w) represents the corresponding parametric coordinates for X, and $P_{i,j,k}$ are the coordinates of every control point in global coordinate system.

2.1.2 Knot Insertion

The adaptive parametrization method we developed progressively adds more design variables by adding more control points to the FFD frame. There are two ways to achieve this goal. The first is to modify the FFD frame directly. Duvigneau [25] used a metric to modify the baseline FFD frame to embed the baseline shape, which requires a least-squares fitting to the changed shape.

A second option is to generate an FFD frame around the changed shape and resolve the parametric coordinates within the new frame. This can be done by adding control points through a B-spline knot insertion method [26]. It is advantageous to increase the number of control points in the FFD and keep the embedded shape unchanged when we add design variables. Since we use a B-spline basis, the knot insertion method can be adopted here [26, 32]. The idea of knot insertion is to insert a new knot entry into the knot vector and generate a new FFD frame based on this new knot vector. During this change, the shape embedded in the FFD frame remains unchanged because it does not need to be fitted after the insertion. Another advantage of this approach is that we do not need to resolve for the parametric coordinates, although the time for this computation is only a small fraction of the whole optimization process.

We use a B-spline curve as example, which is defined by

$$C(u) = \sum_{i=0}^{n} \mathcal{B}_{i,p}(u) P_i, \tag{3}$$

where P is the control point vector of length $n+1$, and the knot vector is $T = (t_0, t_1, \ldots, t_m)$. After insertion, we use Q to represent the new control point vector of length $n+2$ and $T = (t_0, t_1, \ldots, t_k, t^*, t_{k+1}, \ldots, t_m)$ for the new knot vector. Here, t^* is the new knot inserted between t_k and t_{k+1}. Due to the characteristics of B-spline basis functions we get

$$\sum_{i=0}^{n} \mathcal{B}_{i,p}(u) P_i = \sum_{i=0}^{n+1} \mathcal{B}^*{}_{i,p}(u) Q_i, \tag{4}$$

which means that after the insertion operation, both the global and parametric coordinates of the curve are unchanged. Solving the equation above, we get a formula for computing all the control points of new FFD frame:

$$\begin{aligned} Q_i &= \gamma_i P_i + (1-\gamma_i) P_{i-1} \\ \gamma_i &= \begin{cases} 1 & i \le k-p \\ \frac{t^*-t_i}{t_{i+p}-t_i} & k-p+1 \le i < k \\ 0 & i \ge k+1 \end{cases} \end{aligned} \tag{5}$$

Alternatively, we can insert a control point at a specific location P^* and calculate the needed knot value by

$$\begin{aligned} t^* &= t_{k+1} + \beta_i (t_{k+p} - t_{k+1}) \\ \beta_i &= \frac{P^* - P_k}{P_{k+1} - P_k}. \end{aligned} \tag{6}$$

Figure 1 shows the effect of inserting twice in two different locations. More specifically, we can see how the inserted knot affects neighbor control points.

2.1.3 Adaptation Metric

The knot insertion method serves as a tool to add new design variables, but we still need to find which knot value or control point we should insert and when to do that.

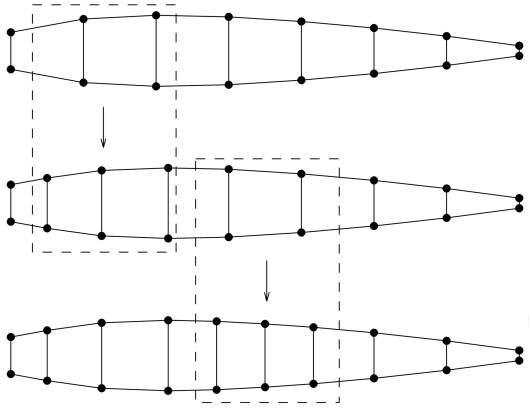


Figure 1: The first knot is inserted near the leading edge, and the second knot is inserted at mid-chord. Note that two neighbor control points move after the insertion.

In a two-dimensional FFD frame, we have two knot vectors: one for each of the i and j directions. We only insert knots in the i direction, which is along the chord, but there are still an infinite number of possibilities.

To determine where to insert the knots, we use an adaptation metric (AM). Our adaptation metric includes constraint information, and it is similar to the adaptation metric developed by Anderson and Aftosmis [27], which uses both the first and second derivative information. Another AM option is the one proposed by Han and Zingg [26], who used the gradient of the single candidate control point as AM. However, when knot insertion changes the location of neighbor control points, taking the gradient information of all the control points into account is more appropriate.

Since we do not have second derivatives readily available, we simplify the metric to use only first derivatives, yielding

$$\mathrm{AM} = \frac{1}{2} \sum_{i=1}^{N_y} \left(\frac{\partial f}{\partial y_i} - \sum_{j=1}^{N_g} \frac{\partial g_j}{\partial y_i} \right)^2 , \tag{7}$$

where N_y is the number of shape design variables, f is the objective function, y_i represents a component of the design variable vector, N_g is the number of active constraints, and $\partial g_j / \partial y_i$ is one element from the constraints Jacobian matrix. The constraints Jacobian matrix is computed by taking the sensitivity of each active constraint with respect to every design variable. The constraints consist of geometric constraints (thickness and volume), and aerodynamic force coefficient constraints (lift and pitching moment).

As explained in the previous section, we build knot vectors for B-spline functions. There are $N_N + 1$ knot values in the $[0, 1]$ interval, where the value of N_N is decided by the number of control points and the degree of the curve. We get N_N intervals, and we can insert new knot values in the middle point of each interval. Then, we generate N_N candidate FFD frames by inserting knots at each candidate interval. The next step is to evaluate the adaptation metric for each candidate frame and choose the

frame that yields the highest adaptation metric value. In this process, we reuse the adjoint solution because the airfoil shape is not changed, and the FFD frame is the only changed part in the chain rule for the gradient computation.

2.2 Mesh Deformation

The parametrization generates new shape surfaces, and during optimization, the CFD surface mesh serves as a representation of the geometry. Each time the optimizer changes the shape design variables, we need to perform a mesh deformation of the original mesh and get a new volume mesh by propagating the surface mesh changes. To accomplish this, we use two different approaches for mesh deformation. The first approach is based on an algebraic method and is implemented in the pyWarp package [23]. pyWarp uses normalized arc length along mesh lines to decay the deformation for the volume mesh points. This approach can only handle small changes, as it usually runs into difficulties when deforming the tightly-spaced points across the boundary layer.

The second approach for mesh deformation is based on inverse distance weight (IDW) interpolation [33] and is implemented in the IDWarp package. IDW considers surface cell displacement and rotation to preserve cell orthogonality and is able to handle large shape deformations, even for meshes that include tightly-spaced points.

For airfoil optimizations starting from a circle or random shapes, large shape changes are required to reach the optimum, which causes a loss in the orthogonality of the near-wall mesh cells. In those cases, the algebraic scheme is not enough to maintain good mesh quality during optimization, so we use IDWarp exclusively.

2.3 CFD Solver

The CFD solver used in this work is ADflow, which can solve Euler and Reynolds-averaged Navier–Stokes (RANS) equations using multiblock or overset meshes [34]. ADflow uses the Jameson—Schmidt—Turkel scheme with artificial dissipation as the discretization scheme and provides a few turbulence models, including the Spalart–Allmaras (SA) turbulence model [35]. Explicit multi-stage Runge–Kutta (RK) or diagonalized diagonally-dominant alternating direction implicit (D3ADI) [36] methods can be used to solve the resulting nonlinear systems.

Alternatively, we can use a combination of approximate Newton–Krylov (ANK) and Newton–Krylov (NK) solvers [37]. There are two main stages when using this approach. In the initial stage of convergence, we use the ANK solver to reduce the total residual norm by five orders of magnitude compared to the first iteration, which is initialized with free-stream conditions. After the initial reduction, we switch to the NK solver for the terminal stage of convergence. Similarly, we can also use the RK or D3ADI solvers instead of the ANK solver for the initial five orders of magnitude of convergence, to obtain a good initial guess for the NK solver.

ADflow includes a discrete adjoint method that efficiently computes the derivatives of objective and constraint functions with respect to large numbers of design variables [38, 39]. The adjoint implementation includes the complete linearization of the

turbulence model. The preconditioned GMRES solver from PETSc [40] is used for solving the adjoint equations.

2.4 Optimization Algorithm

To perform the numerical optimization, we use SNOPT (Sparse Nonlinear OPTimizer), which utilizes a sequential quadratic programming (SQP) approach and approximates the Hessian matrix using a quasi-Newton method [41]. Equality and inequality constraints are handled by SNOPT using a sequential quadratic approach. A Python interface to SNOPT is used for convenience and flexibility [42] [1].

3 Key Features for Robustness

As mentioned in the introduction, what we mean by robustness is the ability to obtain optimal shapes successfully for a variety of starting shapes and flow conditions in a fully automated fashion. There are a few factors that determine the robustness of the optimization process. Based on our experience, the most frequent reasons for a failed optimization are flow solver failures (especially due to the cases that exhibit massive flow separation), inaccurate gradients, mesh deformation failure, and inappropriate parametrization. In this section, we discuss how these issues affect robustness and what we have done to mitigate them.

3.1 Flow Solver Robustness

The flow solver's ability to converge to the solution of the governing equations for a given mesh is of paramount importance for optimization. The additional challenge for a flow solver when using numerical optimization is that the optimizer is likely to come up with designs that a human designer would not even try in the course of a CFD-based manual aerodynamic design process. Some optimization algorithms provide the option to process a point that cannot be evaluated without interrupting the optimization, but numerous failed evaluations add to the computational cost without providing much information to the optimizer. Besides slowing the optimization process down, these intermediate cases can even cause the process to completely stall, and prevent the optimizer from converging to the optimal design.

To avoid these problems, we require a robust flow solver that can always find a solution to a nonlinear system of equations, even for challenging cases. We have developed the ANK solver implemented in ADflow with this requirement in mind [37]. We primarily use the ANK solver for the initial stages of convergence, where a pure NK solver would fail to converge. The simulations are started with the ANK solver until five orders of magnitude reduction in the total residual norm is achieved.

Both the NK and ANK solvers use Newton's method to converge the nonlinear system of equations, and a Krylov subspace method to solve the resulting linear systems in each nonlinear iteration, hence the name Newton–Krylov. However, compared to the NK solver, the ANK solver has a number of additional features that are added for improved robustness.

The ANK solver's main distinction is the approximate Jacobian it uses, compared to the exact Jacobian used in the NK solver. The approximate Jacobian is similar to a first-order accurate Jacobian, and it has a smaller bandwidth compared to the exact Jacobian we use, which is second order accurate. This improves the conditioning of the linear systems that we solve during each nonlinear iteration at the cost of lower accuracy in the nonlinear updates. As a result, we can achieve linear solutions with less computational effort, and the chance of a linear solver failure is reduced.

