Author version

Published as:

Ruh, M. L., & Hwang, J. T. (2026). Surface mesh deformation for large-scale multidisciplinary design optimization of aircraft concepts. In AIAA AVIATION 2026 Forum (Paper No. AIAA 2026-4569). https://doi.org/10.2514/6.2026-4569

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/ruh2026surface/

```
@inproceedings{ruh2026surface,
  author = {Marius L. Ruh and John T. Hwang},
  title = {Surface Mesh Deformation for Large-Scale Multidisciplinary Design Optimization
    of Aircraft Concepts},
  booktitle = {AIAA AVIATION 2026 Forum},
  year = {2026},
  doi = {10.2514/6.2026-4569}
}
```

Surface Mesh Deformation for Large-Scale Multidisciplinary Design Optimization of Aircraft Concepts

Marius L. Ruh*, and John T. Hwang†
University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093

Large-scale multidisciplinary design optimization of aircraft concepts requires differentiable predictive models that remain robust under large geometric changes. While semi-empirical and low-fidelity aerodynamic models capture first-order sizing effects, they are often insufficient for predicting full-configuration aerodynamics, particularly for multi-component configurations. Higher-fidelity models, such as panel methods and computational fluid dynamics, address this limitation but require conforming surface meshes. Maintaining such meshes on the outer mold line (OML) under large deformations is a major bottleneck. In this work, we develop a differentiable surface mesh deformation algorithm for independently parameterized and intersecting components undergoing large shape changes. The method combines explicit parametric geometry with implicit signed distance function representations to avoid explicit intersection computations and enable robust projection of mesh nodes onto a smooth OML. We apply the approach to a representative aircraft configuration subjected to large geometric deformations. In addition, we demonstrate the deformation algorithm by pairing it with a 3D panel code and solving a large-scale fuel-burn-minimization problem involving planform, wing twist and camber, and component placement variables. The results demonstrate robust optimization convergence while maintaining acceptable mesh quality under large deformations; in the most severe case, the 95th-percentile cell aspect ratio remains below 2.2, and fewer than 0.14% of cells exceed an aspect ratio of 5. These findings suggest that the proposed algorithm is a potential enabler for incorporating higher-fidelity aerodynamic models into conceptual aircraft design.

I. Introduction

Large-scale multidisciplinary design optimization (MDO) with analytic, adjoint-based sensitivity analysis provides a systematic framework for aircraft design that can efficiently explore high-dimensional design spaces with hundreds to thousands of variables. Recent advances in computing hardware, automated sensitivity analysis [1], and MDO frameworks [2] have enabled the application of large-scale MDO to a wide range of problems, including high-fidelity aerostructural optimization of tube-and-wing aircraft [3–5], aerodynamic shape optimization of unconventional configurations [6, 7], and system-level design studies of emerging concepts such as electric vertical takeoff and landing (eVTOL) vehicles [8, 9].

A useful way to frame recent progress in MDO is through the multidimensional model-fidelity framework introduced by Venkataraman and Haftka [10]. As shown in Fig. 1, the three axes represent increasing model fidelity, increasing design-variable dimension, and increasing disciplinary or design-condition scope. Many MDO studies advance along one or two of these axes: high-fidelity aerostructural optimization increases model fidelity but is often applied to a narrow set of disciplines and design variables, while system-level conceptual MDO increases design scope and dimensionality but often relies on lower-fidelity aerodynamic models. The broader MDO community has increasingly pursued this high-fidelity, high-dimensional, and system-level region at earlier design stages to improve the reliability and interpretability of optimization results, reduce conceptual-design uncertainty, and support a more efficient aircraft design cycle [11].

*PhD Candidate, Department of Mechanical and Aerospace Engineering, AIAA student member, mruh@ucsd.edu
†Associate Professor, Department of Mechanical and Aerospace Engineering, AIAA senior member

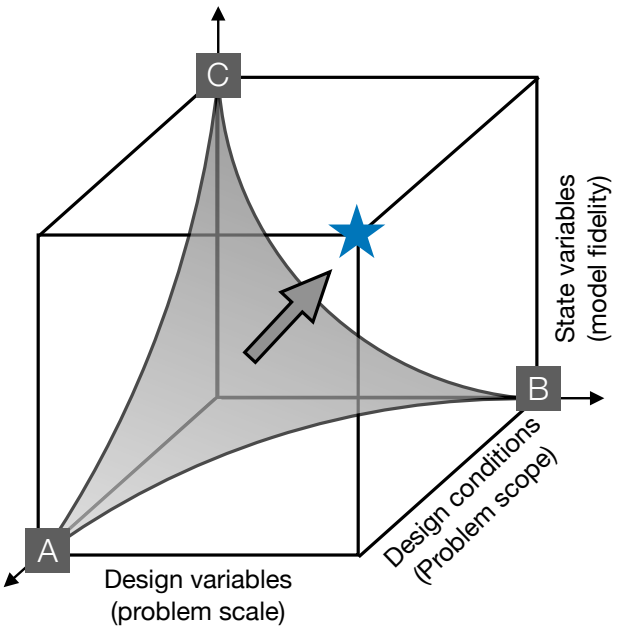


Fig. 1 Three-dimensional complexity–fidelity framework for multidisciplinary design optimization, adapted from Venkataraman and Haftka [10]. The axes represent increasing design-variable dimension, increasing problem scope, and increasing model fidelity. The gray surface notionally illustrates how MDO studies can increase in complexity along different combinations of these axes, while the arrow indicates the broader trend toward higher-fidelity, higher-dimensional, system-level optimization at earlier design stages.

At the conceptual design stage, the primary objective is to rapidly explore a large design space to identify configurations that satisfy mission and performance requirements [12, 13]. This process inherently involves large geometric variations, including changes in wing planform (e.g., area and aspect ratio), component placement (e.g., tail moment arm), and spatially varying quantities such as twist and camber. To support such exploration, predictive models must remain robust under these geometric changes while maintaining computational efficiency.

Traditionally, conceptual design relies on semi-empirical and low-fidelity physics-based models that capture first-order sizing effects without requiring detailed geometric representations. However, these models are often derived from historical data and therefore have limited applicability to novel configurations. In contrast, higher-fidelity physics-based models are largely configuration-independent and can capture important effects such as component interactions and viscous phenomena [14]. Because these models compute aerodynamic field quantities, such as pressure, over the full configuration, they can compute forces, moments, and stability derivatives from a single coupled solution rather than relying on separate component estimates and empirical interference corrections.

A key challenge in incorporating higher-fidelity aerodynamic models into conceptual design is the need for a high-quality surface mesh. For multi-component geometries with intersections, many aerodynamic solvers, such as panel codes (e.g., [15]) and unstructured CFD (e.g., SU2 [16]), require conforming surface meshes that accurately represent the outer mold line (OML). During optimization, these meshes must be updated as the geometry changes. This requirement becomes particularly challenging under large geometric deformations, where component intersections evolve due to changes in planform variables, component translations and rotations, or scaling operations. In such cases, the mesh must remain on the OML, conform to updated intersections, and maintain acceptable element quality.

Existing mesh deformation methods, including Laplacian smoothing [17], spring analogy[18], and radial-basis-function (RBF) interpolation [19], are effective for small or localized deformations. However, they are generally not designed to handle large, multi-component geometric changes with evolving intersections, nor are they typically formulated to be fully differentiable within gradient-based optimization loops. As a result, conceptual-level large-scale MDO studies are often limited to low-fidelity aerodynamic models such as the vortex lattice method (VLM), which do not require conforming surface meshes but cannot capture full-configuration effects such as fuselage-wing interactions without empirical corrections [15]. This limitation motivates the development of a surface mesh deformation algorithm that is suitable for conceptual design and compatible with large-scale MDO. Enabling higher-order aerodynamic models

at this stage would improve the accuracy of performance predictions and reduce reliance on empirical corrections, thereby producing more reliable and interpretable optimization results. Furthermore, increasing the fidelity of conceptual design analyses has the potential to accelerate the overall aircraft design process by reducing the risk of major design changes in later stages.

The goal of this work is to enable full-configuration aerodynamic modeling within large-scale MDO of aircraft concepts under large geometric changes. To achieve this, we develop a differentiable surface mesh deformation algorithm that robustly handles independently parameterized and intersecting components undergoing translations, rotations, and scaling. Our approach combines explicit parametric geometry representations with implicit representations based on signed distance functions (SDFs), allowing us to avoid explicit intersection computations and instead represent the OML as the zero-level set of a smooth scalar field. The resulting deformation procedure is fully differentiable and compatible with adjoint-based sensitivity analysis, making it suitable for integration into gradient-based MDO frameworks. We demonstrate the proposed method on representative aircraft configurations undergoing large geometric changes. In addition, we integrate the deformation algorithm with a panel method and solve a large-scale fuel-burn-minimization problem involving planform variables, wing twist and camber, and component placement variables.

The remainder of this paper is organized as follows. First, we review background concepts related to explicit and implicit geometry representations and existing mesh deformation methods in Sec. II. Next, we present the proposed surface mesh deformation algorithm in detail in Sec. III. We then validate and demonstrate the performance of the method through a series of deformation and optimization results in Sec. IV and Sec. V, and discuss the implications of the proposed approach for conceptual aircraft design as well as outline directions for future work.

II. Background: Geometry Representations

Most geometry tools used in aircraft conceptual design rely on explicit representations such as computer-aided design (CAD), constructive solid geometry (CSG), or deformational geometry parameterizations based on free-form deformation (FFD) [20–23]. We use both explicit and implicit geometry representations in this work. Explicit representations describe each aircraft component through parametric surfaces and provide a convenient mechanism for design-variable-driven geometry manipulation. Implicit representations describe each component through a signed distance function (SDF) and provide a convenient mechanism for projection and smooth union operations. This section introduces the notation and operations that we use in the surface mesh deformation algorithm.

A. Explicit Geometry Representation

We represent the aircraft geometry as a collection of independently parameterized components indexed by $c \in C$. Each component consists of one or more parametric surface patches. We write the evaluation of component c as

$$\mathbf{y} = \mathbf{S}_c(\boldsymbol{\mu}; \mathbf{C}_c), \qquad \boldsymbol{\mu} = (i, u, v), \tag{1}$$

where i is the patch index, $(u, v) \in [0, 1]^2$ are normalized parametric coordinates, and $\mathbf{C}_c$ denotes the coefficients or control points of component c. We use superscripts 0 and $*$ to denote the reference and deformed geometries, respectively, so that $\mathbf{S}_c^0$ and $\mathbf{S}_c^*$ denote the same component before and after geometry manipulation.

