AIAA SciTech Forum
8-12 January 2024, Orlando, FL
AIAA SCITECH 2024 Forum

10.2514/6.2024-1317

Analysis Driven Shape Design using Free-form Deformation of Parametric CAD Geometry

Marlena C. Gomez*, Marshall C. Galbraith†
Massachusetts Institute of Technology, Cambridge, MA, 02139

This paper presents a method for aerodynamic shape design driven by optimization which combines traditional CAD-like parametric solid model construction with free-form local deformation. The method involves placing a free-form deformation (FFD) box around an analytic geometry and converting any geometry inside the box not by default defined by B-splines to a B-spline representation. The control point net of the B-spline representation is then deformed with FFD. Control points of the FFD box serve as design variables during optimization, which allows for the generation of smooth geometry while keeping the number of degrees of freedom manageable during optimization. This approach is demonstrated in gradient-based optimizations of symmetric and asymmetric airfoils driven by the CFD analysis tool `MSES`.

I. Nomenclature

P_i	=	B-spline control point i
$\mathbf{P}$	=	vector of B-spline control points
C_D	=	two-dimensional drag coefficient
C_L	=	two-dimensional lift coefficient
A_i	=	area of a triangle element in an FFD grid
ε	=	a small number (i.e. 10^{-11})
m	=	indicator variable for normalized minimum triangular element area
$\eta(\Delta\mathbf{P})$	=	penalty function

II. Introduction

Researchers have extensively examined methods of aerodynamic shape design, leading to a number of advances in recent years. Advances like the adjoint sensitivity method for computing objective function gradients have decreased the computational cost of shape optimization driven by computational fluid dynamic (CFD) analysis [1]. These computational advances have enabled engineers to broaden high-fidelity optimization techniques from solving for optimal configurations of simple airfoil geometries to more complex components [2–5]. However, despite these advances, high-fidelity aerodynamic shape design tools have not been widely adopted by design engineers.

One challenge for aerodynamic shape designers is the depth of knowledge required to use the myriad of tool-sets involved in an optimization problem. Solving aerodynamic shape optimizations using CFD analysis involves a number of steps, including: geometry parameterization, surface and volume meshing, flow analysis, and the optimization itself [6]. Typically, each of these steps requires a separate tool-set [7]. An additional reason for the large number of tools involved in the shape optimization process stems from the different geometric modelers used. The aerospace design community tends to develop custom in-house discrete, mesh-based tools. Commercially available tools (CAD-based and CAD-free) are also used, but have a number of drawbacks. One drawback is, for proprietary reasons, CAD-based tools may hide geometric components necessary to compute values like parametric sensitivities. Since gradient-based optimization techniques require parametric sensitivities, when these are not provided, finite-differencing is used to approximate the values, which can be computationally costly and introduce inaccuracy [8, 9]. On the other hand, CAD-free tools have their own drawbacks. CAD-free tools lack meaningful design parameters, such as sweep or aspect ratio [10, 11]. Instead, design parameters are typically defined via deformative methods such as Free-form Deformation (FFD), which

*Graduate Student, Computer Science and Engineering.
†Research Engineer, Department of Aeronautics & Astronautics, AIAA Senior member.

allow for generic shape changes, but do not provide any further insight into the parameter space [12–14]. Furthermore, deformative methods generally result in a new discretization, not a new CAD model. The new discretization must be manually remodeled in CAD software, which increases the length of the design-and-test cycle and can introduce manual error [15].

Engineering SketchPad (ESP) is a feature-based and parametric solid modeler [16] designed to address many of the issues outlined above. ESP allows users to define intuitive design parameters and provides parametric sensitivities using a mixture of manually differentiation [17] and operator overloaded automatic differentiation, which is more efficient and robust compared to finite differencing [18]. The build process for a model involves the use of standard primitives solids, applied features, Boolean operators, transformations, and user-defined primitives and functions. While ESP enables gradient-based shape optimization using intuitive design parameters, some geometric features, such as a wing fairing, do not have “natural” or sufficiently flexible parameterizations for finding an optimal aerodynamic shape.

B-splines can be used to bridge parametric solid modeling with freeform shape design. Constructive solid modeling often results in B-spline surfaces and curves, and if not, the analytic geometry can be converted to a B-spline representation. Highly flexible and efficient in approximation [19], B-splines are especially useful when there is not a clear means of parameterizing the geometry [20]. The proposed methodology uses free-form deformation of B-spline geometry using the Engineering Geometry Aerospace Design System (EGADS) framework within ESP. This approach allows for a mixed parametric build using both traditional design parameters and free-form deformation control point locations. This maintains the intuitive connection to the designer with parameters that are part of the discipline’s lexicon while providing an additional suite of free-form parameters attached to complex surfaces in the geometric model.

This paper presents the methodology of using CFD analysis driven optimization using FFD to deform B-spline geometry. The remainder of this paper is organized as follows: Sections III and IV provide background on the FFD technique. Section V describes the implementation of the technique and the penalty function formulation used here in detail. Finally, Sections VII and VIII demonstrate the method on a number of CFD analysis driven optimization cases where the objective function minimizes drag of symmetric and asymmetric airfoils, respectively.

III. Parameterization via B-Spline Control Points

In previous research by the authors, geometry defined by B-splines was locally deformed and optimized with analysis by directly manipulating B-spline control points defining the geometry [21]. Although this work yielded promising results, there were also issues, including the large number of degrees of freedom. Generally, even small shapes defined by B-splines surfaces, like the rounded wingtip described in the paper, may be defined using a significant number of control points. With each control point serving as a design parameter, this results in more than an ideal number of degrees of freedom. Additionally, the design parameters need to be cleverly constrained in order to maintain valid shapes. Specifically, the control point locations were limited to only allow movement in the positive normal direction relative to the original geometry surface. This restriction prevents the generation of self-intersecting geometry, but also limits the possible shapes which can be generated by the design optimization process. Without this constraint, the optimization process regularly generates self-intersecting geometry. An example of this is shown in Fig. 1, which is a self-intersecting geometry generated for an optimization where each control point could move without restriction in the x, y, and z-directions.

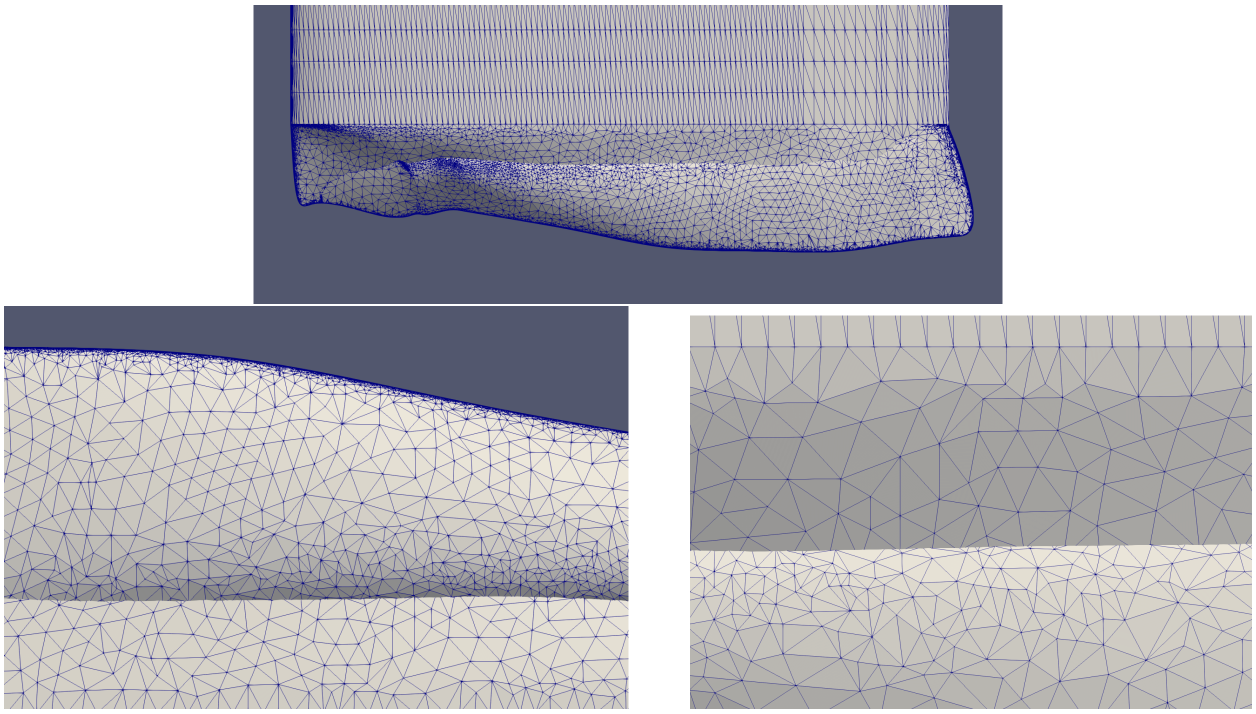


Fig. 1 Self-intersecting geometry resulting from an optimization of a wingtip surface where the B-spline x, y, and z control point locations served as independent design parameters.