Along with the approximations in the Jacobian formulation, we introduce a diagonal time-stepping term in the linear system we solve at each nonlinear iteration, resulting in a backward Euler scheme. The time-step is initialized with a relatively small value, and it is ramped up as the simulations converge, which is commonly known as pseudo-transient continuation [43]. This technique helps the solver settle the strong transients that are present during the initial stages of convergence. Furthermore, it provides favorable nonlinear convergence rates towards the final stages of convergence, where the algorithm approaches a pure Newton method as the time-step increases.

After obtaining the update vector by solving the resulting linear system, the ANK solver uses two algorithms to control the step size in the direction of the update. First, we use a physicality check to ensure that the nonlinear updates do not yield unphysical results such as negative density or pressure values. Secondly, we utilize a simple backtracking line search algorithm to ensure that the unsteady residual norm reduces between nonlinear iterations. These two algorithms are fundamental in achieving the solver's nonlinear robustness.

Besides the details of the ANK solver algorithm, there is often a trade-off between performance and robustness when tuning the parameters that control the solver behavior. A good example of this is the relative nonlinear convergence tolerance that we use to switch from the ANK to the NK solver. Switching to the NK solver earlier would provide improved performance for most cases. However, in some problematic cases, prematurely switching to the NK solver causes the solver algorithm to stall, whereas a lower switch tolerance would avoid these issues by providing a better initial guess for the NK solver using a tighter convergence with the ANK solver. In this context, we have selected a lower relative convergence level (10^{-5}) for the switch rather than a level that is sufficient for most cases (10^{-4}). This choice improves the robustness of our nonlinear solver, although it comes at the cost of sub-optimal performance for easier cases. All of these factors contribute to the nonlinear robustness of the ANK solver, and we are using it extensively as a robust startup strategy.

RANS simulations are shown to reliably predict the performance of wings at cruise conditions where there is little to no separation [44]. On the other hand, for cases where there is massive separation in the flow-field, the equilibrium-flow assumption that is used to derive the RANS equations are not valid, and therefore RANS simulations do not yield accurate results for these cases. However, it is still useful to be able to converge such off-design cases because they tend to provide the correct performance trends. If the goal is to minimize the drag (which is the case for the vast majority of aerodynamic shape optimization problems), the optimizer converges to a smooth shape with little or no separation for which the RANS model is valid.

For example, Lyu et al. [2] studied the aerodynamic shape optimization of a wing in transonic conditions, starting from random shapes that contained separation. For all cases with different starting points, the optimizer converged to smooth wing shapes with no separation. In this example, even though the RANS model may be inaccurate for the random initial designs, the model is valid about the optimal design because there is little to no separation in this part of the design space. Furthermore, the sensitivities obtained with these initial designs were accurate enough to point the optimizer towards a region of the design space without separation. This type of robustness is useful when designers have to start with a rough baseline geometry or a geometry that had been previously designed for different flight conditions.

To demonstrate the robustness of the ANK solver in ADflow, we test 1172 airfoils from the UIUC airfoil database in subsonic and transonic conditions, and we show which cases converged and which did not in Figure 2 [45]. The horizontal and vertical axes are the thickness and camber calculated from the geometry. The combination of large thickness and camber leads to separation that caused difficulties for the flow solver. For $M = 0.45$, all of the 1172 airfoils converged with a density residual of 10^{-12}, regardless of the time marching method. For $M = 0.73$, nearly half of airfoils failed to converge using the D3ADI and NK solvers. Failed cases have larger thickness and camber, which indicates difficulties from combined strong shock and flow separation. However, the ANK solver converged *all* cases. We select 20 airfoils from Figure 2c (see labels) and plot them in more detail in Figure 3. In particular, airfoils number 18, 19, and 20 have the largest thickness, camber, or both. Being able to converge the flow solution for these extreme airfoils and all the others demonstrates ADflow's robustness.

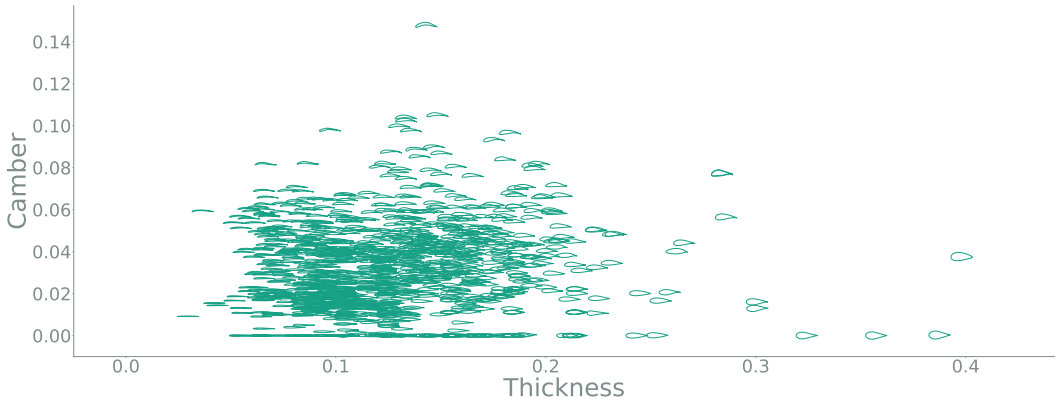
In Section 4, we discuss the flow solver robustness issues in more detail for each case. Here, we provide just a brief overview of the issues encountered in the optimizations. In the NACA 0012 case, we had difficulties converging and found non-unique solutions for the Euler equations. The RAE 2822 case was less problematic when starting from the RAE 2822 baseline or random initial shapes. In this RANS case, we found no evidence of non-unique solutions. In the case starting from a circle, the flow solution is challenging because of the massive flow separation, but ANK was able to converge. Because ANK converges to a static RANS solution, it does not represent the real unsteady flow correctly.

3.2 Influence of Parametrization

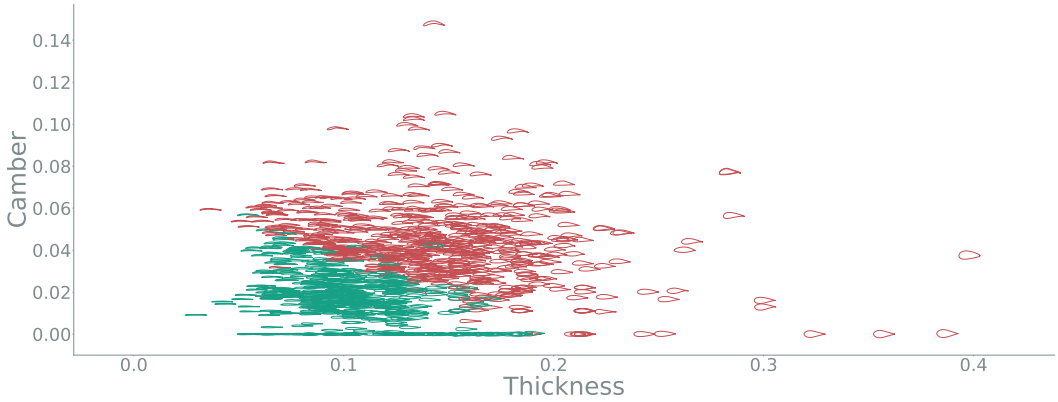
The FFD parametrization generates new shapes based on the design variables, which determine the deformed control point positions. The parametrization method affects the robustness in two ways.

First, the parametrization needs to be able to capture the geometry of the optimal shape, which is not known *a priori*. The distribution of control points along the chord will affect the design space, which means the shape can be parametrized by FFD.

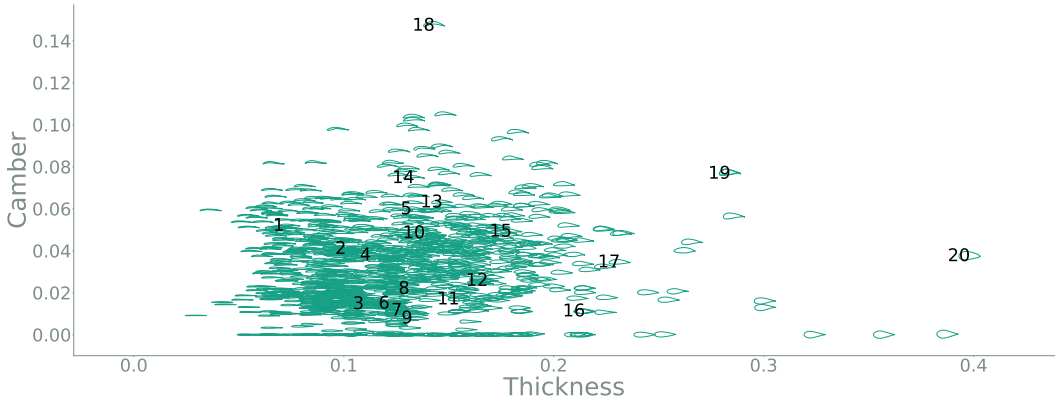
Second, the number of design variables influences not only the speed but also the robustness of the optimization process. Based on our observations, when the number of design variables increases, the optimizer is prone to exploring shapes with more curvature variation that causes difficulties for the flow solver.



(a) ANK/D3ADI and NK solvers at $M = 0.45$.



(b) D3ADI and NK solvers at $M = 0.73$.



(c) ANK and NK solvers at $M = 0.73$.

Figure 2: Demonstration of solver robustness when solving for 1172 airfoils, showing cases that converged (green) and failed (red) at $Re = 6.5 \times 10^6$, $\alpha = 1$.

Figure 4 shows an intermediate airfoil from a failed optimization starting from a circle. The separation and resulting unsteadiness of the flow around this shape caused failure in the optimization because of the low accuracy in the gradient.