Projection is a central operation for explicit geometries. Given a point $\mathbf{x} \in \mathbb{R}^3$, the closest point on component c solves

$$\boldsymbol{\mu}_c(\mathbf{x}) = \arg\min_{\boldsymbol{\mu}} \|\mathbf{S}_c(\boldsymbol{\mu}; \mathbf{C}_c) - \mathbf{x}\|_2 . \tag{2}$$

This operation is useful for setup tasks such as assigning mesh vertices to components and storing reference parametric coordinates. However, repeated explicit projection and intersection computations can become expensive and may introduce nonsmoothness when solutions cross patch boundaries, if the parametric surfaces are only C^0 continuous across patch boundaries. These limitations, in part, motivate the use of implicit geometry representations in the deformation algorithm.

We manipulate the explicit geometry using free-form deformation (FFD) and component-level transformations. An FFD block is a tensor-product spline volume that embeds the control points of one or more components. For an FFD coordinate $\boldsymbol{\xi} \in [0, 1]^3$, the FFD map is

$$\mathbf{F}(\boldsymbol{\xi}; \mathbf{B}) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} N_i(\xi_1) N_j(\xi_2) N_k(\xi_3) \mathbf{B}_{ijk}, \tag{3}$$

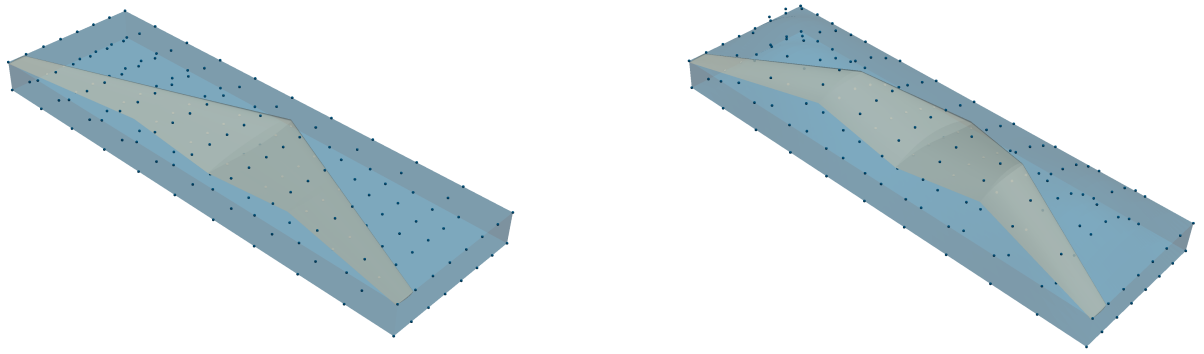
where $\mathbf{B}$ is the FFD control lattice and N_i, N_j, and N_k are spline basis functions. During setup, we embed each geometry control point $\mathbf{C}^0_{c,\ell}$ in the FFD block by storing its parametric coordinate $\boldsymbol{\xi}_{c,\ell}$. After the FFD lattice moves from $\mathbf{B}^0$ to $\mathbf{B}^*$, we update the embedded geometry control point by evaluating the deformed FFD map,

$$\mathbf{C}^*_{c,\ell} = \mathbf{F}(\boldsymbol{\xi}_{c,\ell}; \mathbf{B}^*). \tag{4}$$

This construction provides a smooth, low-dimensional map from design variables to the high-dimensional explicit geometry.

For lifting surfaces, we organize the FFD block by spanwise sections. We then apply intuitive section-level operations such as chord stretch, span stretch, sweep translation, twist rotation, camber change, and thickness change. We represent these section-level quantities with low-dimensional spline functions along the span, which gives smooth deformation patterns with relatively few design variables. Camber variables move the upper and lower FFD layers in the same local-thickness direction, whereas thickness variables move them in opposite directions. We scale the prescribed camber and thickness changes by local chord, so these variables have a nondimensional interpretation. For symmetric configurations, we mirror sectional parameters across the center plane and mirror span stretch antisymmetrically so that the root remains on the symmetry plane. Fletcher and Hwang [24] provide a detailed description of the FFD approach used in this paper.

We also apply component-level transformations when they provide a more direct parameterization than sectional FFD variables. For example, we can translate a wing, rotate a horizontal tail about a root quarter-chord point, or radially scale a fuselage about its centerline with a smooth axial taper. Together with the FFD approach, these operations produce the deformed component maps $\mathbf{S}^*_c$ and the deformed control-point arrays $\mathbf{C}^*_c$ that the mesh deformation algorithm uses.



(a) Wing geometry embedded in an FFD block discretized by a structured lattice of control points

(b) Deformed wing geometry due to perturbations of FFD-block control points

Fig. 2 We manipulate the central B-spline geometry by embedding it in an FFD volume and modifying the FFD control points.

B. Implicit Geometry Representation

An alternative is to represent geometry implicitly using signed distance functions (SDFs), which have become increasingly common in geometry processing and learning-based shape representations [25, 26]. We represent each component implicitly with a signed distance function $\phi_c : \mathbb{R}^3 \to \mathbb{R}$. In this paper, we use the convention that $\phi_c(\mathbf{x}) < 0$ inside component c, $\phi_c(\mathbf{x}) > 0$ outside component c, and $\phi_c(\mathbf{x}) = 0$ on the component boundary. For a smooth component boundary $\partial\Omega_c$, the signed distance is

$$\phi_c(\mathbf{x}) = s_c(\mathbf{x}) \inf_{\mathbf{y}\in\partial\Omega_c} \|\mathbf{x} - \mathbf{y}\|_2, \tag{5}$$

where $s_c(\mathbf{x})$ is the inside–outside sign. Away from the medial axis and for smooth boundaries, an exact SDF satisfies $\|\nabla\phi_c\|_2 = 1$, and $\nabla\phi_c$ equals the outward unit normal on the boundary. This property gives the standard SDF projection formula,

$$\mathbf{y}^* = \mathbf{x} - \phi_c(\mathbf{x})\nabla\phi_c(\mathbf{x}). \tag{6}$$

For a multi-component configuration, we define the OML as the boundary of the union of component volumes. With the sign convention above, the hard union SDF is the pointwise minimum of the component SDFs,

$$\phi_{\mathrm{OML}}(\mathbf{x}) = \min_{c \in C} \phi_c(\mathbf{x}). \tag{7}$$

This hard minimum assigns each point to a single closest or most-interior component. Near component intersections, such assignments can switch abruptly from one component to another, which introduces nonsmoothness in the OML representation and can produce numerical noise during mesh deformation.

To obtain a smooth OML representation near intersections, we replace the hard minimum with the smooth minimum

$$\Phi(\mathbf{x}) = -\frac{1}{\rho} \log \left(\sum_{c \in C} \exp[-\rho \phi_c(\mathbf{x})] \right), \tag{8}$$

where $\rho > 0$ controls the sharpness of the approximation. As ρ increases, Φ approaches the hard minimum. For finite ρ, Φ blends the component SDFs smoothly and produces a rounded transition near intersections. This smooth union is important for mesh deformation because near-intersection vertices can move along a continuous OML representation instead of being assigned discontinuously to a single component surface. In principle, one can reduce computational cost by evaluating only a local subset of component SDFs for a given point, but this is an implementation detail rather than a change in the mathematical definition of the OML.

There are several practical ways to evaluate component SDFs. For simple geometric primitives, such as spheres, boxes, cylinders, and capsules, analytic SDF expressions are available and can be evaluated efficiently. These expressions are useful for prototyping, verification, and simplified configuration studies, but they do not directly represent general aircraft components described by spline surfaces. A second option is to approximate each component by a triangulated surface and compute distances and signs with respect to the triangulation. This approach is robust and widely available (see e.g., [27]), but the resulting distance field is only as accurate as the triangulation and has nonsmooth gradients across triangle boundaries.

A third option is to compute the distance directly from the explicit parametric geometry using the projection operation in Eq. (2). This approach preserves the geometric accuracy of the B-spline representation and avoids replacing the component geometry with a piecewise-linear approximation. However, it requires a robust closest-point projection algorithm, and the sign must still be determined reliably. In this paper, we use a hybrid approach: we compute the unsigned distance from the explicit B-spline geometry and determine the sign using a triangulation-based inside–outside check. This preserves the geometric accuracy of the parametric surface in the distance computation while using a robust sign test. Figure 3 shows a representative SDF computed with this approach for a swept and tapered wing geometry. A fully projection-based sign computation using oriented surface normals is also possible, but it requires a sufficiently smooth and consistently oriented surface normal field.

A fourth option is to train a surrogate model of the component SDF from projection-based distance data. For example, one can sample points near the component boundary, compute signed distances using one of the methods above, and fit a smooth surrogate such as an RBF or neural implicit model. This approach can be particularly useful when the algorithm requires many component or union SDF evaluations, because the surrogate can be cheaper to evaluate and can reduce memory requirements compared with repeated projection queries or dense triangulated representations. However, the surrogate must approximate both SDF values and gradients accurately near the zero-level set, since these quantities directly affect seam motion and final OML projection. For this reason, surrogate-based SDF evaluation introduces an additional accuracy–cost tradeoff.

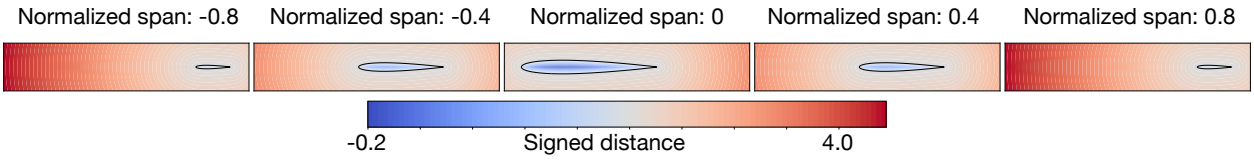


Fig. 3 Example signed distance field for a swept and tapered wing. We compute the unsigned distance using projection onto the B-spline geometry and determine the sign using a triangulation-based inside–outside check.