Another issue observed was, during the optimization, geometry with undesirable "bumps" or "oscillations" would be generated. For example, Fig. 2 shows two intermediate shapes (generated via an optimization where CFD analysis is driving the B-spline surface deformation of a wingtip) with significant bumps. While the final wingtip surface from this optimization was smooth, many intermediate geometries exhibited these "bumps". This is an issue because it is difficult for the flow server to converge on geometries like this, if it does, and the solver will spend a significant amount of time analyzing this complicated shape. It is preferable to exclude geometries like this from the design space, so the solver is not wasting time trying to find a solution for a shape which is "obviously" not optimal.

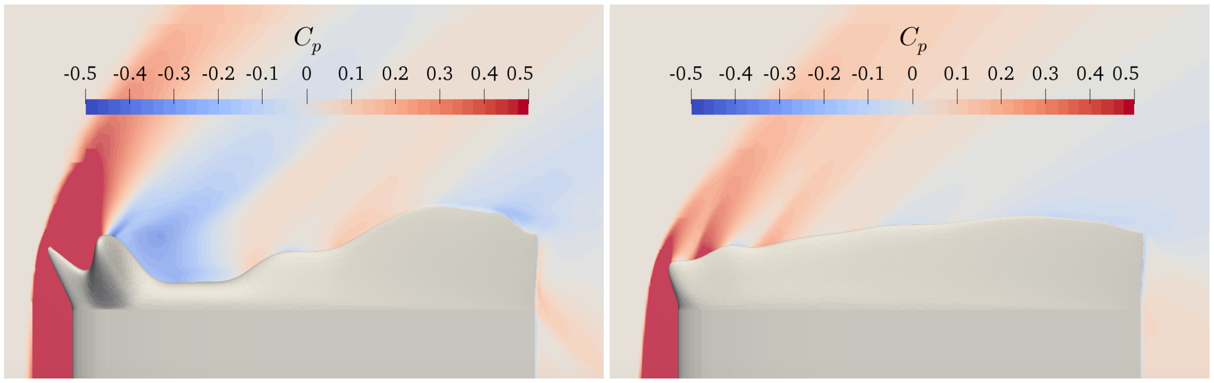


Fig. 2 Intermediate shapes in an optimization of a wingtip surface using the `Cart3D` design framework, where B-spline control points served as the design parameters.

IV. Free-form Deformation of B-spline Geometry

Free-form deformation provides a way to address the concerns above [11, 22]. FFD is a technique for deforming a solid geometric model by deforming the space of the model. This is accomplished by embedding the model in a FFD box defined by a tensor product trivariate B-spline (or Bernstein) polynomial [10] with a grid of control points, and then moving the trivariate B-spline control points to warp the space. The B-spline FFD box must be of a topologically higher dimension than the embedded geometry. For example, in order to morph a 1-dimensional curve in 2-dimensional Cartesian space, the FFD grid must be topologically 2-dimensional. FFD is extremely useful in that it preserves the connectivity and continuity of the underlying geometry when deforming it. Historically, this has been used frequently in aerodynamic shape design to morph mesh points in a CAD-free manor [9]. However, deforming a mesh in this way may cause significant distortion to the mesh elements, resulting in a need to reparameterize and remesh the geometry, which is a time-consuming process. Furthermore, the final shape is a discrete mesh, rather than a Boundary Representation (Brep) geometry which can be subsequently used for other analyses or manufacturing. In order to preserve the Brep geometry, the methodology in this paper applies to FFD to deform the B-spline control point net of B-spline surfaces instead of the points from a discrete mesh of the Brep surface. Since the geometry remains a Brep throughout the process with this approach, there is no need for the steps of reparameterizing or remeshing the geometry.

Using FFD to morph B-spline geometry has a number of advantages over using the geometric B-spline control points directly as the design parameters. Firstly, FFD gives the user explicit control over the number of design parameters, so the degrees of freedom of the optimization can be decreased or increased as desired. Secondly, FFD results in smoother geometry in comparison to the method of directly manipulating B-spline control points. For example, Fig. 3 shows a B-spline curve defining a NACA 0012 with a perturbation to one control point. The deformed airfoil has a distinct bump as a result of the manipulation to the B-spline.

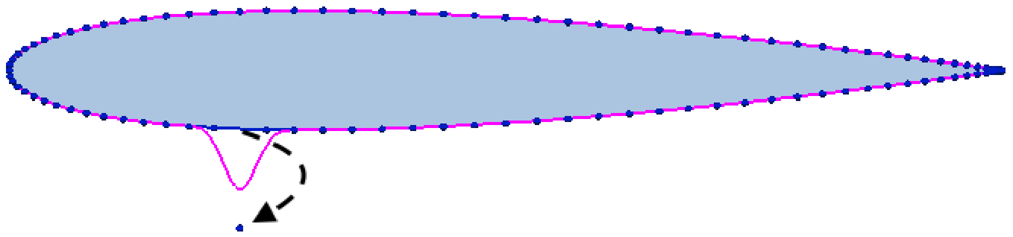


Fig. 3 Movement of one control point of a B-spline curve. The undeformed geometry is shown in blue, and the deformed geometry is shown in pink.

Perturbing one control point of the same airfoil with the FFD setup in Fig. 4, results in the geometry seen in Fig. 5, which does not have a distinct bump. Figure 5 also shows the effect of the degree of the B-splines defining the FFD grid on the deformation of the embedded geometry. When the FFD surface is defined by linear B-splines, the influence of one control point has a more local impact on the embedded geometry, shown in Fig. 5a, in comparison to the influence of one control point when the FFD surface is defined by B-splines of a higher degree, as shown with a cubic FFD surface in Fig. 5b. However, in both cases the deformed geometry is smooth when compared to direct B-spline manipulation. In addition to increasing and decreasing the degree of B-spline, the locality of influence of one control point is also affected by the number and placement of the control points.

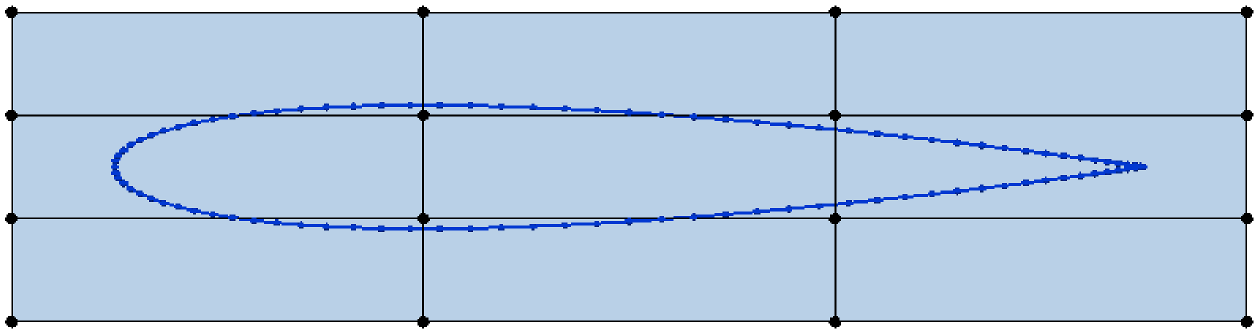
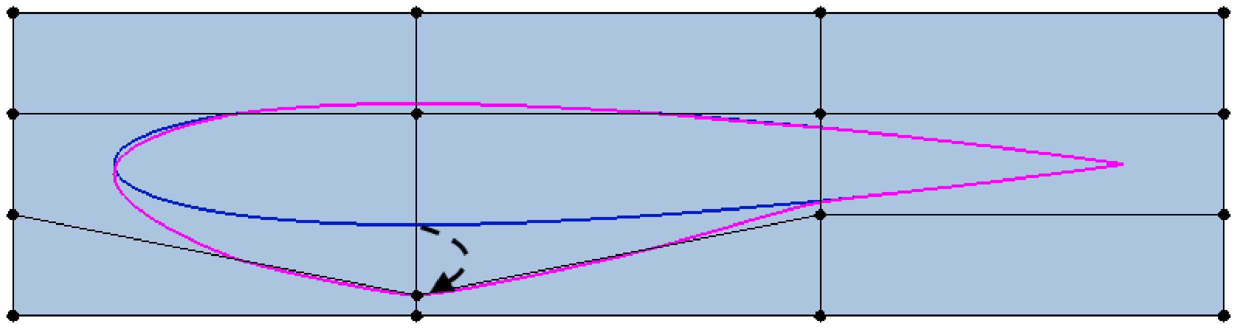


Fig. 4 FFD surface defined by a 4×4 control point net and an embedded B-spline curve defining a NACA 0012 airfoil.


(a) Linear FFD surface.