In addition, when the spacing between control points is too small relative to the spacing of the surface CFD mesh points, an excessive number of FFD control points

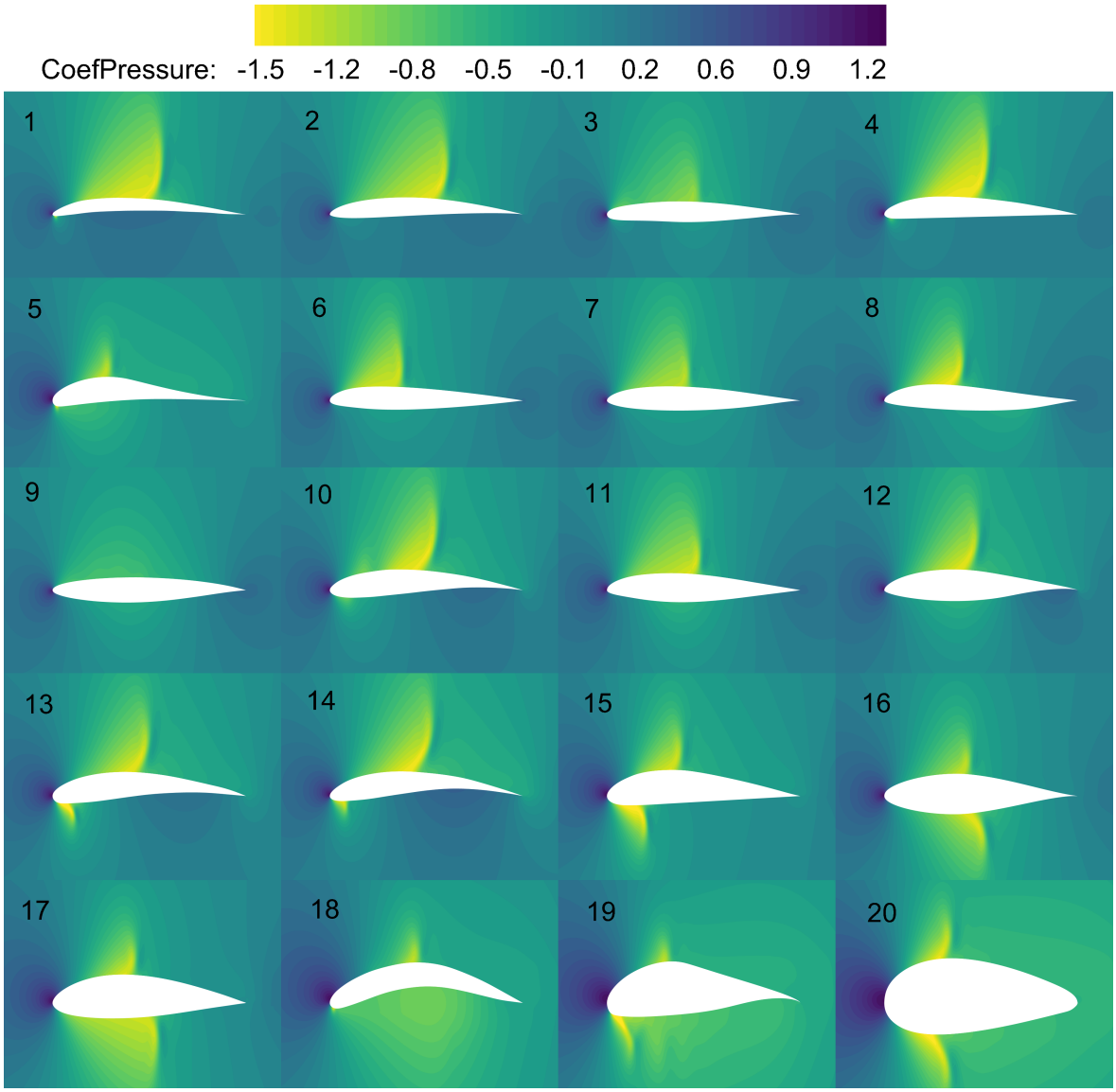


Figure 3: Pressure coefficient contours for 20 sample airfoils.

may result in multimodal shapes. This issue caused the optimization using 160 design variables for the RAE 2822 case to fail. Other researchers also encountered similar issues using other parameterization methods [24]. The issue related to the number of design variables is discussed in the following sections. The solution to this problem is to set the spacing of FFD control points to be at least four times larger than that of CFD mesh points.

To mitigate the flow solution robustness issues, it is possible to tune the flow solver and optimizer parameters. However, this is time consuming, especially if the designers are not familiar with the particular solver and problem. To address this issue, we propose an adaptive parametrization that starts with a few variables to get a rough approximation of the optimal shape, then increases the number of control points to

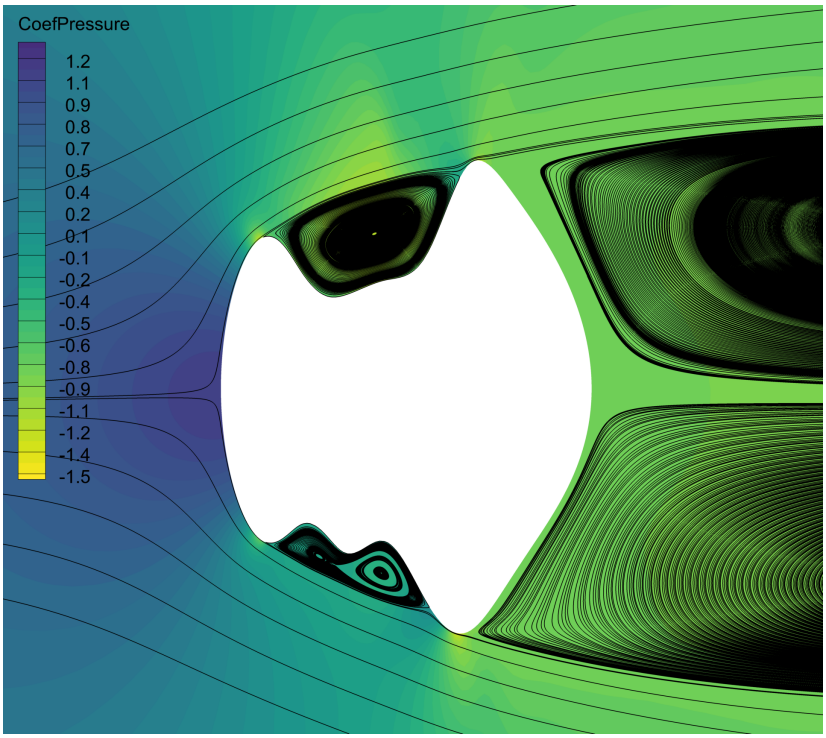
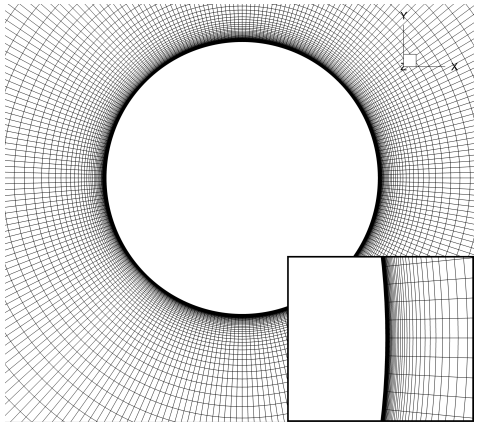


Figure 4: Anomalous shape encountered during an optimization starting from a circle with 20 design variables

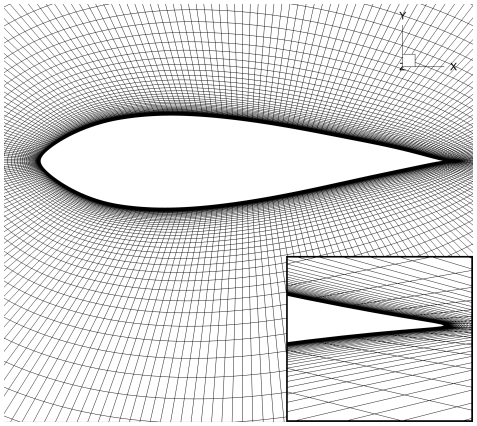
refine the shape.

3.3 Robust Mesh Deformation

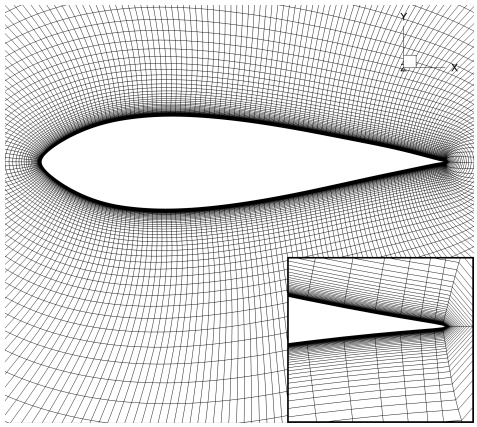
The mesh deformation generates new volume meshes using given a baseline shape and a new surface shape. The quality of the deformed mesh is a key factor in the robustness and accuracy of the overall shape optimization process. When a case requires a large shape change, like going from a circle to an airfoil, maintaining the mesh quality through the deformation is crucial. For small shape changes, the algebraic mesh deformation in pyWarp is fast and adequate. For larger shape changes that cause larger mesh deformations, however, the inverse distance algorithm in IDWarp is best at preserving mesh orthogonality and avoiding the generation of negative volume cells, especially in the off-wall mesh. Of course, there may still be some cases where even IDWarp fails due to cells with negative volumes. In those cases, we could regenerate the mesh using pyHyp, a hyperbolic mesh generation tool that, given a surface mesh, automatically generates volume meshes. Figure 5 compares the results of deforming a mesh from a circle to an airfoil with pyWarp and IDWarp, as well as the mesh regeneration with pyHyp. Since this is a large mesh deformation, pyWarp yields low-quality off-wall cells, but IDWarp performs much better. Regenerating the mesh with pyHyp yielded the best mesh quality but is more costly: While pyWarp takes 0.02 s and IDWarp takes 0.15 s, pyHyp requires 15 s to regenerate the mesh.



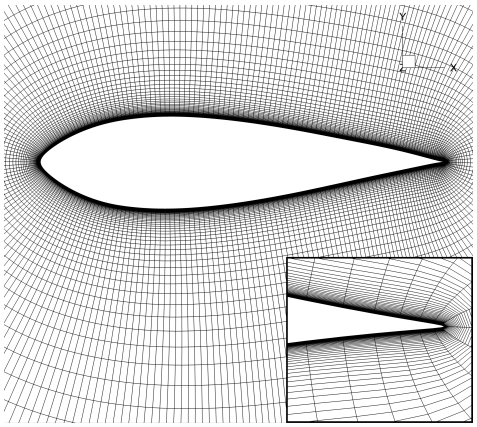
(a) Initial mesh.



(b) pyWarp lost orthogonality near the wall after large deformation from a circle.



(c) IDWarp preserves orthogonality near the wall.



(d) Mesh generated for the same airfoil using pyHyp has better quality near trailing edge.

Figure 5: Comparison of the initial mesh for the circle and corresponding meshes deformed using pyWarp, IDWarp, and pyHyp.

4 Results

This section shows results for three cases, each with the goal of demonstrating different aspects of the aerodynamic shape optimization robustness. In the NACA 0012 and RAE 2822 cases, we show key parameters that affect the robustness the most and show how the adaptive FFD improves the optimization process robustness. The circle case is used as the ultimate test for the robustness of our optimization framework and the adaptive FFD approach.