C. Overview of Existing Mesh Deformation Algorithms

Mesh deformation methods update the positions of existing mesh nodes after a change in the design geometry. For the present application, such a method must not only move a mesh, but do so for independently parameterized, intersecting aircraft components undergoing large translations, rotations, and scalings while preserving mesh quality and differentiability. This requirement is more restrictive than many classical dynamic-mesh settings, because the conforming surface mesh must remain on the OML and must follow the motion of component intersections. The relationship between a mesh deformation method and the underlying geometry representation therefore matters: some methods operate directly on mesh coordinates, whereas others use explicit geometry projections, component intersections, or implicit distance fields to keep the mesh tied to the geometry.

Classical physics-based methods include spring analogies, Laplacian or elliptic smoothing, and linear-elasticity mesh motion. Spring-analogy methods model mesh edges as springs and solve for equilibrium after boundary nodes move [18, 28, 29]. These methods are inexpensive and use sparse systems, but they may require torsional springs, load stepping, or local repair to avoid element inversion under large rotations or translations. Laplacian and elliptic smoothing methods diffuse boundary displacements through the mesh by solving sparse elliptic problems [30, 31]. They are efficient and differentiable for fixed operators, but the diffusion process can over-smooth local features and may not preserve quality in highly compressed regions. Elasticity-based methods treat the mesh as a deformable solid and usually provide better element-quality control for large motions, but they are more expensive and still depend on boundary motion that must be prescribed consistently on the geometry [32, 33].

Interpolation-based approaches, including inverse-distance weighting, thin-plate splines, and radial-basis-function (RBF) interpolation, construct smooth displacement fields from prescribed control or boundary-node displacements [34, 35]. These methods often tolerate larger rigid-body-like motions than purely local smoothing methods and are attractive for MDO because the fitted displacement field is differentiable for fixed control points. Their main drawback is that they do not, by themselves, determine how boundary nodes should move when component intersections change. For multi-component aircraft geometries, a naive interpolation of independently deformed component surfaces can break conformity at seams or force near-intersection nodes to follow the wrong component. Thus, interpolation-based methods are useful as propagation mechanisms, but they require geometry-aware displacement data to handle evolving intersections.

Recent MDO-tailored surface deformation methods address this missing geometry step more directly. Yildirim et al. developed a surface mesh deformation method near component intersections for high-fidelity design optimization, using triangulated surface representations, intersection-aware projections, and mesh updates that preserve topology near seams [36]. In our previous work, we recomputed explicit intersections between independently parameterized components and used thin-plate splines to propagate the updated seam motion to the surface mesh [37]. That approach established a differentiable path for large component translations and rotations, but repeated explicit intersection construction and projection across parametric patches can be expensive and can introduce numerical noise. An earlier projection-error formulation also highlighted the same central requirement: the mesh must remain tied to the OML while preserving quality as intersections move [38]. The present work builds on these ideas but shifts the intersection and projection machinery toward an implicit SDF-based OML representation, while retaining explicit parametric geometry for component motion and intersection-aware displacement construction.

III. Novel Surface Mesh Deformation Algorithm

We now describe the surface mesh deformation algorithm. The algorithm starts from the deformed explicit component maps $\mathbf{S}_c^*$ and the corresponding deformed component SDFs ϕ_c^* introduced in Sec. II. It updates a reference surface mesh while keeping the mesh connectivity fixed. The main challenge is to move mesh vertices consistently near component intersections, where a vertex should not simply follow one component if the deformed intersection has shifted relative to that component.

The algorithm uses two complementary geometry descriptions. It uses the explicit component surfaces to define physically meaningful component and seam motion. It uses the implicit SDF union to project the final mesh onto a smooth OML representation. This combination avoids repeated explicit intersection-curve construction while still accounting for the motion of intersections under large component translations, rotations, and scaling.

Table 1 Summary of mesh deformation approaches in the context of large-scale conceptual-aircraft MDO with independently parameterized, intersecting components.

Approach	Theoretical basis and geometry link	Strengths	Limitations for the present goal
Spring analogy	Sparse equilibrium of edge or torsional springs; operates mainly on mesh connectivity [18, 28, 29].	Low cost; simple implementation; useful for moderate boundary motion.	Local model can invert or distort under large component translations/rotations; does not determine updated intersections.
Laplacian/elliptic smoothing	Diffusion of prescribed boundary motion through sparse elliptic operators [30, 31].	Efficient; smooth displacement fields; differentiable for fixed operators.	Over-smooths local features; weak quality control near compressed seams; requires externally prescribed intersection motion.
Elasticity-based motion	Mesh is treated as a linear or nonlinear elastic solid [32, 33].	Better element-quality control than local smoothing; natural extension to volume meshes.	Higher cost; boundary motion still must come from explicit geometry or intersection updates; topology changes are not automatic.
RBF/TPS/IDW interpolation	Global or compact displacement interpolation from control points or boundary displacements [34, 35].	Handles large smooth motions; differentiable for fixed anchors; useful propagation mechanism.	Dense or large systems may scale poorly; does not by itself enforce OML conformity or evolving component intersections.
Intersection-aware MDO methods	Explicit geometry projections, triangulated surface operations, or fitted seam curves near component intersections [36, 37].	Directly target conforming surface meshes near intersections; compatible with high-fidelity design optimization.	Repeated explicit intersection/projection operations can be costly or nonsmooth across patches; robustness depends on geometric preprocessing.
Present SDF-assisted approach	Explicit parametric surfaces define component and seam motion; smooth SDF union defines the final OML projection.	Avoids explicit intersection-curve construction during each update; supports large multi-component changes; improves differentiability near seams through smooth unions.	Requires accurate component SDF values and gradients; discrete setup data and large topology changes still require care.

A. Algorithm Overview

Let the reference mesh be defined by vertex positions $\{\mathbf{x}_a^0\}_{a=1}^{N_v}$ and fixed edge and face connectivity. During setup, we select a subset $\mathcal{I}$ of mesh vertices as RBF training vertices. We also store setup data that do not change during differentiation, including component ownership, seam labels, candidate target-component sets, graph distances to seams, nearest-seam maps, and protected feature labels.

For a new deformed geometry, we first compute displacement data at the training vertices. Away from intersections, we compute this displacement by evaluating the motion of the associated component at the stored parametric coordinate. At intersections, we recompute where the deformed seam lies by solving a scalar SDF residual along a parametric curve on one component. Near intersections, we blend these two motion models so that seam motion dominates close to the intersection and component motion dominates away from it.

After constructing the training displacements, we fit a smooth vector-valued RBF displacement field and evaluate it at all mesh vertices. This step propagates sparse, geometry-informed displacement data to the full mesh. Finally, we project the displaced vertices onto the smooth OML defined by the soft union SDF and optionally interleave this projection with smoothing to improve element quality. Algorithm 1 summarizes the complete procedure.

B. Component-Based Displacement Data

For each training vertex assigned to a single component, we store a reference parametric coordinate $\boldsymbol{\mu}_i^0$ during setup. When the geometry changes, we evaluate the same parametric coordinate on the reference and deformed component surfaces and compute

$$\Delta\mathbf{x}_i^{\text{comp}} = \mathbf{S}_{c_i}^*(\boldsymbol{\mu}_i^0) - \mathbf{S}_{c_i}^0(\boldsymbol{\mu}_i^0), \tag{9}$$

where c_i denotes the owner component of training vertex i. This displacement model works well for vertices that remain attached to one component. It does not, however, provide the right motion for seam vertices, because seam vertices must remain consistent with the deformed intersection of two or more components.

We identify seam vertices during setup by checking whether a mesh vertex lies close to multiple components. For a pairwise seam, we store a reference component r and a target component t. The reference component provides the parametric curve along which we search, while the target component provides the SDF residual that defines the intersection. For multi-component seams, we replace the single target component with a small candidate set of target components.

C. Soft-Bracketed Intersection Motion

The most important part of the algorithm is the recomputation of seam motion from the deformed analytic geometry. Instead of assigning an intersection vertex to one component or averaging the displacement of two components, we

ask where the deformed reference component intersects the deformed target geometry. This gives seam displacements that follow the actual deformed intersection and therefore provide geometrically meaningful training data for the mesh motion.

Consider a seam training vertex i whose reference seam lies on component r and whose target geometry is represented by a component set $\mathcal{T}_i$. We define a one-dimensional search curve on the deformed reference component by varying one local parametric coordinate while holding the other fixed,

$$\mathbf{q}_i(\lambda) = \mathbf{S}_r^*(\boldsymbol{\mu}_i(\lambda)), \qquad \lambda \in [0, 1]. \tag{10}$$

The function $\boldsymbol{\mu}_i(\lambda)$ encodes the stored patch index, the fixed transverse coordinate, and the searched coordinate. We then define the scalar residual

$$g_i(\lambda) = \Phi_{\mathcal{T}_i}^*(\mathbf{q}_i(\lambda)), \tag{11}$$

where $\Phi_{\mathcal{T}_i}^*$ is the smooth union SDF of the deformed target components. A root of g_i corresponds to a point on the deformed reference component that also lies on the deformed target OML. We therefore compute the deformed seam point as

$$\mathbf{x}_{i,\text{seam}}^* = \mathbf{q}_i(\lambda_i^*), \qquad g_i(\lambda_i^*) = 0, \tag{12}$$

and set the seam displacement to

$$\Delta\mathbf{x}_i^{\text{seam}} = \mathbf{x}_{i,\text{seam}}^* - \mathbf{x}_i^0. \tag{13}$$

We solve the scalar root-finding problem with a fixed number of bracketed iterations. Standard bisection updates the bracket using a binary sign test. We replace this binary decision with a smooth same-side indicator, for example

$$\sigma_k = H_\eta\left(g_i(l_k)g_i(m_k)\right), \qquad H_\eta(z) = \frac{1}{2}\left[1 + \tanh(\eta z)\right], \tag{14}$$

where l_k is the lower bracket endpoint, m_k is the midpoint, and η controls the sharpness of the transition. We then use σ_k to smoothly choose which half of the bracket to keep. This soft-bracket construction preserves the logic of bisection while replacing the discontinuous branch with differentiable arithmetic. The exact smoothing function is not essential; for example, the hyperbolic tangent can be replaced by another smooth indicator such as a sigmoid function. The essential point is that the root location changes smoothly with the deformed geometry as long as the root remains inside the chosen bracket. Figure 4 illustrates how the soft-bracketed search advances seam vertices toward the deformed intersection by combining a parametric search on one component with SDF evaluations of the intersecting component.