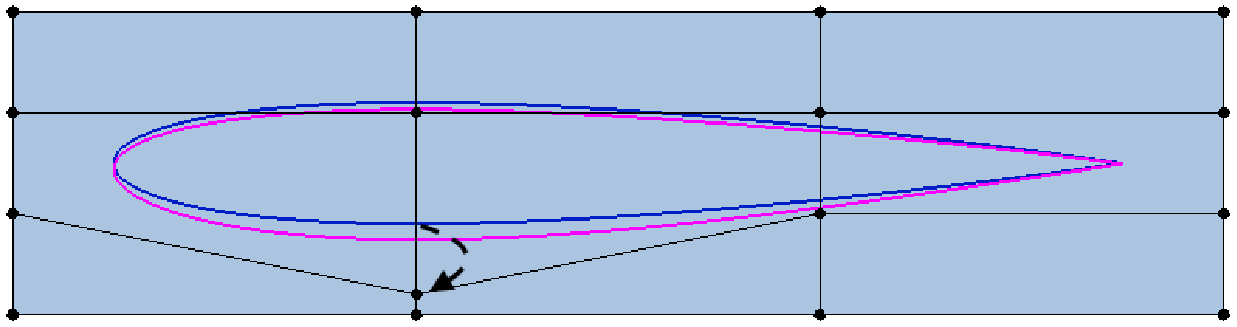

(b) Cubic FFD surface.

Fig. 5 Result of deforming one control point in an FFD surface. The undeformed geometry is shown in blue, and the deformed geometry is shown in pink.

FFD can create invalid, self-intersecting geometry if the control points move in a way which creates a negative Jacobian in the control point net. Figure 6 shows a NACA 0012 airfoil and 4×4 linear FFD grid where one control point has moved, creating a negative Jacobian and an invalid, self-intersecting curve in the embedded geometry. Restricting the FFD control point net to prevent any negative Jacobians ensures embedded geometry will not have any self-intersections [23]. That is, so long as the FFD control point net does not have any folds, the embedded geometry will also not have any folds.

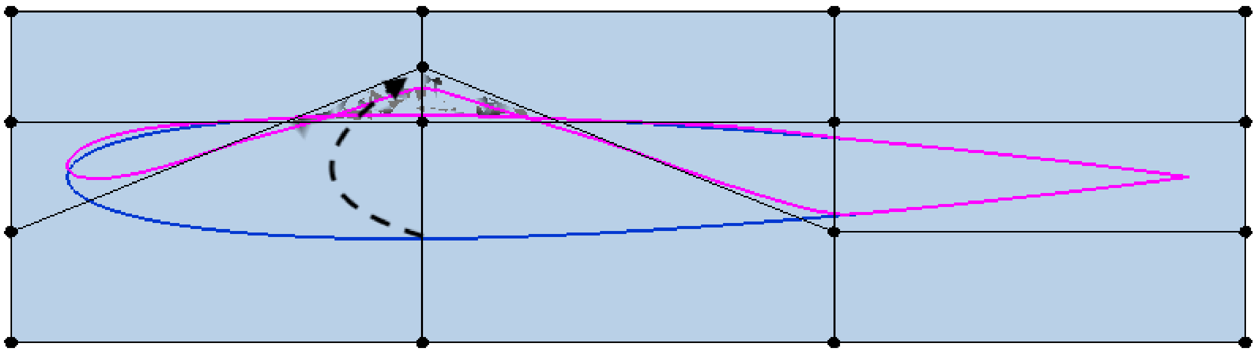


Fig. 6 A control point in the FFD grid moved such that there is a negative Jacobian in the FFD grid, and the deformed geometry is invalid.

V. B-spline Free-form Deformation Implementation

Geometry constructed in ESP will frequently contain B-spline curves and surfaces, however, if the Brep is not by default defined by B-splines, it is first converted to a B-spline representation prior to FFD. The free-form deformation of the B-spline geometry is executed in two steps. The first step maps the Cartesian coordinates of the embedded geometry to the parametric space of the FFD surface or volume, resulting in a set of Cartesian coordinate, (x, y, z), and parametric coordinate, (u, v, w), pairs. This mapping is accomplished using the Newton-Raphson method to solve the linear system of equations for the coordinate transformation. Since the parametric coordinates do not change during optimization, this computation occurs once before any deformation. In traditional aerodynamic optimization practice, the Cartesian coordinates are grid surface node locations. In the methodology presented here, the Cartesian coordinates are B-spline control point locations. The second step in FFD is deformation. After the FFD surface or volume is deformed by moving FFD control points, the new Cartesian coordinates, (x, y, z), of the embedded geometry are computed via a forward evaluation using the parametric coordinates, (u, v, w) of the manipulated FFD surface or volume. In order to ensure FFD does not result in self-intersecting geometry, the FFD control point net must not contain negative Jacobians.

A. Penalty Function

Restricting all FFD control points to not move outside the bounds of their neighboring control points prevents negative Jacobians in the control point net. One way to do this is ensuring all areas formed by an FFD control point and its neighbors always remain positive. For example, an interior control point in a 2-dimensional FFD grid forms four triangular elements with its neighbors, as shown in Fig. 7a. The area of these triangular elements can be computed using vectors, orienting the vectors appropriately so all areas are positive for the initial FFD control point net. During optimization, this interior control point may move, changing the areas of the triangular elements. If the point moves and all triangular element areas remain positive, as they are in Fig. 7b, then no penalty value should be introduced. However, using vectors to compute the area means the control point can move in a way which creates a triangle element with a negative area, as is the case shown in Fig. 7c. When this happens, a penalty value should be introduced. The penalty term should scale such that, when added to the objective function, the optimizer will avoid negative triangular areas. Additionally, the penalty term should scale such that larger negative areas are penalized more than smaller negative areas, driving the optimizer in the direction towards the situation where all triangular elements of the FFD grid have positive areas. All positive triangular element areas ensures all control points are within the bounds of their neighbors. Figure 7 shows an example of the triangular elements for the case where there is only one interior control point, but in a grid with more control points, the elements would overlap throughout.

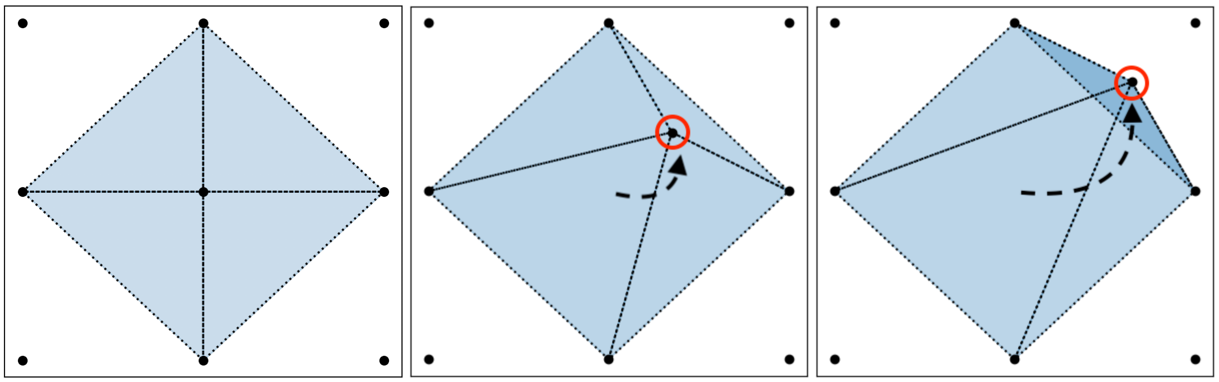

(a) Penalty = 0 **(b) Penalty = 0** **(c) Penalty > 0**

Fig. 7 Interior control point of an FFD surface.

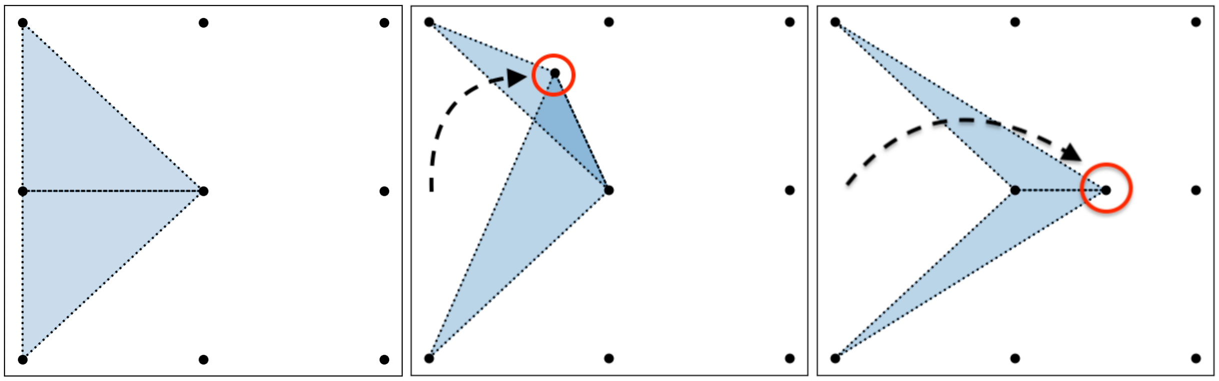

(a) Penalty = 0 (b) Penalty > 0 (c) Penalty > 0

Fig. 8 Boundary control point of an FFD surface.