4.1 Inviscid Airfoil Case

In this section, we solve the ADODG 2-D NACA 0012 case, which is an inviscid airfoil optimization problem. The baseline shape is a modified sharp trailing edge NACA 0012 airfoil defined as

$$y = \pm 0.6 \left(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1036x^4\right), \tag{8}$$

where $x \in [0, 1]$. First, we perform a mesh convergence study, where we generate a family of meshes with different sizes using pyHyp. Since this case is symmetric with an angle of attack of zero, we use only the upper half mesh with a symmetry boundary condition at $y = 0$. The off-wall spacing is 4×10^{-4}, and the far-field is 150 chords away. The mesh sizes and drag coefficients for all meshes are listed in Table 1. The meshes range from 8000 cells (L3) to 2 million cells (L00). The drag values in this table are multiplied by two to get drag for the whole airfoil. The drag difference between L0 and L00 mesh is 0.04 counts. We chose the L1 mesh for optimization for a good compromise between accuracy and computational cost. Figure 6 shows the L1 mesh mirrored about the chord line.

We generate a few FFD frames with different numbers of control points. A 52-control-point example is shown in Figure 7, which we reduce to 26 design variables by enforcing symmetry. The frame is important in the FFD because it is directly related to how the airfoil is parametrized. The FFD frame is fitted close to the airfoil surface to allow for better control. Based on our experience, setting up control points close to the surface with approximately the same distance is beneficial for faster optimization convergence. The control points are offset by a small distance from the surface (1% chord) and are uniformly distributed along the chord. Non-uniformly distributed control points along chord, such as denser control points near the leading edge, usually lead to better optimization results. Here we consider uniform distributed control points for the sake of generality.

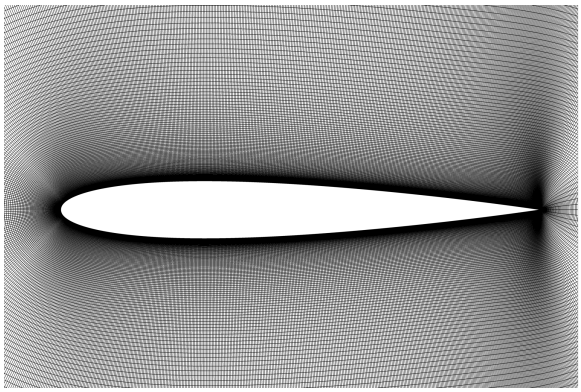


Figure 6: L1 grid.

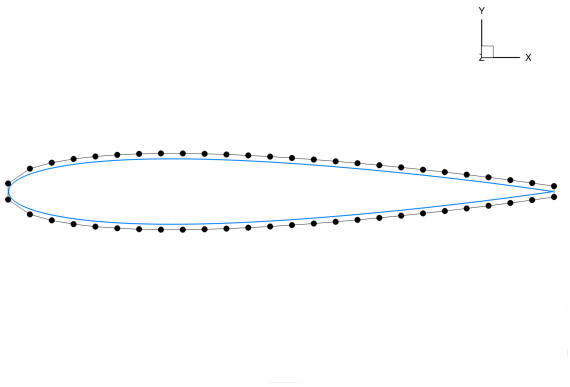


Figure 7: Example FFD frame with 52 control points.

Table 1: Mesh convergence for the NACA 0012 and the optimized airfoil using 50 design variables.

Mesh	Number of cells	Drag counts	
		NACA 0012	Optimized
L00	2,097,152	470.55	73.14
L0	524,288	470.59	69.28
L1	131,072	470.36	72.46
L2	32,768	469.88	70.38
L3	8,192	468.76	77.40

Table 2: ADODG NACA 0012 case problem statement

Category	Name	Quantity	Lower	Upper
Objective	C_d	1	–	–
Variables	y	14/18/22/26/50	-0.5	0.5
Constraints	t/c	25	1	–
	$y_{\text{lower}} = -y_{\text{upper}}$	14/18/22/26/50	0	0

4.1.1 Problem Statement

The mathematical formulation of the NACA 0012 optimization is shown in Table 2. The Mach number is 0.85, and the design variables are the FFD frame control point movements in the vertical direction, y. The number of control point design variables, N_y, ranges from 14 to 50. Relative thickness (t/c) constraints are enforced at 25 positions along the chord to ensure that the airfoil thickness is larger than or equal to that of the baseline.

4.1.2 Optimization Results

For this NACA 0012 inviscid optimization, we converge the flow solution residuals to 10^{-10}, and the optimality in SNOPT is set to 10^{-6}. This optimization case is challenging to converge because it is sensitive to solver and optimization settings, especially for the larger numbers of variables. To address this issue, we found it helpful to use the "major step limit" in SNOPT, which sets the maximum possible step size in the line search, thus limiting the change of design variables during a line search. This was particularly helpful to deal with the non-uniqueness issue of this case.

In addition, we found it useful to set the "Hessian updates" to a value larger than the default (from 10 to 50). This determines the number of updates before resetting the quasi-Newton Hessian approximation, and the larger the value, the better the approximate Hessian.

Figure 8 shows the objective function history for a few cases, where each column corresponds to a different number of design variables. The data corresponding to the given number of variables is shown in color, while the gray lines show the data for the other cases for reference. We can see that the optimum drag value decreases when the design variable number increases and it gets harder to converge. The cases that are

harder to converge need adjustments to the flow and optimizer parameters. When the number of design variables reaches 50, there is still a trend towards lower drag with increasing number of design variables.

This trend is mainly caused by the need to form blunt shapes at the leading and trailing edges in this non-physical inviscid case. Figure 8 also shows the feasibility history, where the constraint violation is within 10^{-6}. All cases with different numbers of design variables converged well.

Table 3 shows our minimum C_d results and compares them with other published results; our result with 50 variables is the lowest. We list the drag result from the finest mesh in each paper. Note that the drag results of the baseline shape differ greatly from paper to paper. The optimized airfoils show a even larger difference for drag. The first reason for this is that the flow solvers and mesh types used in each paper are different. Even if we take zero-spacing drag results from each paper, the difference will still be larger than five counts. The second reason is that these papers are using different parametrization methods. These papers gave different optima because the design space is determined by parametrization.

In this table, our C_d is multiplied by two to obtain the drag coefficient of the whole airfoil, since we solved only the half mesh. All optimized airfoils share the same trend: increasing suction peak and shock moving toward the trailing edge. All optimized airfoils tend to exhibit a larger leading edge radius and an abrupt trailing edge after optimization. There are fluctuations in the C_p distribution near the trailing edge, which may be related to how FFD points are arranged since we used uniformly distributed control points.

Table 3: Comparison of the drag for NACA 0012 Euler optimization, including results reported in the literature; a zero is added in the second decimal place when only one decimal place was reported.

Result	N_y	NACA 0012	C_d (counts)
Present work	50	470.36	7.60
Present work	26	470.36	15.54
Present work	22	470.36	23.53
Masters et al. [20]	16	469.60	25.00
Bisson and Nadarajah [16]	32	464.20	26.20
Present work	18	470.36	34.33
Carrier et al. [12]	96	471.20	36.70
Anderson et al. [15]	31	471.30	41.30
Lee et al. [46]	9	457.33	42.24
Present work	14	470.36	50.49
Zhang et al. [19]	17	481.28	73.08
Poole et al. [47]	15	469.42	83.80
LeDoux et al. [14]	513	471.30	84.50
Gariepy et al. [48]	11	481.60	141.70
Fabiano and Mavriplis [18]	11	466.96	297.02

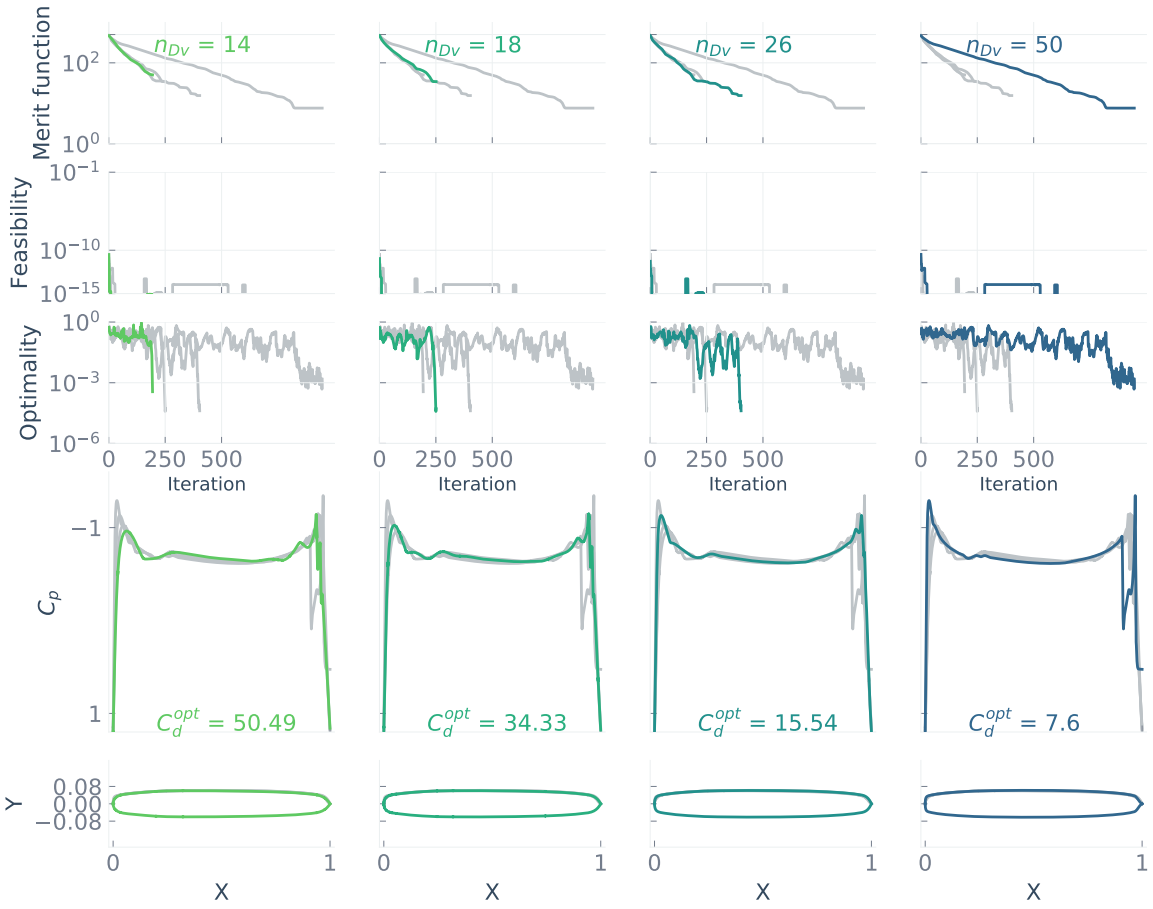
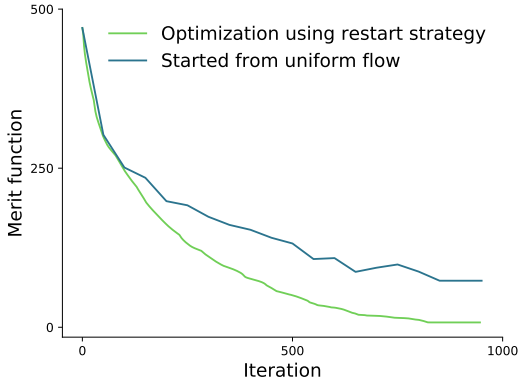


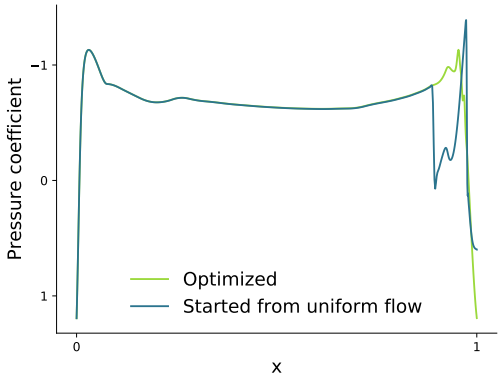
Figure 8: Merit function, feasibility, optimality history, optimized shape, and pressure distribution for the NACA 0012 case. The gray lines show the results from the other cases for comparison.