This intersection-motion model also supports moderate changes in local intersection ownership. For example, a wing-root seam point may switch from intersecting a fairing to intersecting a fuselage. We handle this case by including both the fairing and fuselage in $\mathcal{T}_i$ and using the smooth union residual in Eq. (11). Large topology changes, such as a seam leaving its bracket or a new component entering the relevant target set, require regenerating the setup data.

D. Near-Intersection Blending

The exact seam displacements provide reliable motion at intersection vertices, but we also need to move nearby vertices smoothly. If we only prescribed seam displacements at the intersection and component displacements elsewhere, the training data could change abruptly across a few mesh edges. We therefore blend seam motion into component motion over a finite graph-distance neighborhood of each seam.

Let $\mathcal{J}$ denote the set of seam training vertices, and let $d_G(i, \mathcal{J})$ denote the graph distance from training vertex i to the nearest seam vertex. We define a smooth blending weight $\alpha_i \in [0, 1]$ that equals one at the seam and decays to zero away from the seam. If $j(i)$ denotes the nearest seam vertex or a weighted average of nearby seam vertices, the blended training displacement is

$$\Delta\mathbf{x}_i = (1 - \alpha_i)\Delta\mathbf{x}_i^{\text{comp}} + \alpha_i\Delta\mathbf{x}_{j(i)}^{\text{seam}}. \tag{15}$$

This blending makes the mesh respond to the movement of the deformed intersection without forcing all nearby vertices to lie exactly on the seam. It also prevents the RBF displacement field from fitting incompatible motion data near component intersections.

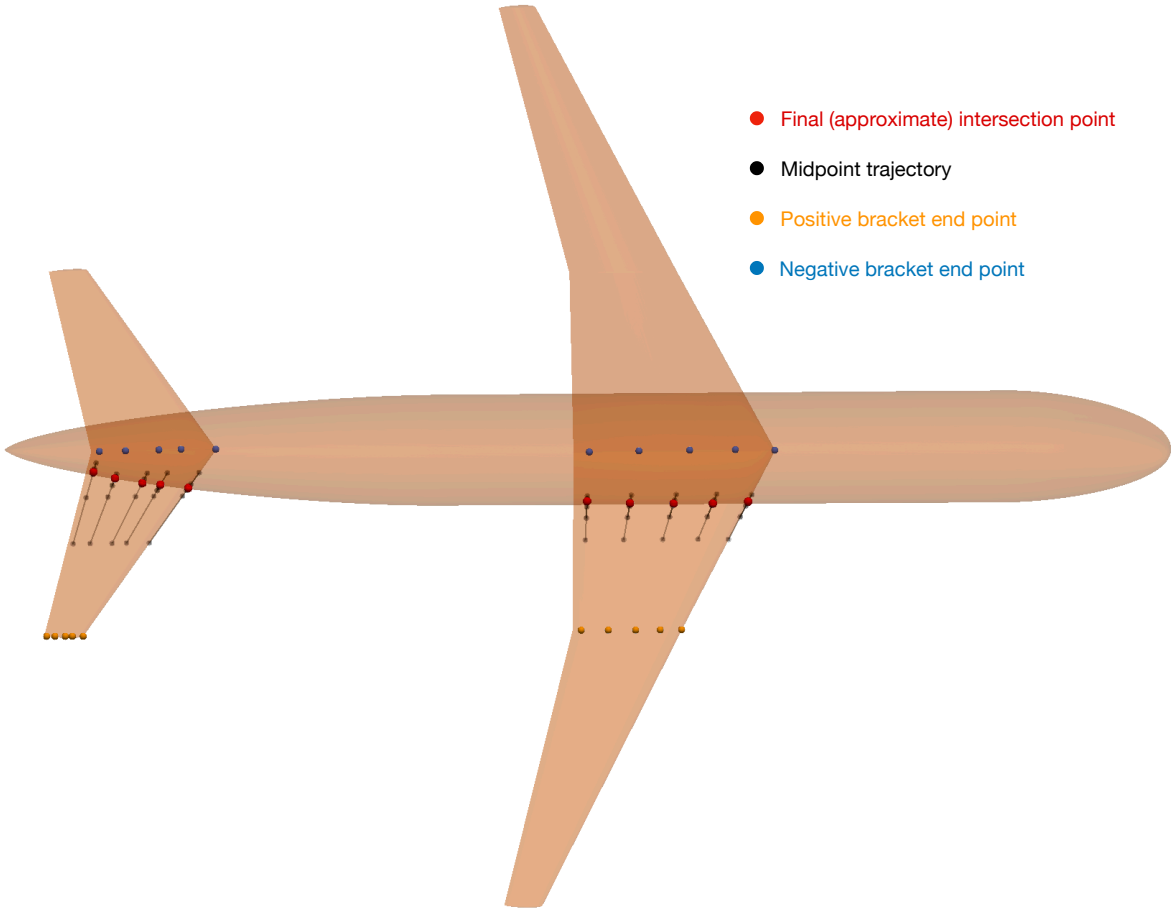


Fig. 4 Soft-bracketed intersection update for seam vertices. The algorithm searches along one parametric direction on the explicit surface of one component while evaluating the signed distance function of the intersecting component. The lower and upper bracket endpoints define a differentiable bracketed search whose midpoint trajectory converges toward the final seam point. This seam point defines the intersection displacement used to drive nearby mesh vertices.

E. RBF Mesh Displacement Field

We use the training pairs $(\mathbf{x}_i^0, \Delta\mathbf{x}_i)$ for $i \in \mathcal{I}$ to construct a smooth displacement field over the reference mesh. We use a vector-valued RBF interpolant with an affine polynomial term,

$$\mathbf{d}(\mathbf{x}) = \sum_{j \in \mathcal{I}} \psi\left(\|\mathbf{x} - \mathbf{x}_j^0\|_2\right) \mathbf{w}_j + \mathbf{A}^T \mathbf{p}(\mathbf{x}), \tag{16}$$

where ψ is a radial kernel, $\mathbf{w}_j \in \mathbb{R}^3$ are RBF weights, $\mathbf{p}(\mathbf{x}) = [1, x_1, x_2, x_3]^T$, and $\mathbf{A}$ contains the affine coefficients. In this work, we use the cubic kernel $\psi(r) = r^3$. We obtain the coefficients by solving the standard RBF interpolation system with the usual polynomial side condition, or a regularized least-squares variant when we use oversampled training data.

After fitting the RBF field, we displace every mesh vertex according to

$$\hat{\mathbf{x}}_a = \mathbf{x}_a^0 + \mathbf{d}(\mathbf{x}_a^0), \qquad a = 1, \ldots, N_v. \tag{17}$$

This step propagates sparse training data to the full surface mesh and keeps the mesh connectivity fixed. We can also reapply protected feature constraints, symmetry-plane constraints, or stationary-row constraints after evaluating the RBF field.

F. Projection onto the Smooth OML

The RBF displacement field gives a smooth mesh motion, but it does not guarantee that the moved vertices lie exactly on the deformed OML. We therefore project active vertices onto the zero-level set of the smooth union SDF Φ^*_C. For vertex a, we use the fixed-point update

$$\mathbf{x}_a^{k+1} = \mathbf{x}_a^k - \frac{\Phi^*_C(\mathbf{x}_a^k)}{\left\|\nabla\Phi^*_C(\mathbf{x}_a^k)\right\|_2^2 + \varepsilon}\nabla\Phi^*_C(\mathbf{x}_a^k), \tag{18}$$

initialized with $\mathbf{x}_a^0 = \hat{\mathbf{x}}_a$ for the projection loop. If Φ^*_C were an exact SDF, this update would reduce to the standard SDF projection in Eq. (6). The denominator improves robustness near smooth-min transition regions, where $\|\nabla\Phi^*_C\|_2$ may differ from one.

The smooth union is crucial near component intersections. A hard projection would require each vertex to select exactly one component SDF, so small changes in geometry could cause a near-intersection vertex to switch components discontinuously. Such switches can introduce numerical noise in both the mesh motion and its sensitivities. By contrast, the soft-min OML creates a continuous transition between components for finite ρ. Near-intersection vertices can therefore slide along a rounded, continuous boundary instead of being forced onto one side of a sharp component intersection. This behavior improves mesh movement and supports differentiability in the regions where conforming surface meshes are most difficult to maintain.

We optionally augment the projection with graph-Laplacian smoothing to improve element quality after large deformations, while the projection step returns the smoothed vertices to the smooth OML. We exclude protected features such as sharp trailing edge, fixed rows, and other constrained vertices (e.g., the tail region of a fuselage intersected with a horizontal tail) from smoothing and reapply their constraints after projection.

G. Differentiability and Sensitivity Requirements

For fixed setup data, the deformation map consists of differentiable operations. These operations include parametric surface evaluation, SDF evaluation, soft-bracketed seam search, near-intersection blending, RBF fitting and evaluation, smoothing, and smooth-OML projection. The fixed setup data include the selected training vertices, component ownership labels, seam labels, candidate target-component sets, graph-distance neighborhoods, nearest-seam maps, and protected feature labels. We do not differentiate through changes in these discrete setup quantities.

The soft-bracket seam search makes the intersection-motion step compatible with gradient-based optimization. It replaces the branch in standard bisection with a smooth approximation while retaining the robustness of bracketed root finding. This treatment is most accurate when the root remains inside the setup bracket and when the active target components remain within the candidate set $\mathcal{T}_i$. If the design change violates these assumptions, we regenerate the setup.

The projection step imposes an additional sensitivity requirement. The forward projection in Eq. (18) uses both Φ^*_C and its spatial gradient $\nabla\Phi^*_C$. Therefore, differentiating the final projected vertex positions with respect to geometry design variables requires differentiating through the SDF-gradient evaluation. If we compute component SDFs through closest-point projection onto B-spline surfaces, this derivative involves mixed second derivatives of the projection-defined distance function with respect to the spatial point and the geometry control points. Equivalently, an adjoint implementation must use second-order information from the implicit closest-point problem, often described as a second-order adjoint contribution. Without this contribution, gradients through the final OML projection are incomplete, although the forward mesh motion may still be useful.

Surface smoothness also affects differentiability. If adjacent B-spline patches are only C^0 or have discontinuous normals, then the closest-point map, SDF gradient, and projected mesh motion can exhibit derivative jumps near patch boundaries. A smoother surface representation, ideally at least G^1 and preferably C^1 across patch interfaces, reduces this source of numerical noise. This requirement is particularly important because both the seam search and the final projection use SDF gradients.