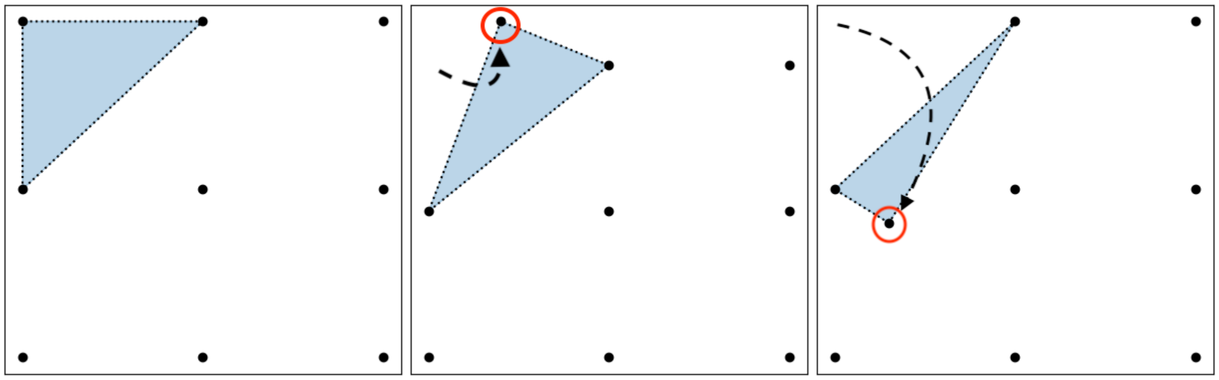

(a) Penalty = 0 (b) Penalty = 0 (c) Penalty > 0

Fig. 9 Corner control point of an FFD surface.

The minimum area of the neighboring triangle elements divided by the summation of all neighboring element areas serves as an indicator, m, for the penalty function. This indicator is positive when the minimum neighboring triangle element is positive, indicating the control point is within the bounds of its neighbors. Otherwise, m is negative, indicating the control point may not be within the bounds of its neighbors and a penalty value should be introduced. The indicator, m, normalizes to one for the initial FFD grid where all triangle elements have equivalent areas:

$$m = \begin{cases} 4 \cdot \frac{\min(\mathrm{A_i})}{\sum_{i=1}^{4} A_i} & \text{if interior control point} \\ 2 \cdot \frac{\min(\mathrm{A_i})}{\sum_{i=1}^{2} |A_i| + \varepsilon \sum_{i=1}^{2} A_i^0} & \text{if edge control point} \\ \frac{A_i}{A_i^0} & \text{else (corner control point)} \end{cases} \tag{1}$$

where A_i is the area of a neighboring triangular element, A_i^0 is the area of a triangular element in the un-deformed FFD surface, and ε is a small number (i.e. 10^{-11}). The ε term in the denominator serves to prevent division by zero.

For control points on the boundary of the FFD grid, only two triangle areas are influenced by the movement of the point, as opposed to the four elements in the interior case, as shown in Fig. 8. Hence, the indicator m only includes two triangles and becomes negative if one triangle area changes sign, as shown in Fig. 8b. However, it is possible for a boundary control point to move in a manner which results in both of the influenced triangular elements having negative areas, as shown in Fig. 8c. In order for the indicator to change sign in this situation, the denominator for boundary points is the sum of absolute values. Corner control points on the FFD grid only influence one neighboring triangular element, as shown in Fig. 9. The corner triangular element areas are divided by the corresponding corner triangular element area in the initial FFD grid.

The penalty function needs to be differentiable, smooth, and monotonic for gradient-based optimization. In addition, the penalty function needs to be zero when the indicator, m, is positive, and grow in magnitude when m becomes relatively close to or less than zero. This behavior can be accomplished with the complementary error function, shown in Fig. 10, which is a continuous, differentiable, switch-like function.

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \tag{2}$$

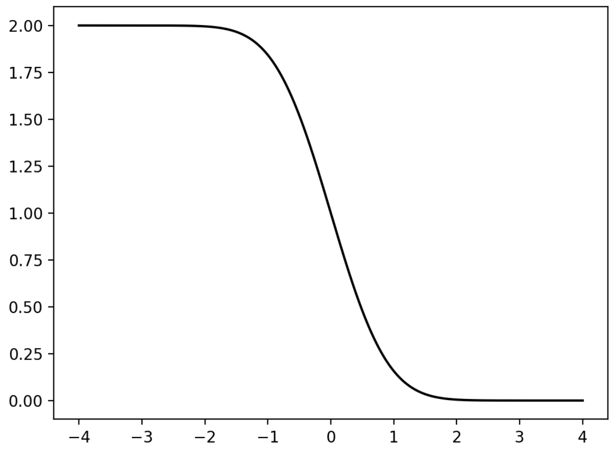


Fig. 10 Complementary error function (erfc).

The function for the indicator m and the complementary error function combined provides a framework for testing out a number of constants to appropriately scale the penalty so enough penalty value is introduced in comparison to the objective function. A number of possible penalty functions were considered and tested. Equation 3 is the penalty function formulated with the constants determined through testing to show the desired behavior. Figure 11 is a plot of the penalty value versus the value of the indicator, m. The penalty value approximates zero until the argument m begins approaching zero. A value of zero for m indicates the minimum triangular element area is zero, and in order to prevent this, the penalty function introduces a small value slightly before m is zero, which increases as m goes to zero and continues to increase as m becomes more negative. A linear growth of the penalty value when $m \leq -1$ worked best with the optimizations in testing.

$$\eta(m) = \mathrm{erfc}(10 \cdot \mathrm{m}) \cdot (\frac{1}{4} - \frac{1}{2} \cdot \mathrm{m}) \tag{3}$$

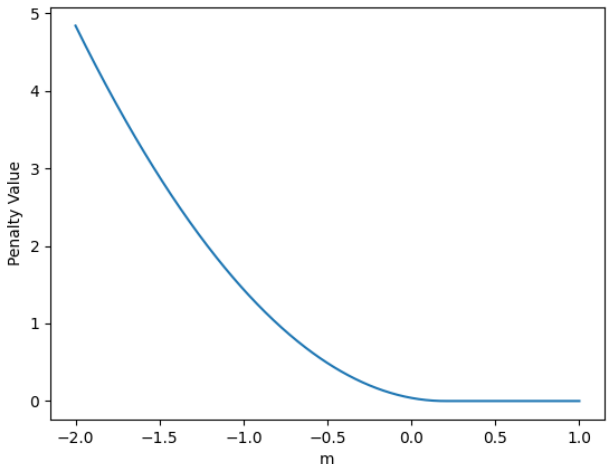


Fig. 11 Penalty function vs. indicator variable.

Figure 12 is a colormapping of the penalty function contour values when an interior control point in a 3×3 FFD grid is move and all other control points are fixed. The penalty function is non-zero when the minimum triangular element area is positive but close to zero, and grows as the minimum triangular element area becomes more negative. Unfortunately, this penalty function has one issue. The minimum function used in the computation for the indicator variable, m, in Eq. 1 is $C0$-continuous, so it is not differentiable. As seen in Fig. 12, the sharp corners in the contours indicate where the penalty function is $C0$-continuous.

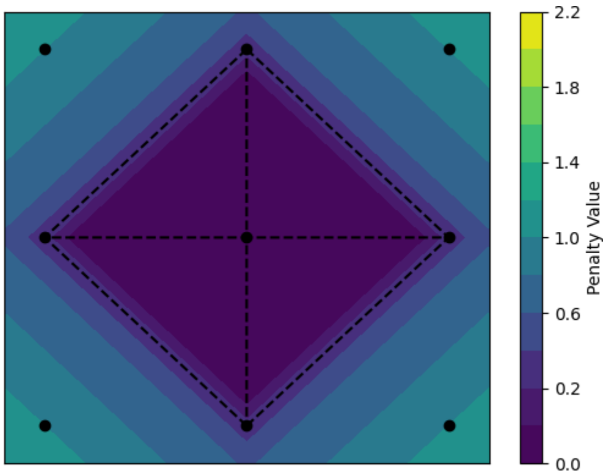
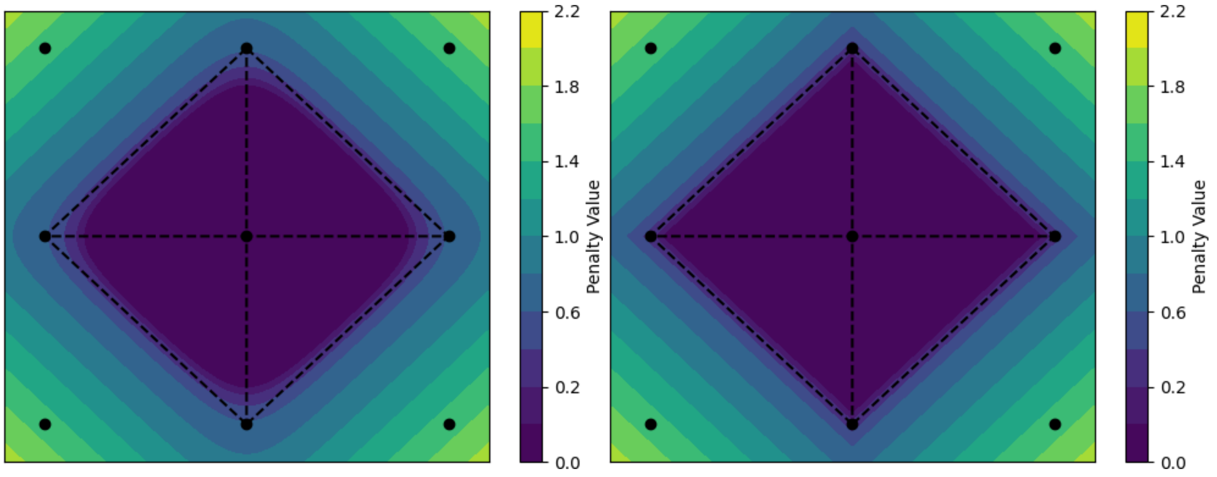