To examine the issue of non-uniqueness in the solution that was previously reported for this case [14, 20], we re-analyze the flow equations for some intermediate optimization steps. When performing aerodynamic shape optimization, the flow solution of the previous optimization iteration is used as the starting point for solving the governing equations. For the re-analysis, the flow solution is always started from a uniform flow field. Figure 9a compares the drag coefficient history from optimization and re-analysis values, which reveals the non-uniqueness problem of the Euler equations solution. Figure 9b shows that starting from uniform inflow yields different C_p distributions that mainly differ in the shock shape. This difference in C_p partially explains why solving the flow equations from the uniform flow and from the previous solution yields different drag values. Given this result, we can see that in this case, the optimization result is clearly affected by the non-uniqueness issue.

We first performed these optimizations using the L1 mesh. Then, we generated a mesh family for the optimized airfoil by performing the same deformation on the baseline airfoil mesh family. In the mesh convergence study, we initialized the flow



(a) Comparison of drag from optimization and re-analysis starting from uniform flow



(b) Non-unique solution of the optimized airfoil

Figure 9: Re-analysis for optimization intermediate steps shows non-unique flow solutions.

Table 4: Mesh convergence study for the RAE 2822 at $M = 0.734$, $Re = 6.5 \times 10^6$, $C_l = 0.824$

Mesh	α	Number of cells	C_d (counts)	
			RAE 2822	Optimized
O1	2.785806	131,072	194.40	108.91
O2	2.817267	32,768	198.41	109.89
O3	2.915244	8,192	214.66	119.01
C1	2.757314	2,240,000	191.00	109.51
C2	2.743502	560,000	192.15	110.53
C3	2.773300	130,000	199.15	115.60
C4	2.800754	35,000	205.53	118.28

solution from uniform flow.

4.2 Viscous Airfoil Case

We now consider the ADODG RANS airfoil optimization problem, which starts from the RAE 2822 airfoil [2]. We also performed a mesh convergence study for the RAE 2822 with two families of meshes generated using pyHyp. The number of cells and drag coefficients for all the meshes is listed in Table 4. The optimization used 40 design variables for the O2 mesh and 52 design variables for the C4 mesh. An O-type mesh family was used for the drag comparison along with the result from the circle case to keep the mesh type consistent. The O2 mesh, which has 32,768 cells, is shown in Figure 10. The nominal flow condition is $M = 0.734$ and $Re = 6.5 \times 10^6$, where the lift coefficient is constrained to 0.824.

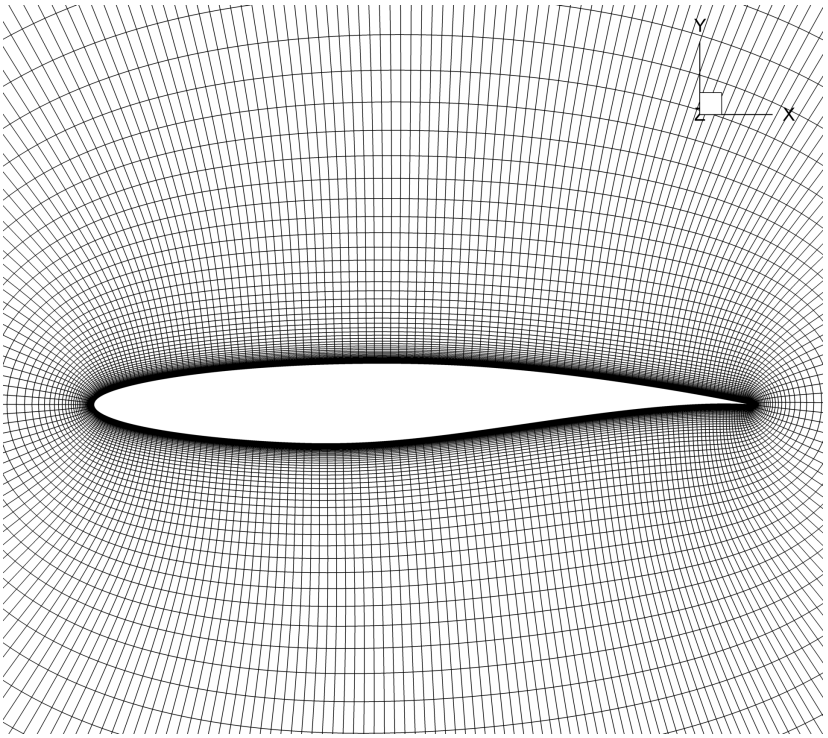


Figure 10: O2 mesh.

4.2.1 Problem Statement

The optimization problem statement for the RAE 2822 case is shown in Table 5. As before, the design variables are the vertical movements of the FFD control points, y. The number of control point design variables ranges from 8 to 52 (one half parametrizing the top and the other half parametrizing the bottom). The area of the airfoil, A, is constrained from exceeding the area of the baseline airfoil.

4.2.2 Optimization Results

The flow solver was set to converge to a density residual of 10^{-15}, and the convergence tolerance for the optimality in SNOPT was set to 10^{-6}. The step limit was set to 10^{-3}, and the Hessian update option was set to 50 in SNOPT. We tried different numbers of design variable, but only the more interesting results are shown here for conciseness. Adjoint solutions with frozen turbulence are often used in the literature because it is difficult to implement the computation of turbulence model partial derivatives; however, this results in lower accuracy gradients. We have the option to either solve the adjoint with frozen turbulence or solve the full adjoint. In this case, with a frozen turbulence adjoint, the optimizer stopped prematurely because of the inaccurate gradient information, so this option was turned off in the subsequent results.

Figure 11 shows the optimization results using the O-type mesh. Each column shows results for a given number of design variables in color, and all the other cases with different numbers of design variable are shown in gray for comparison. The change in the merit function is not visible in this linear plot above about 120 iterations.

Table 5: ADODG RAE 2822 case problem statement.

Category	Name	Quantity	Lower	Upper
Objective	C_d	1	–	–
Variables	y	N_y	-0.5	0.5
	α	1	1.0	5.0
Constraints	A	1	A_{initial}	–
	C_l	1	0.824	0.824
	C_m	1	-0.092	–
	$y_{\text{LE, lower}} = -y_{\text{LE, upper}}$	1	0.0	0.0
	$y_{\text{TE, lower}} = -y_{\text{TE, upper}}$	1	0.0	0.0
	t/c	400	10^{-4}	–

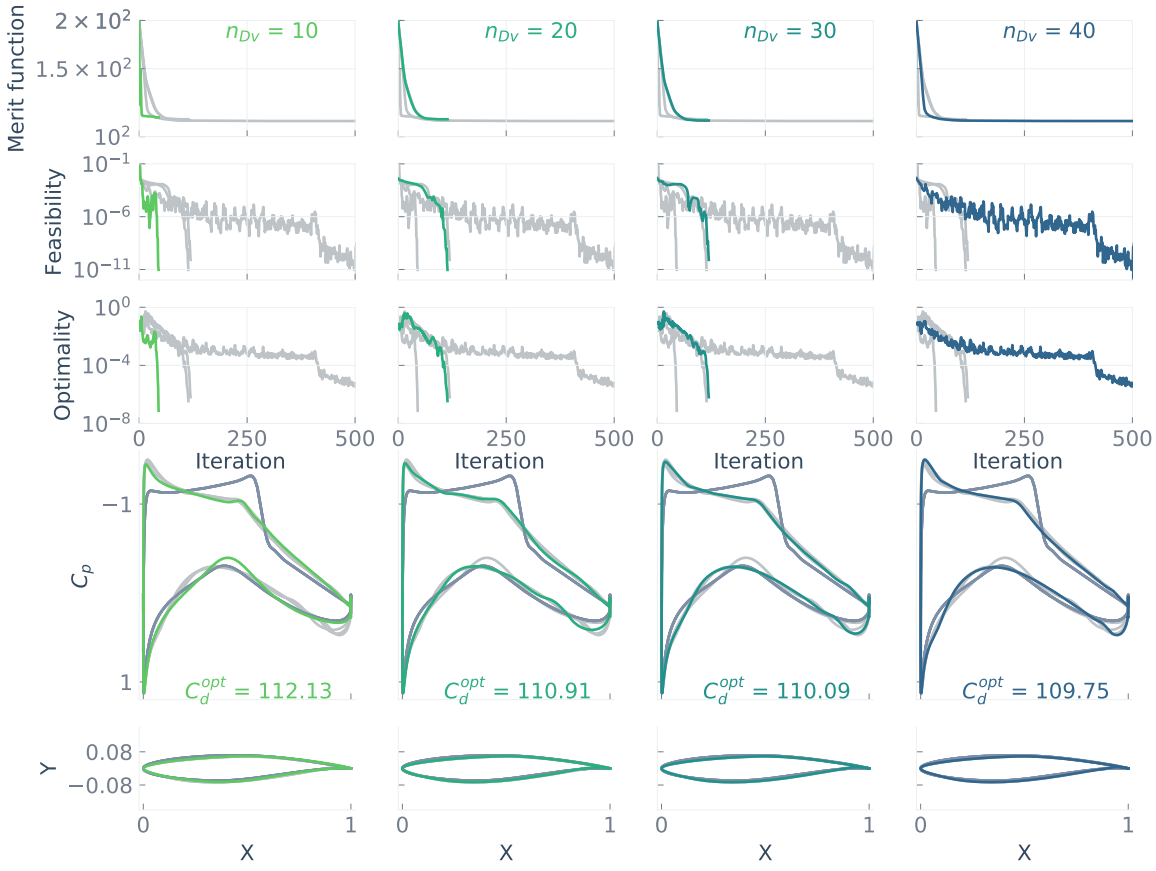


Figure 11: Merit function, feasibility, optimality history, optimized shape, and pressure distribution for the RAE 2822 case. The gray lines show the results from the other cases for comparison.