H. Summary of Algorithmic Properties

The proposed algorithm has four properties that make it suitable for large-scale MDO of aircraft concepts. First, it preserves fixed mesh connectivity while allowing large component translations, rotations, and scaling. Second, it uses deformed analytic geometry to recompute seam-consistent displacement data rather than relying only on component-wise displacement interpolation. Third, it projects the displaced mesh onto a smooth SDF union of the OML, which lets near-intersection vertices move along a continuous boundary and avoids discontinuous component assignment. Fourth,

it expresses the full deformation procedure with differentiable operations for fixed setup data, making the method compatible with adjoint-based sensitivity analysis.

These properties address the main limitation of conventional surface mesh deformation methods in conceptual aircraft design. Instead of assuming small, localized mesh changes, the algorithm explicitly accounts for the motion of intersections between independently parameterized components. This capability enables higher-order aerodynamic models that require conforming surface meshes to be used in large-scale conceptual design optimization.

Algorithm 1 Surface mesh deformation algorithm

Input: Reference mesh vertices $\{\mathbf{x}_a^0\}_{a=1}^{N_v}$, fixed mesh connectivity, setup data, deformed component maps $\mathbf{S}_c^*$, and deformed SDFs ϕ_c^*.

Output: Deformed mesh vertices $\{\mathbf{x}_a^*\}_{a=1}^{N_v}$.

1: Compute component-based displacements at component-owned training vertices using Eq. (9).
2: Recompute seam displacements at seam training vertices using the soft-bracketed residual in Eq. (11).
3: Blend component and seam displacements near intersections using Eq. (15).
4: Fit the RBF displacement field in Eq. (16) to the training displacements.
5: Move all mesh vertices using Eq. (17).
6: Reapply protected feature, symmetry-plane, and stationary-row constraints.
7: Project active vertices onto the smooth OML using Eq. (18), optionally interleaved with smoothing.

IV. Validation

The test case for this study is an Embraer 175 regional jet, for which we use a representative geometry built in `OpenVSP` [21]. For simplicity, we only consider the fuselage, wing, and horizontal tail components. We first show the effects of the smoothed SDF projection on the OML mesh near component intersection in Sec. IV.A and then validate the mesh deformation algorithm in Sec. IV.B by showing its ability to tolerate large geometry changes while preserving acceptable mesh quality.

A. Intersection Smoothing

As discussed in Sec. III.F, the last step of the surface mesh deformation algorithm is a projection onto the OML, represented by the zero-level set of the union of the component signed distance functions. For exact SDFs, the union operator is a hard minimum. However, to maintain differentiability we use a soft-minimum function. A desirable side-effect is that intersection regions are smoothed so that filleting between components is a natural part of the formulation. In Fig. 5, we show the effect of the smoothing parameter ρ (see Eq. 8) on the intersection behavior for a relatively fine surface mesh of about 100,000 elements. For high values of ρ, the intersection behavior approaches a sharp, discontinuous intersection with a kink. On the other hand, for low values of ρ, we see a smooth fillet forming between the intersecting components, which is especially visible near the horizontal tail intersection with the fuselage. The filleting behavior is desirable for mesh movement of nodes near and on component intersections. Rather than forcing such nodes to be on a single component, they may smoothly move along a continuous OML representation.

B. Mesh Deformation Results

For the following mesh deformation test cases, we generate a coarser quad-dominant surface mesh with about 15,000 elements, which we also use in the large-scale MDO study. The first deformation test case involves a simultaneous translation and rotation of the wing along the fuselage as well as a rotation of the horizontal stabilizer. The maximum wing translation is approximately ± 1.5 times the mean aerodynamic chord (MAC), which corresponds to ± 5 m. The extent of the wing and tail rotation is $\pm 10^\circ$ for each component. Figure 6 shows the results. Under the extreme stretching and warping of the surface mesh, the algorithm preserves reasonable mesh quality and avoids element collapse or inversion. In Appendix Fig. 12, we show several mesh quality metrics for the two most extreme deformations as well as for the baseline mesh. The results show that while number of high-aspect-ratio elements increases under extreme deformation, which is to be expected, the overall mesh quality is maintained.

The second mesh deformation test case shows the algorithm's ability to tolerate geometric changes in the spanwise direction of the wing in addition to translations and rotations. Similar to the previous test case, we translate the wing

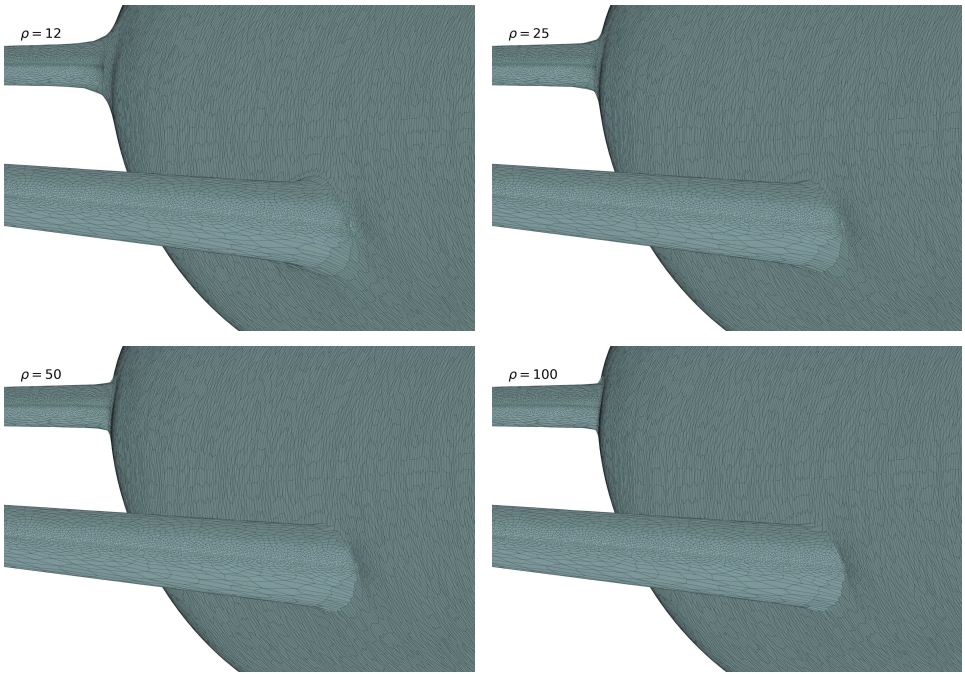


Fig. 5 For high values of smoothness parameter ρ, the intersection approaches a kink with a 90° angle while for relaxed values, a smooth fillet is created between intersecting components.

along the direction of the fuselage but also change the fuselage diameter simultaneously from 150% to 70% of the initial diameter. Figure 7 shows the corresponding mesh deformation results, with the fuselage diameter decreasing from top to bottom and left to right. We see that near the intersection of the wing and the fuselage, mesh nodes that lie on the wing move in the spanwise direction, driven by the outward and inward motion of the fuselage. Similar to the previous deformation test case, the mesh quality remains high overall.

V. Application to Large-Scale MDO

In this section we present the results of a large-scale MDO study, in which we couple the mesh deformation algorithm to a panel code developed by Scotzniovsky and Hwang [39]. The primary goal of this study is to demonstrate the mesh deformation algorithm in a practical MDO setting, consistent with a realistic aircraft design problem, not to find a more optimal design than the baseline E175 aircraft.

A. Problem Formulation

The optimization problem considers a single design point (cruise), which determines the fuel weight objective W_f based on the Breguet range equation, computed as

$$W_f = \left(1 - \exp\left(\frac{-R \cdot c_T \cdot g}{(L/D) \cdot V}\right)\right) \cdot W_0 \tag{19}$$

The range R is assumed to be 2,000 nm, the thrust specific fuel consumption c_T is held constant at 0.68 lb/lbf/h (taken from a representative General Electric CF34 turbofan engine), g is the gravitational acceleration, and V is the cruise speed, computed from the cruise Mach number ($M = 0.7$) and the speed of sound at an altitude of 25,000 ft. We compute the lift and induced drag using the panel code. To compute the parasite drag, we use semi-empirical drag build-up equations by Raymer [40], which are based on skin-friction coefficient scaling laws. Similarly, to compute the gross weight, we use empirical component weight relations also developed by Raymer [40]. The component center of gravity locations are tied to parametric locations of the central B-spline geometry of the baseline design and are automatically updated through the free form deformation (see Sec. II.A).

In Table 2, we summarize the optimization problem. The design variables are all geometric in nature, ranging from