Fig. 12 Colormapping of the penalty function value when the interior control point in a 3×3 FFD surface is moved and all other control points are fixed.



(a) α = 10 **(b)** α = 1000

Fig. 13 Colormapping of the penalty function value when the smooth minimum function is used in replacement of the minimum function.

To resolve this issue, a smooth minimum, which is a differentiable approximation of the minimum function, replaces the minimum function:

$$\text{smoothmin}(\text{A}_0, \text{A}_1) = \begin{cases} A_0 - \frac{\log(1+\exp(-\alpha*(A_1-A_0)))}{\alpha} & \text{if } A_0 < A_1 \\ A_1 - \frac{\log(1+\exp(-\alpha*(A_0-A_1)))}{\alpha} & \text{else} \end{cases} \tag{4}$$

As the plot in Fig. 13a illustrates, there are no longer sharp corners in the contours, indicating the function is continuous and differentiable. The variable α in the smooth minimum function controls the approximation. A higher value of

α results in a closer approximation to the minimum function, as seen through the sharper corners in the contours in Fig. 13b. The smaller $\alpha = 10$ better demonstrates the diffentiability of the smooth minimum function, but any value of α is differentiable for this smooth minimum function. In all of the following optimizations in this work, the penalty function uses $\alpha = 1000$.

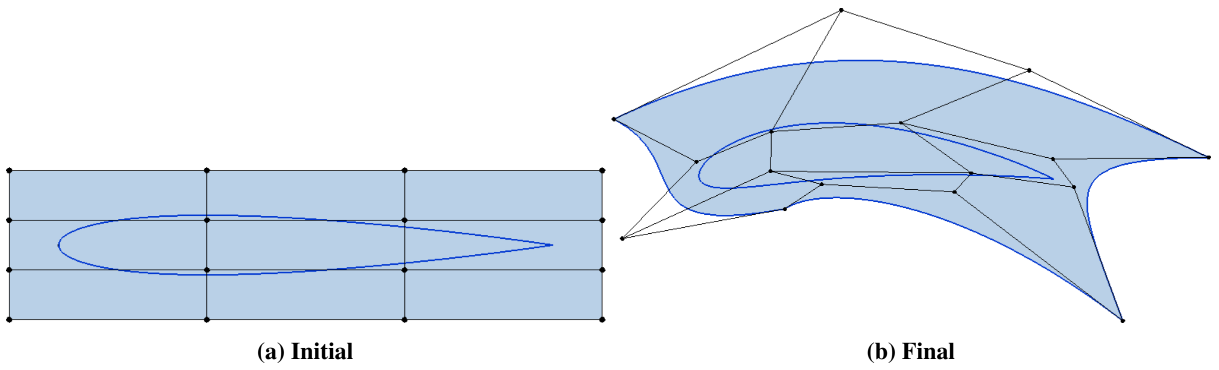
VI. Shape Design via L^2-norm Minimization

An L^2-norm error minimization optimization is performed to verify this method can accurately match a target shape and the formulated penalty function serves to prevent the generation of self-intersecting geometry. In this optimization, a B-spline defining a NACA 0012 geometry is morphed using this method to match a target NACA 8416 airfoil geometry. To define the airfoil geometries, cubic B-splines are fit using the cosine distribution of parameter values for the NACA 4-series equations. The last coefficient of the NACA 4-series equations is changed from −0.1015 to −0.1036 to produce airfoils with a sharp trailing edge [24]. The airfoil geometries are generated using `ESP` and are shown in Fig. 14. The objective function is the addition of the penalty function in Eq. 3 and the L^2-norm error between the morphed and target airfoil shapes. A rectangular cubic B-spline FFD surface encompassing the space of the airfoil is created, and perturbations to the FFD control points serve as the design parameters in the optimization. The nonlinear optimization library, `NLopt` [25], is used with the gradient-based Low-store BFGS (Broyden–Fletcher–Goldfarb–Shanno) [26, 27] algorithm to perform the optimization.

(a) NACA 0012 **(b) NACA 8416**

Fig. 14 Initial and target airfoil geometries for the L^2-norm difference minimization.

Figure 15 shows the results of an optimization with a 4 by 4 control point net for the FFD surface. This results in $4 \times 4 \times 2 = 32$ degrees of freedom for the optimization, a significant reduction compared to using B-spline control points directly as design parameters. The B-spline curve defining the airfoil geometry has 101 control points, which would results in $101 \times 2 = 202$ degrees of freedom for an optimization with B-spline control points serving as design parameters [28].


(a) Initial **(b) Final**

X-range	[-0.1, 1.1]
Y-range	[-0.15, 0.15]
Degrees of Freedom	32
Number of Iterations	244
Best Objective Function Value	4.0×10^{-7}

(c) Summary of the optimization

Fig. 15 Summary of a 4×4 cubic FFD grid L^2-norm difference optimization. The FFD grid surface geometry is blue and the control point locations are black points.

The optimization terminates after 244 objective function iterations, with a final objective function value of 4.0×10^{-7}. The optimizer termination occurs when the difference between the value of subsequent objective function evaluations is $\leq 1 \times 10^{-17}$. Despite a small number of degrees of freedom in the optimization, the L^2-norm difference between the morphing and target airfoil shapes is remarkably small, verifying this methodology can be successfully used to accurately match a target shape in a gradient-based optimization.

VII. Drag Minimization of a Symmetric Airfoil

After verifying using control points of B-splines as design parameters for optimization could match a known target shape well, the technique is extended to optimization with CFD analysis. To demonstrate this method of aerodynamic shape design with analysis, an optimization similar to the case in Section II.C in [21] is performed. The initial shape of the optimization is a NACA 0024 airfoil, and the objective function minimizes the sum of the drag and penalty function in Eq. 3, with a minimum constraint on area as half that of the initial airfoil area. The optimization statement is written as:

$$\begin{aligned} \underset{\Delta \mathbf{P}}{\text{minimize}} \quad & C_D + \eta(\Delta \mathbf{P}) \\ \text{subject to} \quad & A_{\text{airfoil}} \geq \frac{1}{2} \cdot A_{\text{NACA0024}} \end{aligned} \tag{5}$$

where $\Delta \mathbf{P}$ represents the set of control point perturbations of the FFD surface.

The initial NACA 0024 airfoil is generated in `ESP` along with an FFD box. The pyCAPS interface to `MSES` in `ESP` is used to define and execute the CFD analysis of the airfoil [29]. The analysis is executed with an angle-of-attack of zero degrees, a Mach number of 0.7, and a Reynolds number of 5×10^6. A number of Multi-Disciplinary Analysis and Optimization (MDAO) frameworks can directly interface with pyCAPS to execute optimizations. In this paper, the open-source framework OpenMDAO [30] is used with the gradient-based Sequential Least Squares Programming Optimize (SLSQP) which ships with OpenMDAO. The optimization objective convergence tolerance is set to 1×10^{-7} [31]. The gradients of C_D, the penalty function, and the airfoil area with respect to each design variable are verified using finite differencing, as shown in Appendix A in Ref. [28].

For the optimization, a 5×5 FFD grid defined in the Cartesian range $[-0.05, 1.05] \times [-0.16, 0.16]$ is used. To maintain symmetry, the central row of control points is kept fixed, and, as in Section II.C in [21], the control points defining the lower half of the airfoil are mirrors of the upper half. The 10 control points defining the upper two rows of the FFD grid are thus the design parameters in the optimization. To ensure the chord length of the airfoil does not change, only the y-component of the control points serve as free parameters. This results in 10 degrees of freedom.