The constraint violation is within the 10^{-6} tolerance, which indicates optimization is well converged for all cases. Table 6 lists the optimal drag coefficient values for the various numbers of design variables, and shows that as the number of design variables increases, the drag values monotonically decrease, as also shown in Figure 12. This is

a desirable feature that is not necessarily observed for all parameterizations [20]. This trend can help us find the number of design variables that is adequate for a specific problem. In this case, with the O2 mesh, we achieved good results with 20 design variables. Increasing the number of design variables from 20 to 40 only reduced the drag by 1.16 counts.

In Table 7, we compare our results to those reported in other papers. Similarly to the NACA 0012 case, the results from various sources differ significantly. Again, we believe that the main reasons for these differences are the different flow solvers and meshes. Similarly to the results from the viscous transonic airfoil (VTA) workshop held in 1988 [49], the differences in the two-dimensional airfoil drag values show even larger variance than the three-dimensional Drag Prediction Workshop cases [44]. Note that when we increase the number of design variables to 160 using a C-type mesh, the optimization failed because the ratio of the number of control points to the number of surface mesh points is too large. This causes difficulties because the design variables get to control details in the shape that are not adequately resolved by the RANS flow solver. As a rule of thumb, we found that there should be at least four mesh points for every shape control point; otherwise, there is a risk that the optimizer will stop prematurely due to numerical difficulties.

Table 6: The drag decreases monotonically as the number of design variables increases.

N_y	Mesh	C_d (counts)
10	O2	112.13
20	O2	110.91
30	O2	110.09
40	O2	109.75
8	C4	119.91
10	C4	119.72
12	C4	119.33
16	C4	118.92
20	C4	118.70
28	C4	118.48
36	C4	118.37
44	C4	118.31
52	C4	118.28
60	C4	118.27
68	C4	118.23
76	C4	118.19
100	C4	118.11
160	C4	(Failed)

Figure 11 compares the baseline and optimized airfoil shapes and pressure distributions. We can see that the shock is eliminated and that, when there are more than 20 control point design variables, the difference among the results is small. The mesh convergence on the optimized airfoil for 40 design variables is shown in Table 4.

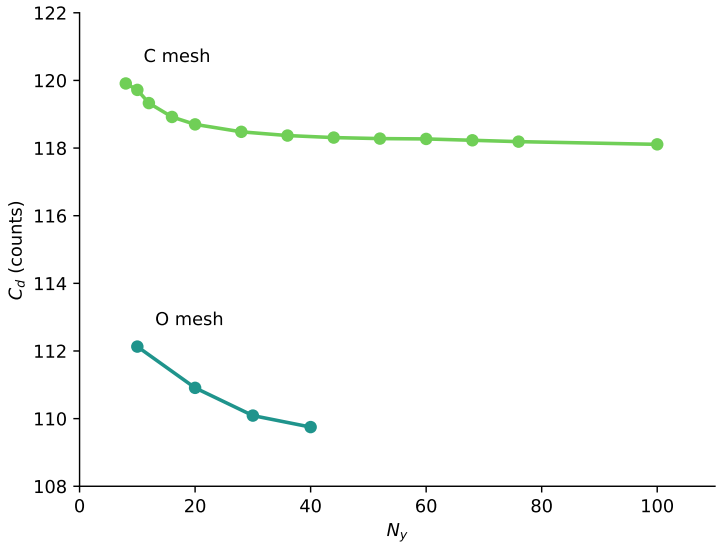


Figure 12: The optimum drag value decreases monotonically as number of design variables increases.

Table 7: Comparison of the drag for the RAE 2822 RANS optimization with results reported in the literature; a zero is added in the second decimal place when only one decimal place was reported.

Result	Mesh type	Cells	N_{DV}	RAE 2822	Optimized	Change
Present work	Structured	131,072	40	194.40	108.91	−85.49
Anderson et al. [15]	Cartesian	(Not reported)	14	196.00	124.00	−72.00
Bisson and Nadarajah [16]	Structured	3,264	16	177.80	102.30	−75.50
Carrier et al. [12]	Structured	7,894,172	10	189.20	103.90	−85.30
Gariepy et al. [48]	Structured	211,968	24	187.30	104.30	−83.00
Lee et al. [46]	Structured	47,824	17	234.44	131.81	−102.63
Poole et al. [47]	Structured	98,304	6	174.30	90.40	−83.90
Zhang et al. [19]	Structured	165,888	18	194.09	103.62	−90.47

4.2.3 Starting from Random Airfoils

We now explore how the optimizer performs when starting from different baseline airfoils. The problem formulations are the same as for the ADODG RAE 2822 case, except the initial airfoils are different.

Our hypothesis is that the design space for viscous airfoil design is unimodal and therefore, the final optimal airfoil should not depend on the initial design. Our previous studies on RANS-based, 3-D wing aerodynamic optimization with fixed planform showed that gradient-based optimization converged to essentially the same design when starting from different shapes, including some random shapes [2, 11]. While it is not possible to prove that the design space is unimodal in this complex case, one need only find two different optima to disprove the hypothesis. Although some researchers have argued that aerodynamic optimization problems may be multimodal [50, 51], no credible multiple optima have been presented. Here, we use our optimization tools on the ADODG airfoil to see if we can find multiple minima to disprove our hypothesis

for the 2-D case.

Based on the airfoil optimized from the RAE 2822 airfoil, we generate 10 airfoils using Latin hypercube sampling. Then, starting from these random airfoils and using 20 design variables, we perform airfoil optimizations with the same problem formulation. Figure 13 shows the 10 random airfoil shapes and the corresponding optimization histories. Every optimization converges to the same drag value as the reference optimization starting from the RAE 2822, which adds evidence that supports our hypothesis. The drag coefficient results from each optimization differ by only 10^{-6}, and the maximum difference between results for all design variables is lower than 2×10^{-7}.

4.3 Optimization Starting From a Circle

In the optimizations above, we started from the RAE 2822—as well as from random perturbations of the RAE 2822—to demonstrate the robustness of our aerodynamic shape optimization framework.

We now tackle an even bigger challenge: obtaining an optimal airfoil starting from a circle. This challenge might not be of interest for industrial applications because in practice, we know to start with at least a basic airfoil shape. However, we decided to tackle this challenge to demonstrate the software's ability to converge to the optimal airfoil starting from a drastically different shape—a "blank slate" of sorts. This not only demonstrates robustness in the numerical methods but also shows the effectiveness of the adaptive FFD parametrization approach. We use the same flow conditions and constraints as for the RAE 2822 case.

Because of the massive separation downstream of the circle, it is challenging to solve the flow in transonic conditions, but ANK is able to converge without much difficulty. We tackle these optimization problems with and without the adaptive FFD technique to show the effect of the adaptation. We generate a 32,768-cell O-mesh using pyHyp and construct a fixed FFD frame with 20 control points, as shown in Figure 14.

Figure 15 shows the evolution of FFD frame during the adaptive parametrization process. In the first step, we start with three control points along the chord. These three points accomplish a large thickness reduction in a few iterations. After the first adaption operation, three control points are added for a total of six to refine the shape near the leading and trailing edges. The optimization for this stage manages to achieve a supercritical airfoil shape. The second adaption operation adds more control points in the mid-chord position where the shock occurs, and the optimization refines the shape.

The optimization histories for the fixed and adaptive FFD are compared in Figure 16a. The optimization using ANK and the adaptive FFD converged successfully, requiring 184 iterations. The fixed FFD approach failed to converge in this case because of inaccurate gradients for the anomalous intermediate shape shown in Figure 4. In Figure 16b, we can see that the optimized shapes using fixed FFD and adaptive FFD are similar to each other, except for slight differences. The drag coefficient of optimized shapes starting from a circle with adaptive FFD and from the RAE 2822 with fixed FFD differ only by 0.51 counts.

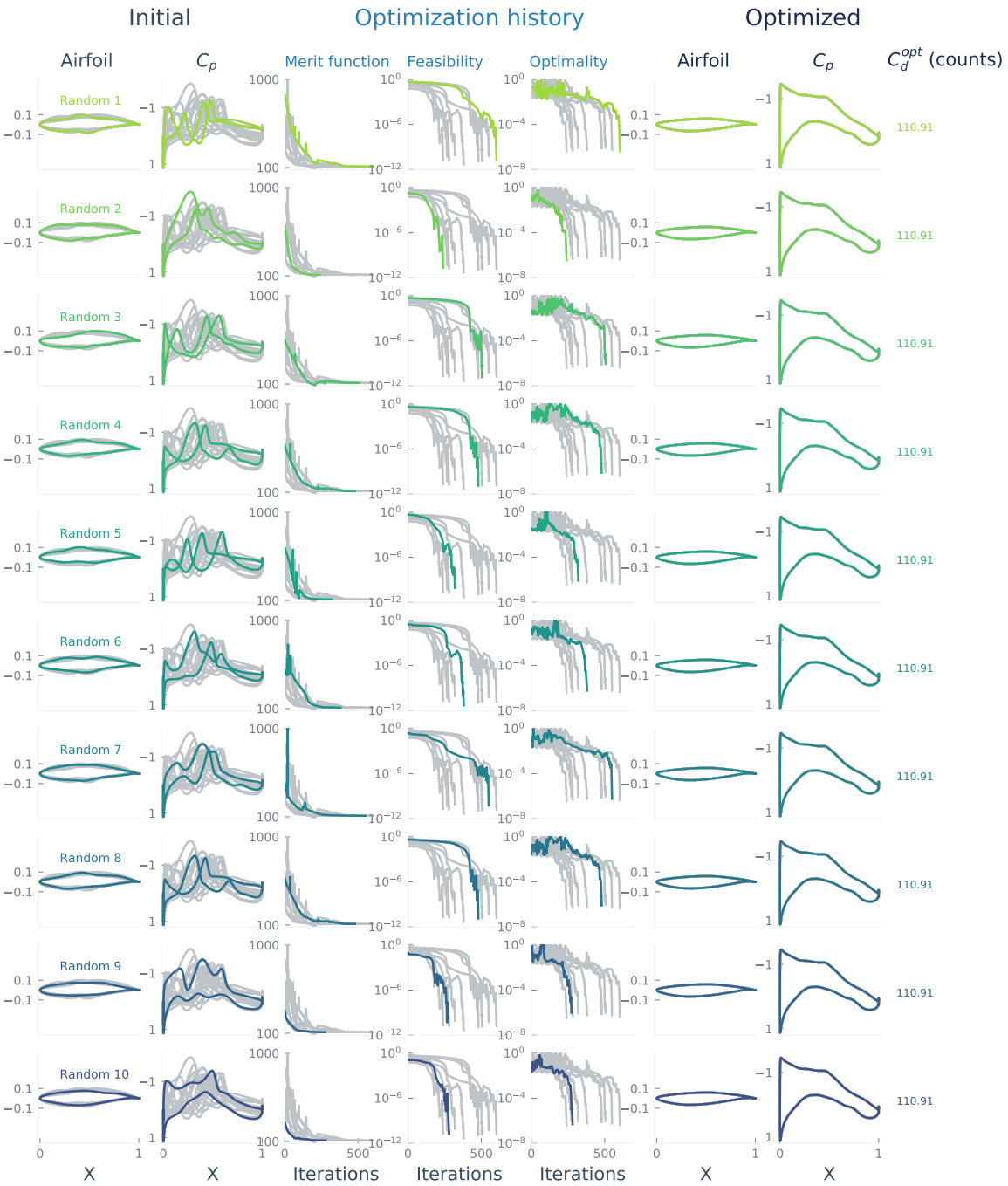


Figure 13: Optimization results when starting from random airfoils.