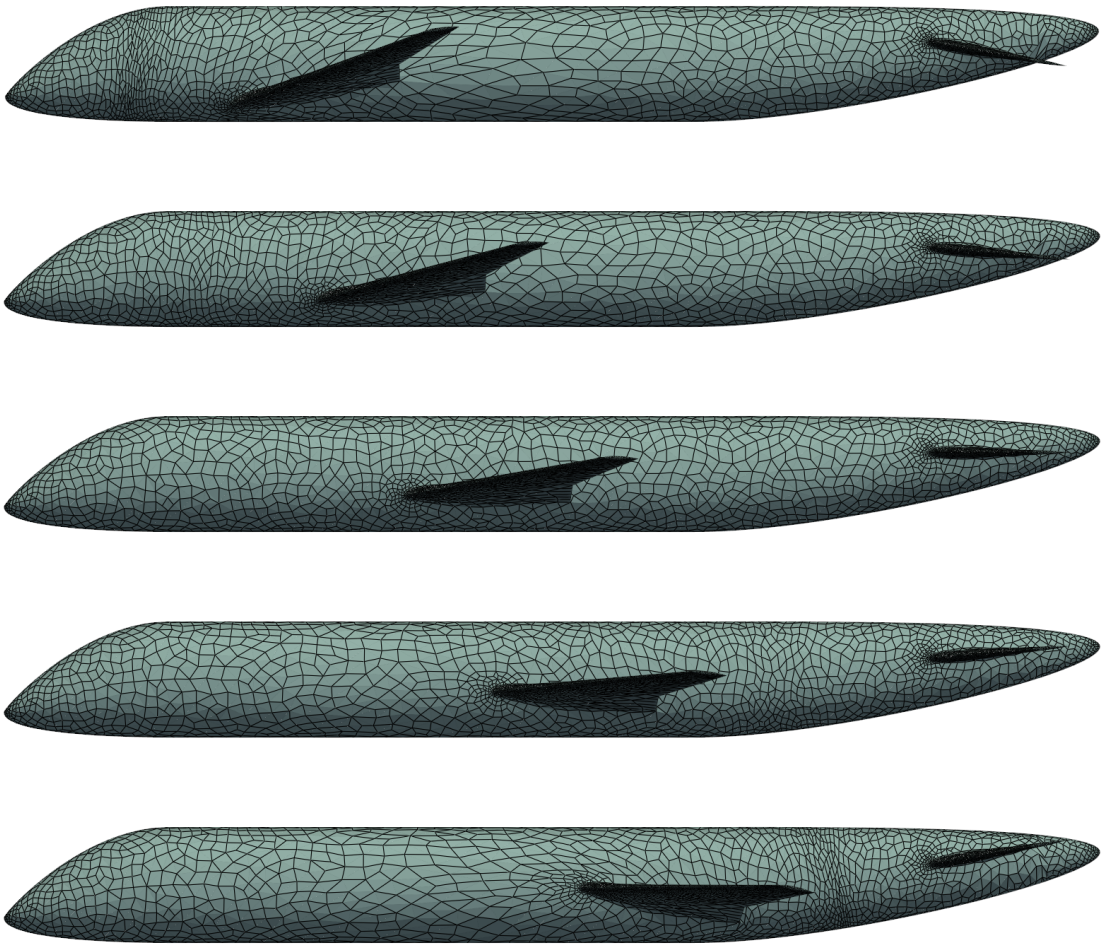


Fig. 6 Mesh deformation for simultaneous wing translation, rotation and tail rotation. The surface mesh smoothly deforms (consistent topology) as the wing is moved and rotated toward the nose and tail. As expected, mesh cells in front and aft of the wing become compressed or stretched depending on deformation. Mesh quality remains reasonable with no inverted or collapsed elements.

planform variables like wing area and aspect ratio to cross-sectional shape variables like twist and camber. As explained in Sec. II.A, we use an FFD approach with parameterized spanwise sections that allows us to directly map changes in the design variables to deformations of the embedded B-spline geometry. In terms of constraints, we enforce force and pitching moment equilibrium constraints as well as a minimum payload constraint, based on the ratio of the baseline design's payload to the fuselage volume, which we recompute from the changing geometry during optimization. In addition, in the absence of a physics-based structural model, we enforce a wing loading constraint. Furthermore, we find that the wing-loading constraint prevents the optimizer from driving the wing toward a design with an artificially small wing area due to the objective of minimizing fuel-burn, which is largely driven by reducing parasite (skin friction) drag via a smaller wetted wing area.

For static stability, we enforce a static margin constraint of 5%, where the static margin is the normalized (by MAC) distance between the neutral point and the center of gravity. We use a finite-difference approximation of the moment coefficient derivative based on the panel code solution to compute the neutral point. Maeder and Martins [6] provide a concise derivation of the static margin computation used in this paper. Finally, we place a constraint on the horizontal

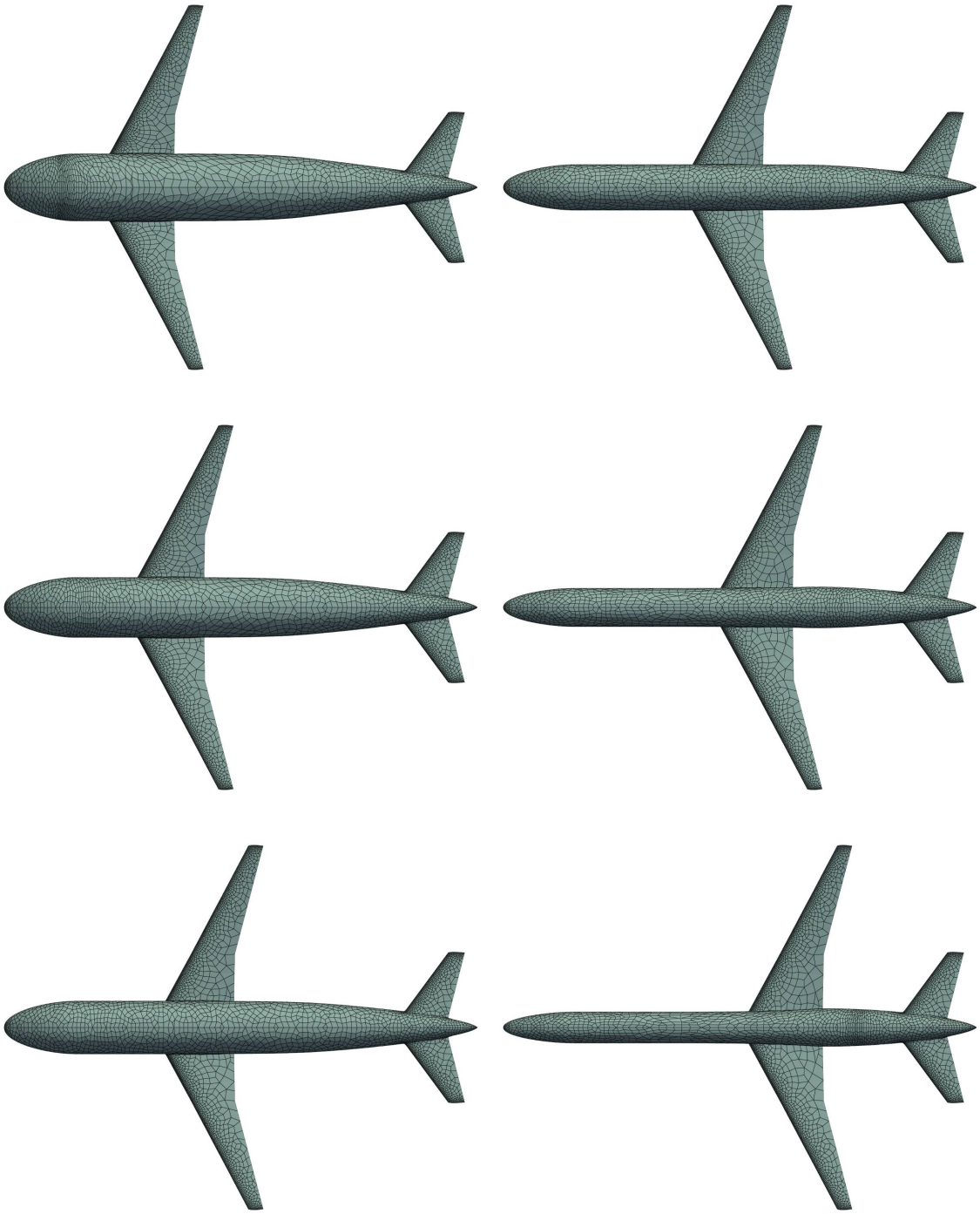


Fig. 7 Mesh deformation for simultaneous wing translation and fuselage diameter scaling. The mesh nodes near the wing-fuselage intersection smoothly deform as the wing is rotated. The effect of the fuselage diameter scaling is propagated to wing mesh nodes that lie in the vicinity of the intersection.

tail volume coefficient C_{V_T} as a proxy for longitudinal dynamic stability, defined as

$$C_{V_T} = \frac{S_T \cdot l}{S_W \cdot \mathrm{MAC}}, \tag{20}$$

where S_T, is the horizontal tail reference area, l is the tail moment arm, measured from the quarter chord of the wing to the horizontal tail (at MAC), and S_W is the wing reference area.

Table 2 We solve a fuel burn minimization problem, representative of early-stage conceptual aircraft design.

	Variable/function	**Quantity**	**Bounds/Constraints**
minimize	fuel weight W_f	1	–
w.r.t.	wing twist	8	$\pm 5°$
	wing camber	30	$\pm 5\%$
	wing area S_W (ft^2)	1	$560 \leq S_w \leq 940$
	wing aspect ratio AR_W	1	$6.3 \leq AR_w \leq 10.5$
	tail area S_T (ft^2)	1	$188 \leq S_t \leq 310$
	tail aspect ratio AR_T	1	$3.2 \leq AR_T \leq 5.4$
	wing translation Δx (ft)	1	± 16.4
	tail rotation	1	$\pm 8°$
	fuselage diameter D_f (ft)	1	$7.4 \leq D_f \leq 12.3$
	# design variables	45	
subject to	minimum payload weight W_p (lbf)	1	$W_p \geq 22050$
	force equilibrium	1	$L - W_0 = 0$
	moment equilibrium	1	$\sum C_{M_y} = 0$
	wing loading W_0/S_w (lbf/ft^2)	1	$60 \leq W_0/S_w \leq 100$
	tail volume coefficient C_{V_T}	1	$0.7 \leq C_{V_T} \leq 1.2$
	static margin (SM)	1	SM = 0.05
	# constraints	6	

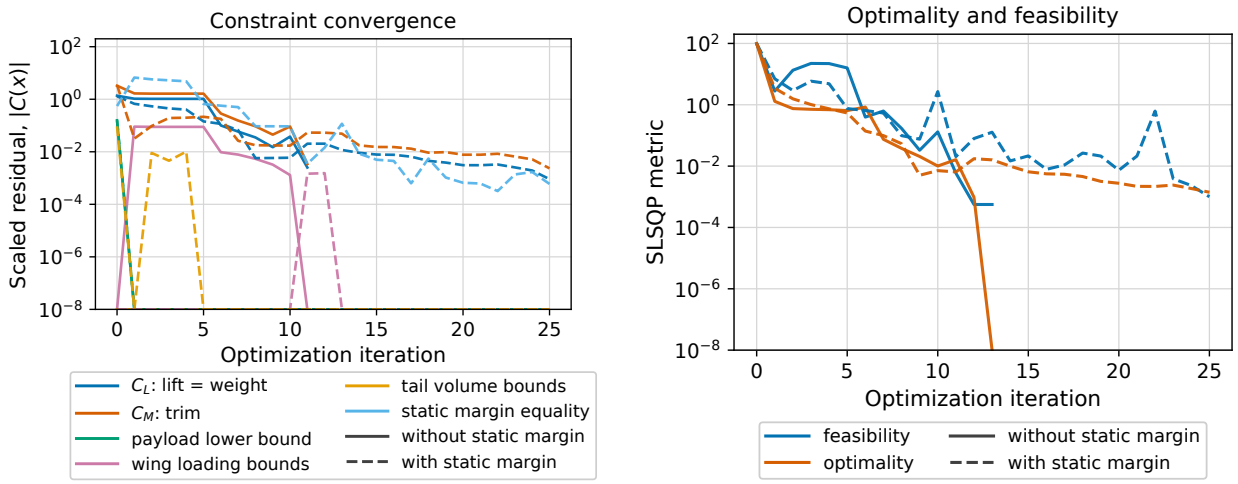
For this study, we use the Sequential Least Squares Quadratic Programming (SLSQP) algorithm, used through a modular Python-based interface [41]. We consider two variations of the optimization problem shown in Table 2, one in which we enforce a static margin constraint and one where we omit this constraint. In Fig. 8, we show the convergence of the optimization constraints as well as the optimality and feasibility convergence as reported by the optimizer. We see that both variations of the optimization problem converge to acceptable tolerances, with the static-margin-constrained problem taking about twice as many iterations to converge. It is noteworthy to point out that the net force and moment equilibrium residuals may seem large at the converged solution, however, their magnitudes are on the order of 10 lbf and 10 lbf-ft, respectively, which is negligible given the gross weight of the aircraft.