For a linear FFD grid using this set-up, the optimization terminates after 50 design iterations and the objective function value reduces from 0.032 to approximately 0.008. The area of the airfoil reduces from approximately 0.16 to 0.08, which satisfies the constraint placed on the minimum area of the airfoil. Figure 19 shows the initial and best airfoil shapes with the associated FFD grid surfaces and control points in black, as well as a plot of the objective function history and table summarizing the optimization.

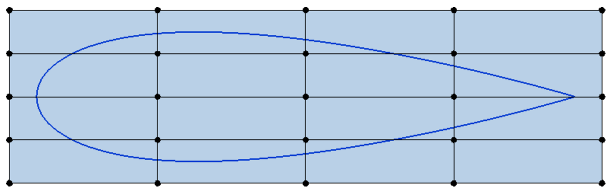


Fig. 16 Initial NACA 0024

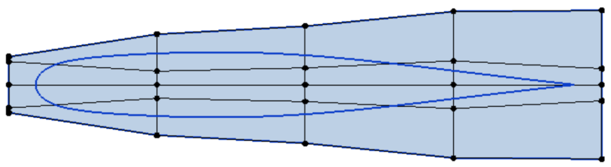


Fig. 17 Best airfoil from the optimization.

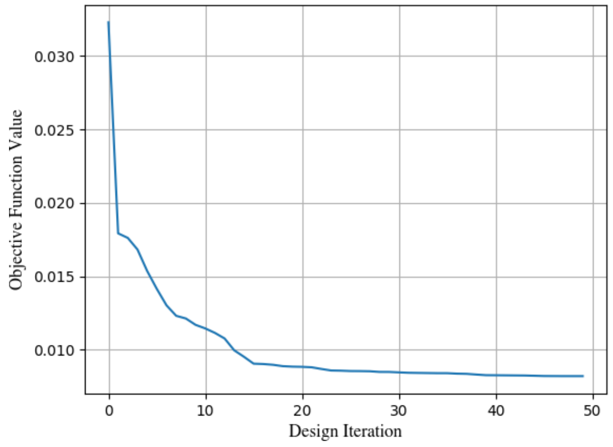


Fig. 18 Object Function Convergence History

X-range	[-0.05, 1.05]
Y-range	[-0.16, 0.16]
Degrees of Freedom	10
Number of Iterations	50
Best Objective Function Value	8.2×10^{-3}

(a) Summary of the optimization

Fig. 19 A drag minimization of a NACA 0024 with a 5×5 linear FFD grid.

The same case above is executed with an FFD grid defined by cubic B-splines instead of linear. The summary of this optimization is shown in Fig. 23. The same objective function value, 0.008, is achieved after 50 design iterations. Examining the objective function convergence of the cubic case in comparison to the previous linear case shows it took fewer design iterations for the cubic case to converge. By around design iteration 15, the cubic case has nearly reached the best value of 0.008, whereas the linear case takes closer to 25 design iteration to reach this value. Since linear splines have more local influence over the shape than cubic splines, optimizations with linear splines may converge more slowly than optimizations with cubic splines.

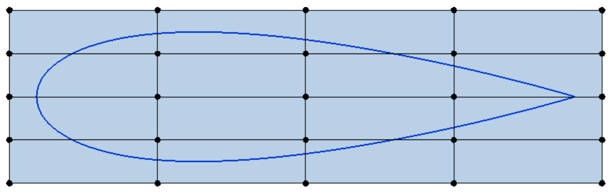


Fig. 20 Initial NACA 0024

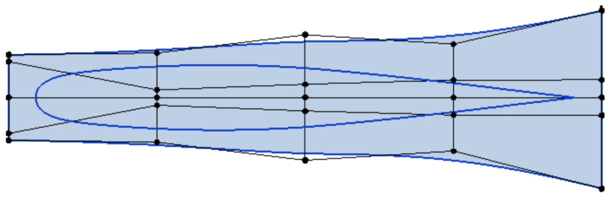
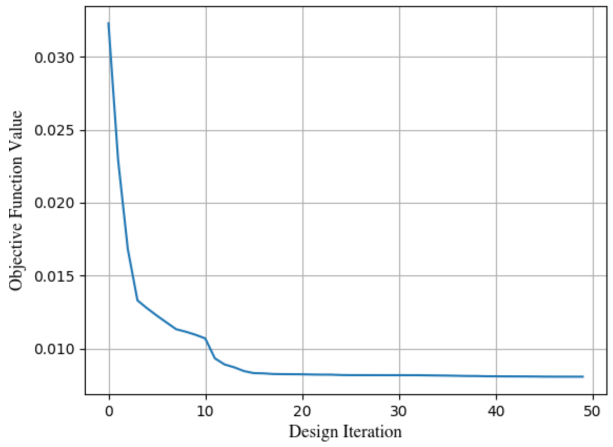


Fig. 21 Final



X-range	[-0.05, 1.05]
Y-range	[-0.16, 0.16]
Degrees of Freedom	10
Number of Iterations	50
Best Objective Function Value	8.0×10^{-3}

(a) Summary of the optimization

Fig. 22 Object Function Convergence History

Fig. 23 A drag minimization of a NACA 0024 with a 5×5 cubic FFD grid.

The same case is executed a final time, with an increased quartic degree of the FFD box splines. With quartic B-splines, the optimization converged to the objective function value of 0.008 after 28 design iterations. The smaller number of design iterations to converge could again be a result of a higher degree of B-spline for the FFD box resulting in less local influence for the control points. Figure 27 summarizes the results from this optimization.

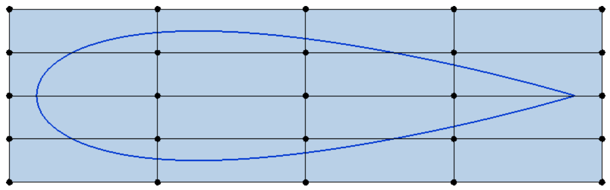


Fig. 24 Initial NACA 0024

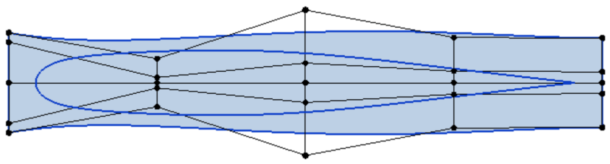


Fig. 25 Final

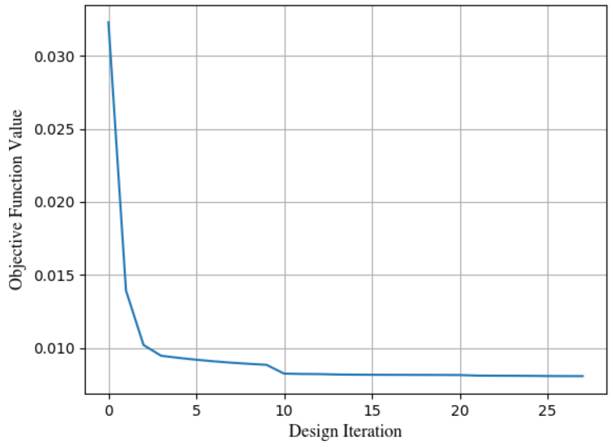


Fig. 26 Object Function Convergence History

X-range	[-0.05, 1.05]
Y-range	[-0.16, 0.16]
Degrees of Freedom	10
Number of Iterations	28
Best Objective Function Value	8.0×10^{-3}

(a) Summary of the optimization

Fig. 27 A drag minimization of a NACA 0024 with a 5×5 quartic FFD grid.

VIII. Drag Minimization of an Asymmetric Airfoil

Another set of MSES design cases introduces camber to the initial airfoil shape, with the goal of having the analysis drive the initially asymmetric airfoil to a symmetric airfoil shape. This case uses a Kulfan airfoil, where the geometry is defined by a Class function and Class Shape Transformation (CST) [32]. The Kulfan CST function can be modified with n coefficients for the upper surface and n coefficients for the lower surface to produce any round nose, sharp aft-end airfoil. For this case, 4 coefficients are used for the upper and lower surface each. The coefficients used are in Appendix B.4 in Ref. [28], along with the full ESP script for generating the Kulfan airfoil and a 5×5 FFD grid. The FFD grid is defined in the Cartesian range $[-0.001, 1.0015] \times [-0.14, 0.14]$. Both the first and last columns of the FFD control point grid are only free to move in the y-direction during the optimization, so the chord length of the airfoil is kept at one. Furthermore, the central control point in both of these columns is excluded from the set of design parameters. These two control points are fixed so the nose and tail of the airfoil do not change position during the optimization. Both the x and y-components of the remaining FFD control points are design parameters in the optimization. The analysis is done with MSES at an angle-of-attack of zero degrees, a Mach number of 0.2, and a Reynolds number of 5×10^6. As in the previous cases, OpenMDAO and the SLSQP algorithm are used for the optimization. The area of the airfoil is constrained to a minimum of half of the initial airfoil area. Additionally, an equivalence constraint on lift, $C_L = 0$, is introduced. To satisfy this constraint, the optimization must produce a symmetric airfoil. The optimization statement is written as:

$$\begin{aligned} \underset{\Delta\mathbf{P}}{\text{minimize}} \quad & C_D + \eta(\Delta\mathbf{P}) \\ \text{subject to} \quad & A_{\text{airfoil}} \geq \frac{1}{2} \cdot A_{\text{initial}} \\ & C_L = 0 \end{aligned} \tag{6}$$

In the first case with the Kulfan airfoil, a cubic 5×5 FFD grid is used. This results in $4 + 4 + 2 \cdot 3 \cdot 5 = 38$ degrees of freedom. After 50 design iterations, the optimization reduces the objective function value from 0.010 to 0.0075. The area of the airfoil is reduced from approximately 0.156 to 0.078, satisfying the constraint on airfoil area. The coefficient of lift, initially 0.213, is 0.001 in the best shape from the optimization. A summary of the optimization is shown in Fig. 33.