5 Conclusions

In this paper, we developed ways to address robustness issues in airfoil aerodynamic shape optimization and demonstrated their effectiveness by benchmarking our approach for the ADODG NACA 0012 and RAE 2822 optimization cases. We address

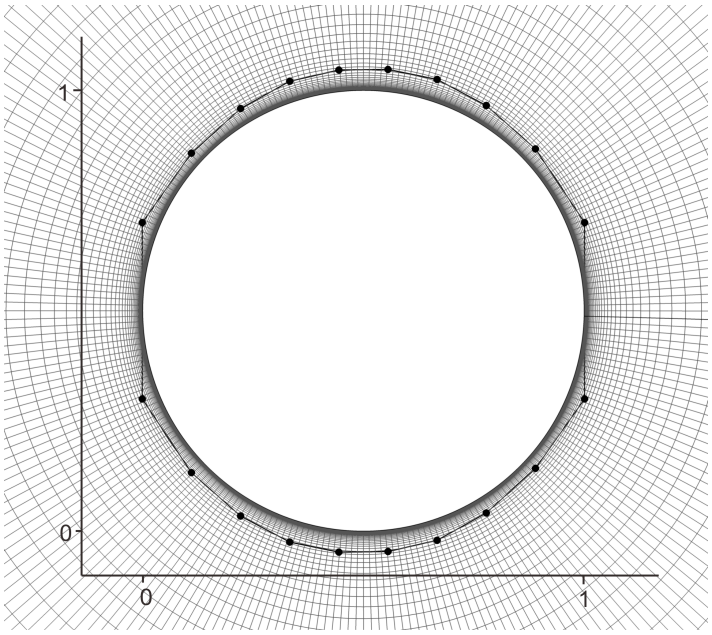


Figure 14: Mesh and FFD for the starting circle.

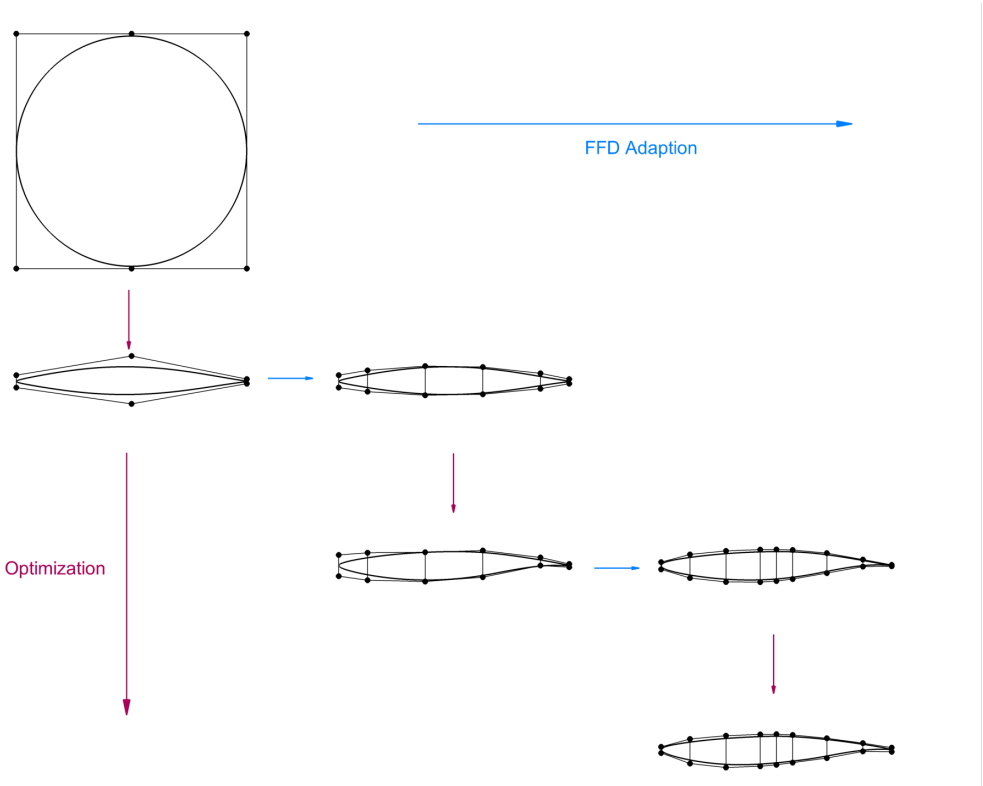
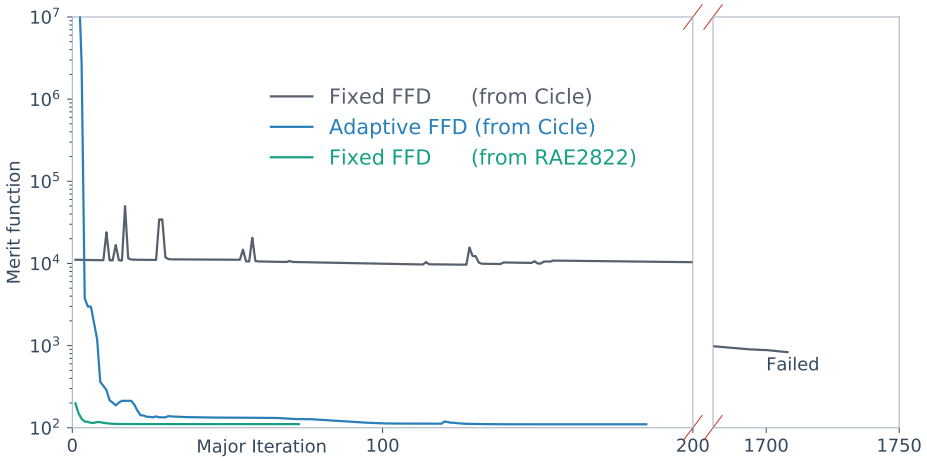
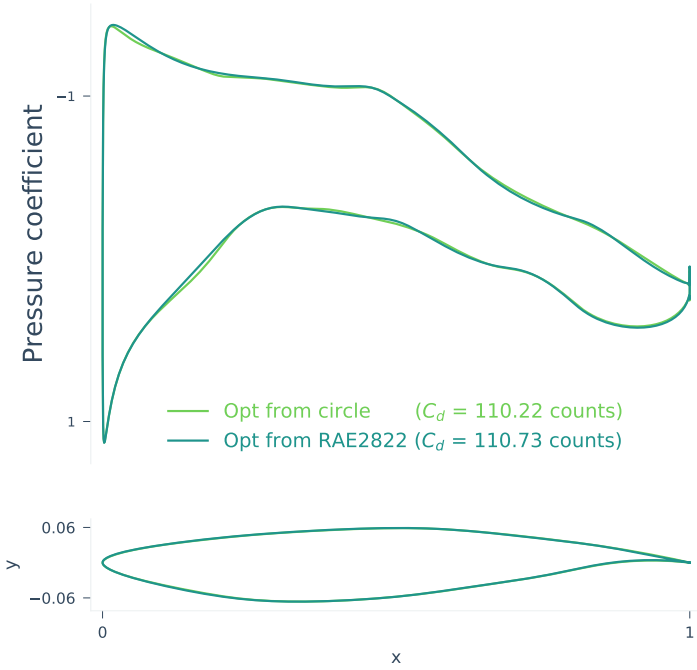


Figure 15: Evolution of the adaptive FFD for the optimization starting from a circle.

and discuss robustness from the point of view of the flow solver, the mesh deformation, and the design variable parametrization. For both ADODG airfoil cases, we obtain



(a) Merit function history comparison



(b) Airfoil shape and pressure coefficient comparison

Figure 16: The optimized shape using the adaptive FFD technique starting from a circle is close to the one starting from the RAE 2822, while the fixed FFD failed to converge.

well-converged results that are comparable to the results from previous work.

For the NACA 0012 Euler case, we obtained 7.6 counts for the optimum shape, which is the lowest reported in the literature so far. The difference in the optimal drag for the NACA 0012 is up to 23 counts, and the optimal airfoils show a much larger difference, ranging from 25 counts to 297.02 counts. Factors such as the flow solver, mesh size and type, parametrization, and design constraint handling can cause

the observed differences. We analyzed the non-unique solution issue due to the use of the Euler equations and found that the solution depends on how the flow field is initialized. However, the optimum shape based on Euler is not practical because it is not physically representative.

In the RAE 2822 case, we obtained a reduction of 88.66 drag counts, which is comparable to the drag reductions reported in previously published results. When compared with the finest mesh results from papers, the results show less scatter. The RAE 2822 case exposed issues related to gradient accuracy and flow solver failure caused by shock and separation interaction. We believe that the inclusion of the turbulence model in the adjoint-based gradient computation is necessary for well-converged optimization results. As for the complex flow phenomena that we may encounter during optimization, we rely on a robust flow solver to solve the governing equations. The robustness of the flow solver is one of the most important requirements for efficient and effective optimization. As for the effects of the number of design variables, we found that the FFD method yields a monotonic decrease in the minimum drag as the number of design variables increases, which is a good property.

We started optimizations from various initial shapes, including random perturbations of the baseline RAE 2822 shape, and all optimizations converged to the reference result that started from the RAE 2822 airfoil. This adds evidence towards the hypothesis that airfoil aerodynamic shape optimization problems based on the RANS equations are unimodal.