B. Summary of Results

Table 3 summarizes key geometric planform variables, the fuel and vehicle gross weights, the wing translation amount, as well as the wing loading, tail volume coefficient and static margin constraint values for the baseline and optimized designs. Compared to the baseline, both optimized designs converged to approximately the same reduced wing area of roughly 660 ft^2 and reached the upper bound of 10.5 set for the wing aspect ratio. The tail sizing results are similar, with the tail areas reaching the lower bound while the tail aspect ratio is slightly higher when including the static margin constraints.

Table 3 Results of the large-scale MDO study in terms of high-level design variables and constraints.

Design	W_0 (lbf)	W_f (lbf)	AR_W	S_W (ft^2)	AR_T	S_T (ft^2)	Δx (ft)	W_0/S_w (lbf/ft^2)	C_{V_T}	SM (%)
Baseline	288800	70400	8.3	740	4.3	250	0.0	90	1.16	-0.3
Optimized w/o SM	293000	57600	10.5	658	3.7	188	-1.0	100	1.18	-23.0
Optimized w/ SM	294500	58850	10.5	661	4.0	188	+3.7	100	1.20	5.0



(a) Constraint convergence

(b) Optimality and feasibility convergence

Fig. 8 Convergence history. The optimization problem converges robustly within 12-25 iterations, with the static margin constraint increasing the number of iterations required for convergence.

We show the optimized planform geometries overlaid with the baseline geometry in Fig. 9. As expected, the most noticeable differences between the baseline and optimized designs are the wing placement and corresponding static margins. For the baseline design at trim state, we find that the aircraft is marginally unstable, with a static margin of -0.3%. When excluding the static margin constraint, the optimizer moves the wing forward (toward the nose) by roughly 1 ft, resulting in a statically unstable configuration with a static margin of less than -20%. The payoff is a reduced fuel weight that is roughly 20% lower compared to the baseline design and about 3% lower than when including the static margin constraint. To achieve the prescribed static stability, the optimizer moves the wing aft (toward the fuselage tail) by over 3.5 ft. A likely explanation for this large required wing translation in order to achieve static stability is the crude empirical component weight build-up procedure and the approximation of the center of gravity locations of individual components, which are related to the central B-spline geometry and fixed in the parameter space. In addition, due to the wing sweep, the neutral point likely moves aft faster than the center of gravity, increasing the static margin. In terms of the wing loading and tail volume coefficient constraint, we observe that both optimized designs are driven to the upper bounds. This is a reasonable result because the fuel-weight minimization is largely driven by reducing (parasite) drag, which is crudely estimated based on the component skin friction coefficients and wetted areas. Due to the reduced wing area, the optimizer must compensate to achieve the trim and static margin constraints by increasing the tail volume coefficient.

Lastly, we examine the aerodynamic shape changes produced by the twist and camber design variables. Figure 10 shows the baseline and optimized airfoil sections at four spanwise locations. The optimized airfoils for the two problem formulations are similar, especially outboard, and show two notable trends. First, the optimized wings use positive incidence near the root and wash out toward the tip. Second, the optimizer introduces substantial negative camber, especially at the inboard sections. This result is consistent with the trim requirements and the modeling assumptions of the problem. For a conventional positively lifting airfoil, increasing positive camber generally increases lift at a given incidence but also tends to produce a stronger nose-down pitching moment. By introducing negative camber, the optimizer can reduce the nose-down pitching-moment contribution of the wing. At the same time, the optimizer can use twist and incidence to maintain the lift required by the force-equilibrium constraint. Thus, the optimized design uses camber and twist together: camber helps control pitching moment, while twist controls the sectional loading distribution.

Figure 11 shows the corresponding pressure contours. Compared with the baseline, both optimized designs exhibit stronger suction on the upper surface near the wing leading edge and increased pressure on the lower surface near the leading edge. These pressure changes indicate that the optimized wings remain strongly loaded despite the negative camber of the airfoil sections. This behavior is plausible within the panel-code model because the flow remains inviscid and attached by construction. Consequently, the optimizer can exploit large twist angles, negative camber, and strong leading-edge suction without incurring separation, stall, or airfoil-profile-drag penalties. The fuselage pressure contours

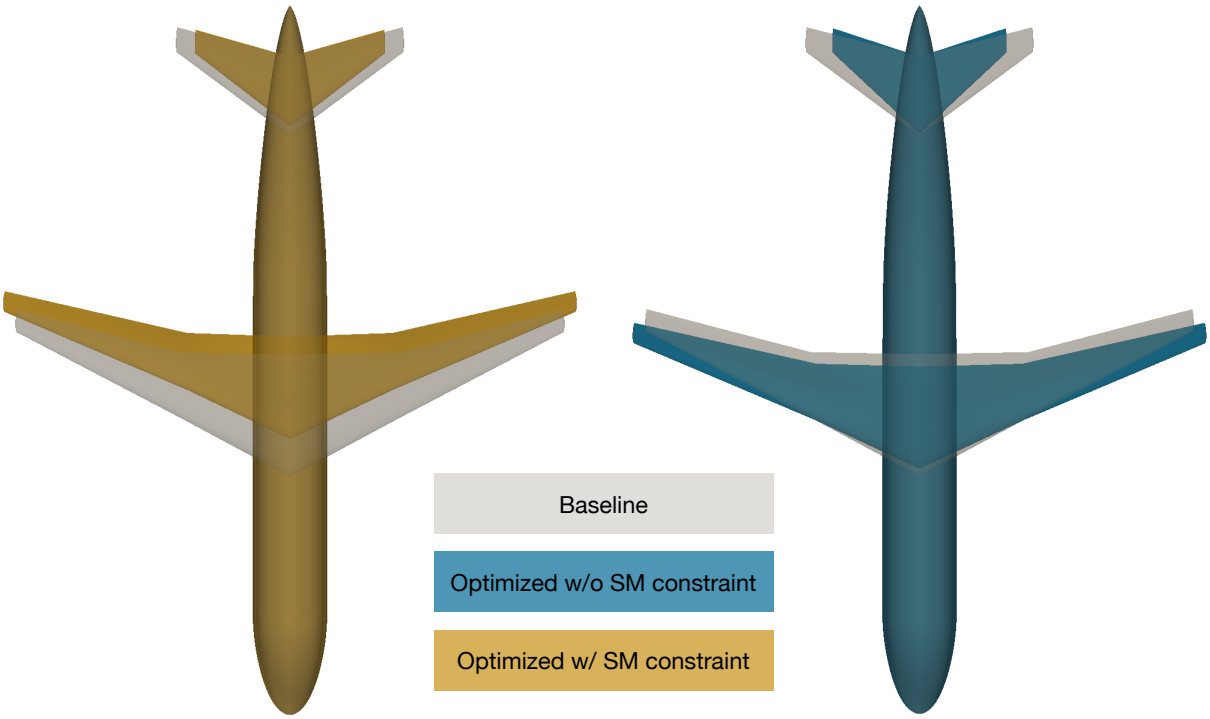


Fig. 9 Planform geometry comparison. Compared to the baseline geometry, the optimized geometry with the static margin constraint moves the wing backward and increases the wing span. In the case of the optimized design without the static margin constraint, the wing is moved slightly forward and has a slightly larger wing span compared to the baseline.

also follow the expected qualitative pattern, with a stagnation region near the nose and a pressure drop around the high-curvature cockpit region.

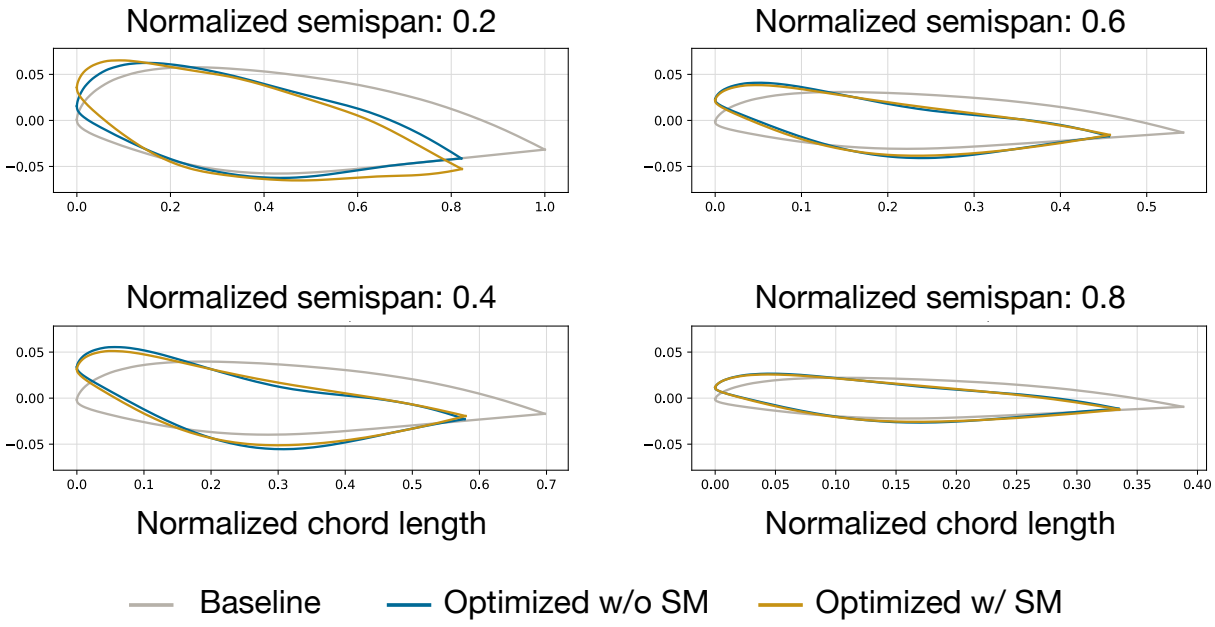


Fig. 10 Optimized airfoil shapes and wing twist at four spanwise locations.