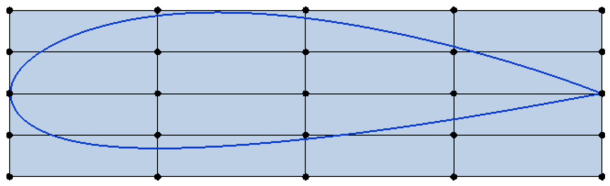


Fig. 28 Initial Kulfan Airfoil

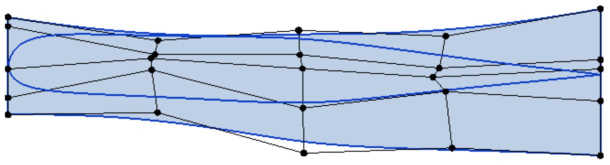


Fig. 29 Final

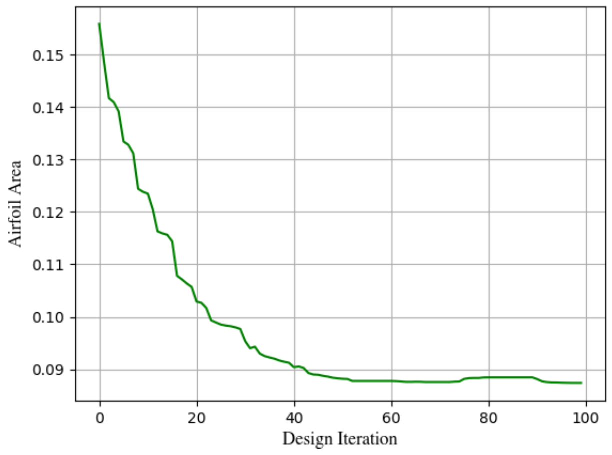


Fig. 30 Airfoil Area History

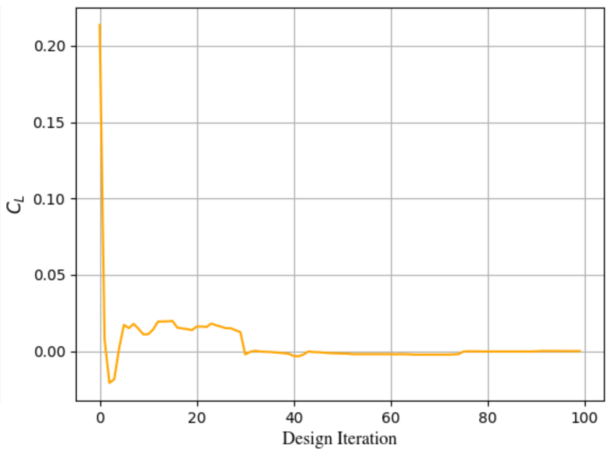


Fig. 31 C_L History

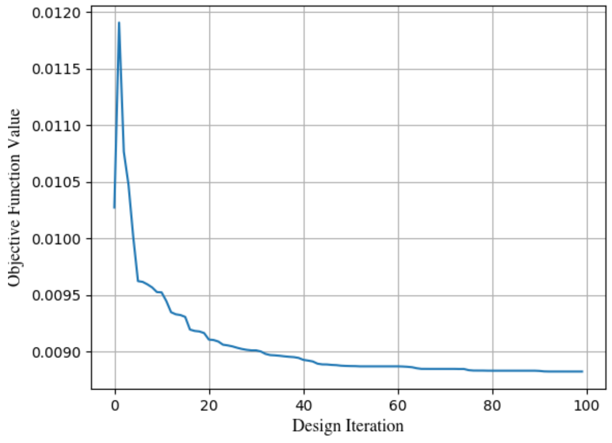


Fig. 32 Object Function Convergence History

X-range	[-0.001, 1.0015]
Y-range	[-0.14, 0.14]
Degrees of Freedom	38
Number of Iterations	50
Best Objective Function Value	7.5×10^{-3}

(a) Summary of the optimization

Fig. 33 A drag minimization of a Kulfan airfoil with a 5×5 cubic FFD grid.

In a second design optimization using the Kulfan airfoil, the same set-up is used with more control points. The FFD grid used in this case is cubic and has a 7×5 control point net. This results in $4 + 4 + 5 \times 5 \times 2 = 58$ degrees of freedom. After 50 design iterations, the optimization achieved the same objective function value as in the 5×5 FFD grid case, 7.5×10^{-3}. Figure 39 summarizes the results from this case. Since there are more degrees of freedom for the 7×5 FFD grid in comparison to the 5×5 FFD grid, it is expected the optimizer converges more slowly. This is illustrated by comparing the convergence history for the objective function, shown in Fig. 39 for this case, with the previous case shown in Fig. 33.

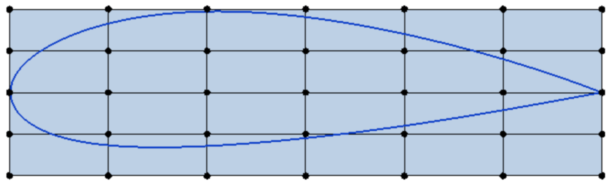


Fig. 34 **Initial Kulfan Airfoil**

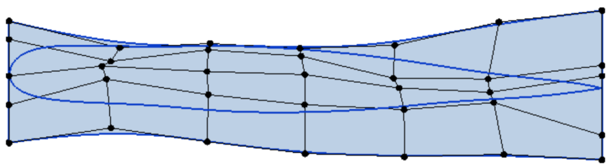


Fig. 35 Final

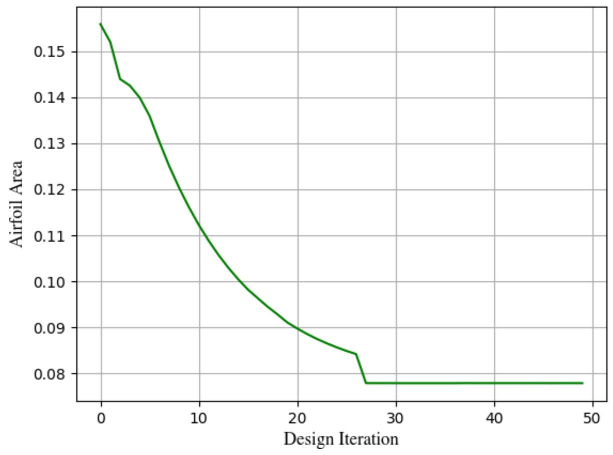


Fig. 36 Airfoil Area History

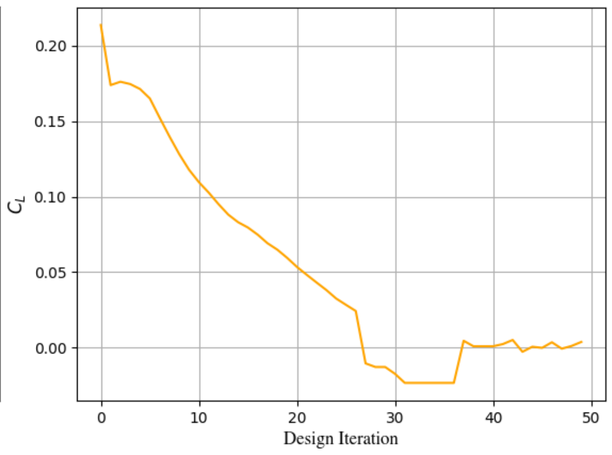


Fig. 37 C_L History

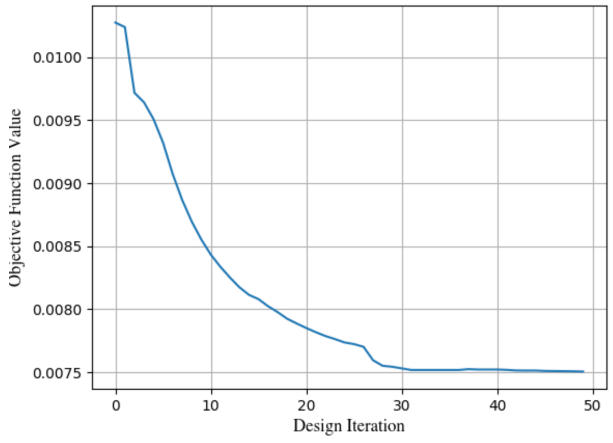


Fig. 38 Object Function Convergence History

X-range	[-0.001, 1.0015]
Y-range	[-0.14, 0.14]
Degrees of Freedom	58
Number of Iterations	50
Best Objective Function Value	7.5×10^{-3}

(a) Summary of the optimization

Fig. 39 A drag minimization of a Kulfan airfoil with a 7×5 cubic FFD grid.