Not all the problems cited above could be solved by tuning the parameters of the flow solver and optimizer. To address some of the issues in these cases, we proposed an adaptive FFD for greater robustness in exploratory optimization.

As a final demonstration of the robustness and flexibility of our aerodynamic shape optimization framework and the adaptive FFD approach, we performed an optimization starting from a circle. The resulting airfoil was very similar to the reference case, and the differences were attributed to differences in the discretization.

These results indicate that the adaptive FFD approach enhances the robustness of the aerodynamic shape optimization and could be particularly useful in exploratory design optimization. Although the results presented herein still required a high level of expertise, we are closer than ever to a "push-button" solution for airfoil design optimization problems.

References

[1] T. H. Pulliam, M. Nemec, T. Holst, D. W. Zingg, Comparison of evolutionary (genetic) algorithm and adjoint methods for multi-objective viscous airfoil optimizations, in: Proceedings of the 41st AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 2003.

[2] Z. Lyu, G. K. W. Kenway, J. R. R. A. Martins, Aerodynamic shape optimization investigations of the Common Research Model wing benchmark, AIAA Journal 53 (2015) 968–985.

[3] J. E. V. Peter, R. P. Dwight, Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches, Computers and Fluids 39 (2010) 373–391.

[4] J. R. R. A. Martins, J. T. Hwang, Review and unification of methods for computing derivatives of multidisciplinary computational models, AIAA Journal 51 (2013) 2582–2599.

[5] J. R. R. A. Martins, P. Sturdza, J. J. Alonso, The complex-step derivative approximation, ACM Transactions on Mathematical Software 29 (2003) 245–262. September.

[6] A. Jameson, Aerodynamic design via control theory, Journal of Scientific Computing 3 (1988) 233–260.

[7] A. Jameson, Computational algorithms for aerodynamic analysis and design, Applied Numerical Mathematics 13 (1993) 383 –422.

[8] W. K. Anderson, V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, Computers and Fluids 28 (1999) 443–480.

[9] E. J. Nielsen, W. K. Anderson, Aerodynamic design optimization on unstructured meshes using the Navier–Stokes equations, AIAA Journal 37 (1999) 1411–1419.

[10] S. Chen, Z. Lyu, G. K. W. Kenway, J. R. R. A. Martins, Aerodynamic shape optimization of the Common Research Model wing-body-tail configuration, Journal of Aircraft 53 (2016) 276–293.

[11] Y. Yu, Z. Lyu, Z. Xu, J. R. R. A. Martins, On the influence of optimization algorithm and starting design on wing aerodynamic shape optimization, Aerospace Science and Technology 75 (2018) 183–199.

[12] G. Carrier, D. Destarac, A. Dumont, M. Méheut, I. S. E. Din, J. Peter, S. B. Khelil, J. Brezillon, M. Pestana, Gradient-based aerodynamic optimization with the elsA software, in: 52nd Aerospace Sciences Meeting, 2014. doi:10.2514/6.2014-0568.

[13] F. Bisson, S. Nadarajah, D. Shi-Dong, Adjoint-based aerodynamic optimization of benchmark problems, in: 52nd Aerospace Sciences Meeting, 2014.

[14] S. T. LeDoux, J. C. Vassberg, D. P. Young, S. Fugal, D. Kamenetskiy, W. P. Huffman, R. G. Melvin, M. F. Smith, Study based on the AIAA aerodynamic design optimization discussion group test cases, AIAA Journal 53 (2015) 1910–1935.

[15] G. R. Anderson, M. Nemec, M. J. Aftosmis, Aerodynamic shape optimization benchmarks with error control and automatic parameterization, in: 53rd AIAA Aerospace Sciences Meeting, 2015, p. 1719.

[16] F. Bisson, Nadarajah, Adjoint-based aerodynamic optimization of benchmark problems, in: 53nd Aerospace Sciences Meeting, 2015.

[17] D. A. Masters, N. J. Taylor, T. Rendall, C. B. Allen, D. J. Poole, A geometric comparison of aerofoil shape parameterisation methods, in: 54th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics, 2016. doi:10.2514/6.2016-0558.

[18] E. Fabiano, D. Mavriplis, Adjoint-based aerodynamic design on unstructured meshes, in: 54nd Aerospace Sciences Meeting, 2016.

[19] Y. Zhang, Z.-H. Han, L. Shi, W.-P. Song, Multi-round surrogate-based optimization for benchmark aerodynamic design problems, in: 54th AIAA Aerospace Sciences Meeting, 2016, p. 1545.

[20] D. A. Masters, D. J. Poole, N. J. Taylor, T. Rendall, C. B. Allen, Impact of shape parameterisation on aerodynamic optimisation of benchmark problems, in: 54rd AIAA Aerospace Sciences Meeting, 2016.

[21] T. W. Sederberg, S. R. Parry, Free-form deformation of solid geometric models, SIGGRAPH Comput. Graph. 20 (1986) 151–160.

[22] H. J. Lamousin, W. N. Waggenspack Jr, Nurbs-based free-form deformations, Computer Graphics and Applications, IEEE 14 (1994) 59–65.

[23] G. K. Kenway, G. J. Kennedy, J. R. R. A. Martins, A CAD-free approach to high-fidelity aerostructural optimization, in: Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, AIAA 2010-9231, Fort Worth, TX, 2010. doi:10.2514/6.2010-9231.

[24] D. Masters, N. Taylor, T. Rendall, C. Allen, Multilevel subdivision parameterization scheme for aerodynamic shape optimization, AIAA Journal (2017) 1–16.

[25] R. Duvigneau, Adaptive parameterization using free-form deformation for aerodynamic shape optimization, Ph.D. thesis, INRIA, 2006.

[26] X. Han, D. W. Zingg, An adaptive geometry parametrization for aerodynamic shape optimization, Optimization and Engineering 15 (2014) 69–91.

[27] G. R. Anderson, M. J. Aftosmis, Adaptive shape control for aerodynamic design, in: 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2015, p. 0398.

[28] J. S. Gray, C. A. Mader, G. K. W. Kenway, J. R. R. A. Martins, Modeling boundary layer ingestion using a coupled aeropropulsive analysis, Journal of Aircraft 55 (2018) 1191–1199.

[29] T. Dhert, T. Ashuri, J. R. R. A. Martins, Aerodynamic shape optimization of wind turbine blades using a Reynolds-averaged Navier–Stokes model and an adjoint method, Wind Energy 20 (2017) 909–926.

[30] N. Garg, G. K. W. Kenway, J. R. R. A. Martins, Y. L. Young, High-fidelity multipoint hydrostructural optimization of a 3-D hydrofoil, Journal of Fluids and Structures 71 (2017) 15–39.

[31] G. K. W. Kenway, J. R. R. A. Martins, Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration, Journal of Aircraft 51 (2014) 144–160.

[32] L. Piegl, W. Tiller, The NURBS book, Springer Science & Business Media, 2012.

[33] E. Luke, E. Collins, E. Blades, A fast mesh deformation method using explicit interpolation, Journal of Computational Physics 231 (2012) 586–601.

[34] G. K. W. Kenway, N. Secco, J. R. R. A. Martins, A. Mishra, K. Duraisamy, An efficient parallel overset method for aerodynamic shape optimization, in: Proceedings of the 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech Forum, Grapevine, TX, 2017. doi:10.2514/6.2017-0357.

[35] P. Spalart, S. Allmaras, A one-equation turbulence model for aerodynamic flows, in: 30th Aerospace Sciences Meeting and Exhibit, 1992. URL: http://dx.doi.org/10.2514/6.1992-439. doi:10.2514/6.1992-439.

[36] G. Klopfer, C. Hung, R. V. d. Wijngaart, J. Onufer, A diagonalized diagonal dominant alternating direction implicit (D3ADI) scheme and subiteration correction, in: 29th AIAA, Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 1998. doi:10.2514/6.1998-2824.

[37] A. Yildirim, G. K. W. Kenway, C. A. Mader, J. R. R. A. Martins, A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations, Journal of Computational Physics (2018). (Submitted).

[38] C. A. Mader, J. R. R. A. Martins, J. J. Alonso, E. van der Weide, ADjoint: An approach for the rapid development of discrete adjoint solvers, AIAA Journal 46 (2008) 863–873.

[39] Z. Lyu, G. K. Kenway, C. Paige, J. R. R. A. Martins, Automatic differentiation adjoint of the Reynolds-averaged Navier–Stokes equations with a turbulence model, in: 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, 2013. doi:10.2514/6.2013-2581.

[40] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, H. Zhang, PETSc Users Manual, Technical Report ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.

[41] P. E. Gill, W. Murray, M. A. Saunders, SNOPT: An SQP algorithm for large-scale constrained optimization, SIAM Journal of Optimization 12 (2002) 979–1006.

[42] R. E. Perez, P. W. Jansen, J. R. R. A. Martins, pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization, Structural and Multidisciplinary Optimization 45 (2012) 101–118.

[43] C. Kelley, D. Keyes, Convergence Analysis of Pseudo-Transient Continuation, SIAM Journal on Numerical Analysis 35 (1998) 508–523.

[44] E. N. Tinoco, O. Brodersen, S. Keye, K. Laflin, Summary of data from the sixth aiaa cfd drag prediction workshop: Crm cases 2 to 5, in: 55th AIAA Aerospace Sciences Meeting, 2017, p. 1208.

[45] J. Li, M. A. Bouhlel, J. R. R. A. Martins, Data-based approach for fast airfoil analysis and optimization, Journal of Aircraft 57 (2019) 581–596.

[46] C. Lee, D. Koo, K. Telidetzki, H. Buckley, H. Gagnon, D. W. Zingg, Aerodynamic shape optimization of benchmark problems using jetstream, in: Proceedings of the 53rd AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-0262.

[47] D. J. Poole, C. B. Allen, T. Rendall, Control point-based aerodynamic shape optimization applied to AIAA ADODG test cases, in: 53rd AIAA Aerospace Sciences Meeting, 2015, p. 1947.

[48] M. Gariepy, J.-Y. Trepanier, B. Malouin, C. Tribes, Direct search airfoil optimization using far-field drag decomposition results, in: 53rd AIAA Aerospace Sciences Meeting, 2015, p. 1720.

[49] T. L. Holst, Viscous transonic airfoil workshop compendium of results, Journal of Aircraft 25 (1988) 1073–1087.

[50] H. Namgoong, W. A. Crossley, A. S. Lyrintzis, Global optimization issues for transonic airfoil design, in: 9th AIAA/ISSMO Symposium and Exhibit on Multidisciplinary Analysis and Optimization, Atlanta, GA, 2002. doi:10.2514/6.2002-5641.

[51] M. S. Khurana, H. Winarto, A. K. Sinha, Airfoil optimization by swarm algorithm with mutation and artificial neural networks, in: AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, 2009.