The best airfoil shape from this optimization is approximately symmetric. The airfoil tail, originally at $[1, 0]$ shifted in both cases, despite the restrictions on control point movement for the last column of the FFD grid control point net. As seen in Fig. 35, the tail of the airfoil shifted downward during the optimization. As a result, the C_L of the best airfoil is approximately 0.001, and not exactly zero.

IX. Conclusion

A method for aerodynamic design driven by optimization using free-form deformation to morph Brep geometry is presented. In this method, the control point net of a B-spline representation of a geometry is deformed via free-form deformation. Self-intersections of geometry are prevented by a penalty function, which is formulated such that self-intersect in the FFD grid control point net are avoided, thereby preventing self-intersections in geometry embedded in the FFD grid. The control points of the free-form deformation grid serve as design parameters for optimization. Using this method results in smoother shapes throughout the design process and allows for more control over the number of design parameters in the optimization in comparison to direct B-spline manipulation. The method uses the geometry generation software in ESP to compute parametric sensitivities necessary for gradient-based optimization. Optimization is performed using the pyCAPS framework in ESP with MSES for analysis. The method is demonstrated on airfoil optimization cases for both symmetric and asymmetric airfoils, where the optimization minimizes drag.

Acknowledgements

This work was funded by the EnCAPS project, AFRL Contract FA8650–20–2–2002: "EnCAPS: Enhanced Computational Aircraft Prototype Syntheses", with Dr. Ryan Durscher as the Technical Monitor. The authors would like to thank Bob Haimes for his contributions to the ideas of this paper and mentorship throughout this work.

References

[1] Antony Jameson, and Sriram Shankaran, and Luigi Martinelli, "Continuous Adjoint Method for Unstructured Grids," AIAA 2008-1226, May 2008.

[2] Pironneau, O., "On optimum design in fluid mechanics," *Journal of fluid mechanics*, Vol. 64, 1974, pp. 97–110.

[3] Kim, J.-E., and Rao, V. N., and Koomullil, R. P., and Ross, D. H., and Soni, B. K., and Shih, A. M., "Development of an efficient aerodynamic shape optimization framework," *Mathematics and Computers in Simulation*, Vol. 79, No. 8, 2009, pp. 2373–2384.

[4] Nadarajah, S. K. and Tatossian, C, "Multi-objective aerodynamic shape optimization for unsteady viscous flows," *Optimization and Engineering*, Vol. 11, No. 1, 2010, pp. 67–106.

[5] D. J. Poole, and C. B. Allen, and T. C. S. Rendall, "Application of Control Point-Based Aerodynamic Shape Optimization to Two-Dimensional Drag Minimization," AIAA 2014-0413, January 2014.

[6] Nadarajah, S., and Jameson, A., *A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization*, 2012. https://doi.org/10.2514/6.2000-667, URL https://arc.aiaa.org/doi/abs/10.2514/6.2000-667.

[7] Marian Nemec and Michael J. Aftosmis, "Parallel Adjoint Framework for Aerodynamic Shape Optimization of Component-Based Geometry," AIAA 2011-1249, January 2011.

[8] Hugo Gagnon and David W. Zingg, "Two-Level Free-Form Deformation for High-Fidelity Aerodynamic Shape Optimization," AIAA 2012-5447, September 2012.

[9] George R. Anderson, and Michael J. Aftosmis, and Marian Nemec, "Parametric Deformation of Discrete Geometry for Aerodynamic Shape Design," , No. 2012-0965, 2012.

[10] Sederberg, T. W., and Parry, S. R., "Free-Form Deformation of Solid Geometric Models," *SIGGRAPH Comput. Graph.*, Vol. 20, No. 4, 1986, p. 151–160. https://doi.org/10.1145/15886.15903, URL https://doi.org/10.1145/15886.15903.

[11] Gagnon, H., and Zingg, D. W., "Two-Level Free-Form and Axial Deformation for Exploratory Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 53, No. 7, 2015, pp. 2015–2026. https://doi.org/10.2514/1.J053575.

[12] Kenway, G., Kennedy, G., and Martins, J., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," 2010. https://doi.org/10.2514/6.2010-9231.

[13] Hicken, J. E., and Zingg, D. W., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," *AIAA Journal*, Vol. 48, No. 2, 2010, pp. 400–413. https://doi.org/10.2514/1.44033.

[14] Gomez, M., Galbraith, M. C., and Haimes, R., *Free-Form Deformation of Parametric CAD Geometry via B-Spline Transformations*, 2023. https://doi.org/10.2514/6.2023-3601, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-3601.

[15] Martin, M., Andres, E., Widhalm, M., Bitrian, P., and Lozano, C., "Non-uniform rational B-splines based aerodynamic shape design optimization with the DLR TAU code," *Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering*, Vol. 226, 2012, pp. 1225–1242. https://doi.org/10.1177/0954410011421704.

[16] Haimes, R., and Dannenhoffer, J, "The Engineering Sketch Pad: A solid-modeling, feature-based, web-enabled system for building parametric geometry," 2013.

[17] Dannenhoffer, J., and Haimes, R., "Design Sensitivity Calculations Directly on CAD-based Geometry," 2015. https://doi.org/10.2514/6.2015-1370.

[18] Marshall C. Galbraith and Steven R. Allmaras and David L. Darmofal, “A verification driven process for rapid development of CFD software,” AIAA 2015-0818, January 2015.

[19] de Boor, C., *A Practical Guide to Splines,*, Springer-Verlag, 1978.

[20] Amoiralis, E. I. and Nikolos, I. K., “Freeform Deformation Versus B-Spline Representation in Inverse Airfoil Design,” *Journal of Computing and Information Science in Engineering*, Vol. 8, No. 2, 2008, pp. 152–172.

[21] Gomez, M., Galbraith, M. C., and Haimes, R., “On Analysis Driven Shape Design Using B-Splines,” *AIAA 2022-1736*, 2022. https://doi.org/10.2514/6.2022-1736, URL https://acdl.mit.edu/ESP/Publications/AIAApaper2022-1736.pdf.

[22] Lee, C., Koo, D., and Zingg, D. W., “Comparison of B-Spline Surface and Free-Form Deformation Geometry Control for Aerodynamic Optimization,” *AIAA Journal*, Vol. 55, No. 1, 2017, pp. 228–240. https://doi.org/10.2514/1.J055102.

[23] Wang, X., and Qian, X., “An optimization approach for constructing trivariate B-spline solids,” *Computer-Aided Design*, Vol. 46, 2014, pp. 179–191. https://doi.org/https://doi.org/10.1016/j.cad.2013.08.030, URL https://www.sciencedirect.com/science/article/pii/S001044851300170X, 2013 SIAM Conference on Geometric and Physical Modeling.

[24] Charles L. Ladson, and Cuyler W. Brooks, Jr., and Acquilla S. Hill, and Darrell W. Sproles, “Computer Program to Obtain Ordinates for NACA Airfoils,” *National Aeronautics and Space Administation*, 1996.

[25] Johnson, S. G., “The NLopt nonlinear-optimization package,” 2007.

[26] J. Nocedal, “Updating quasi-Newton matrices with limited storage,” *Math. Comput.*, Vol. 35, 1980, pp. 773–782.

[27] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Math. Programming*, Vol. 45, 1989, pp. 503–528.

[28] Gomez, M., “Analysis Driven Shape Design of Parametric Geometry Using B-Splines and Free-Form Deformation,” 2023. URL https://dspace.mit.edu/handle/1721.1/152947.

[29] Durscher, R., and Reedy, D., “pyCAPS: A Python Interface to the Computational Aircraft Prototype Syntheses,” *AIAA Scitech 2019 Forum*, 2019. URL https://acdl.mit.edu/esp/Publications/AIAApaper2019-2226.pdf.

[30] Heath, C., and Gray, J., “OpenMDAO: Framework for Flexible Multidisciplinary Design, Analysis and Optimization Methods,” 2012. https://doi.org/10.2514/6.2012-1673.

[31] Kraft, D., “A Software Package for Sequential Quadratic Programming,” 1988.

[32] Kulfan, B. M., “Universal Parametric Geometry Representation Method,” *Journal of Aircraft*, Vol. 45, No. 1, 2008, pp. 142–158. https://doi.org/10.2514/1.29958.