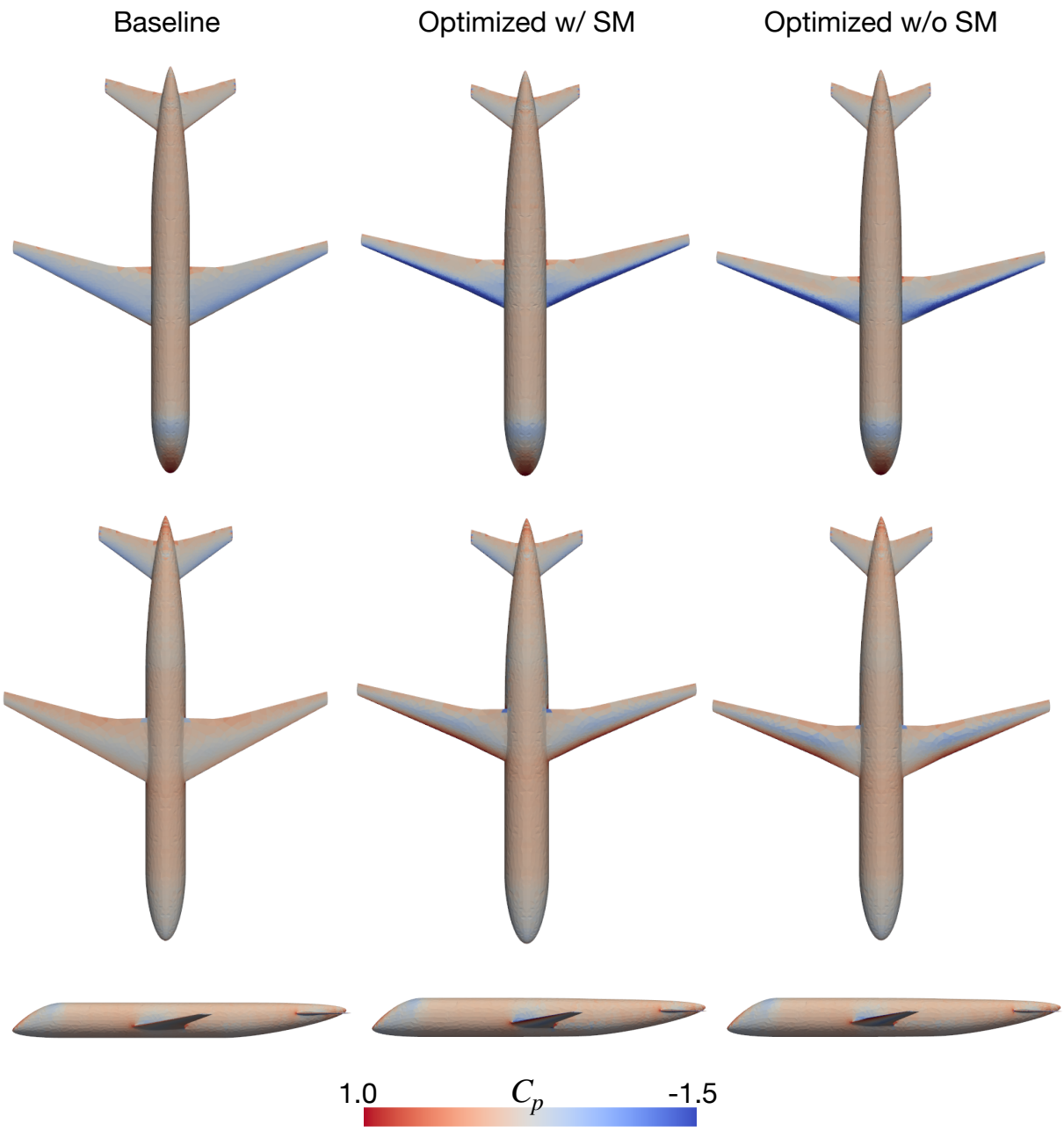


Fig. 11 Pressure contour comparison. Compared to the baseline design, the optimized designs with and without the static margin constraint feature a pressure profile with a significant increase in suction (top row) and an increase in pressure (middle row) near the leading edge.

C. Discussion and Limitations of Results

We interpret the optimization results as a demonstration of the proposed mesh deformation algorithm in a realistic MDO workflow, rather than as a closed redesign of the E175 aircraft. The optimization problem includes enough geometric freedom, constraints, and aerodynamic coupling to exercise the mesh deformation algorithm under practical design changes, but the disciplinary models remain simplified. Therefore, the optimized geometries should be interpreted as model-consistent designs that demonstrate algorithmic capability, not as design recommendations.

The static-margin results provide one example of this distinction. The baseline geometry is predicted to be marginally unstable in this model, and the no-static-margin optimization produces an even more unstable design. This result should

not be interpreted as a statement about the actual aircraft. The center-of-gravity locations come from an empirical component-weight buildup and are tied to parametric locations in the central B-spline geometry. The neutral point is computed from finite differences of the panel-code moment coefficient. Both quantities therefore contain modeling uncertainty. The large aft wing translation required to satisfy the prescribed static margin should be viewed in this context.

The aerodynamic shape results also reflect the assumptions of the aerodynamic and drag models. The panel code captures full-configuration inviscid pressure loading, but it does not model viscous separation, boundary-layer transition, shock formation, or stall. The parasite drag model is based on semi-empirical wetted-area and skin-friction estimates rather than local viscous airfoil drag. As a result, the optimizer can exploit high leading-edge suction, large twist, and negative camber without incurring the penalties that would appear in a viscous aerodynamic model. Thus, the optimized airfoil sections should be viewed as model-consistent outcomes that demonstrate shape sensitivity and mesh deformation robustness, rather than as final, realistic aerodynamic designs.

The problem also omits several disciplinary effects that would be required for a design-level conclusion. We do not include a structural or aeroelastic model, so the optimized wing planform and airfoil shapes are not constrained by stress, deflection, flutter, or manufacturing requirements. We also consider a single cruise design condition and do not enforce constraints due to takeoff, climb, low-speed stall margin, maneuver loads, or control authority. In addition, we model longitudinal static stability through a static-margin constraint and a tail-volume constraint, but we do not perform a dynamic stability or control analysis. The propulsion system and other subsystems are also simplified or omitted.

These limitations are acceptable for the purpose of this paper because the primary objective is to demonstrate that the proposed mesh deformation algorithm can support large-scale gradient-based optimization with a conforming full-configuration surface mesh. The results show that the algorithm can tolerate large planform changes, component translations, tail rotations, fuselage-diameter changes, wing twist, and wing camber while allowing the coupled optimization problem to converge. Increasing the physical fidelity of the optimization problem is a natural next step. Future work should couple the mesh deformation algorithm to higher-fidelity viscous aerodynamics, structural and aeroelastic models, propulsion models, and multiple design conditions. These extensions would reduce the modeling freedom that currently allows the optimizer to exploit inviscid and empirical-model assumptions, and would move the optimization results closer to design-level conclusions.

VI. Conclusion

Large-scale MDO is moving toward increasingly complex system-level aircraft design problems with more design variables, more design conditions, and higher-fidelity disciplinary models earlier in the design cycle. At the conceptual level, this trend requires predictive models that can tolerate large changes in geometry while remaining efficient and differentiable. This requirement is especially important for aerodynamics because changes in the outer mold line directly affect the aerodynamic discretization and predicted flow field. Low-fidelity aerodynamic models avoid many of these geometric difficulties, but their limited accuracy reduces their usefulness for full-configuration design studies. Higher-fidelity aerodynamic models, such as panel methods and unstructured CFD, can provide more realistic full-configuration predictions, but they typically require conforming surface meshes that remain on the outer hull as the geometry changes. For multi-component aircraft configurations, this requirement becomes difficult because component intersections can move significantly and may even change local intersection topology during optimization. Conventional mesh smoothing and deformation methods are not generally designed for this setting, and many MDO workflows either rely on localized surface updates, explicit intersection recomputation, or repeated remeshing, which can introduce robustness, differentiability, and file-I/O bottlenecks.

In this paper, we develop a differentiable surface mesh deformation algorithm for large shape changes of independently parameterized components. This algorithm combines explicit and implicit (SDF) geometry representations, where the explicit parametric geometry defines component motion and deformation through the use of free form deformation (FFD). The implicit representation defines a smooth OML and provides a projection mechanism that avoids repeatedly computing explicit intersection curves. The main methodological contribution is the construction of intersection-consistent training displacements using a soft-bracketed root-finding procedure. This procedure moves seam vertices by searching along the explicit parametric representation of one component while using the SDF of the intersecting component to locate the deformed seam. We then blend these seam displacements with component-based displacements near intersections, fit a radial basis function (RBF) displacement field surrogate to propagate the motion to the full mesh, and project the displaced vertices onto a smooth SDF union of the OML. The soft-min projection produces a rounded, continuous transition near component intersections, allowing near-intersection vertices to move along a smooth OML representation

rather than being discontinuously mapped to a single component surface.

We demonstrate the method on a representative Embraer 175 geometry undergoing large component translations, rotations, and scaling. The deformation results show that the algorithm preserves mesh quality under severe geometric changes and avoids element collapse or inversion, with fewer than 0.14% of cells exceeding an aspect ratio of 5 when translating the wing by ± 1.5 mean aerodynamic chord lengths along the fuselage while simultaneously applying a wing rotation of $\pm 10^\circ$. We then couple the mesh deformation algorithm to a 3D panel code and solve a large-scale fuel-burn-minimization problem with 45 geometric design variables, including planform, component-placement, wing-twist, and wing-camber variables. The optimization results should be interpreted as a demonstration of the mesh deformation algorithm in a practical MDO workflow rather than as a redesign of the reference aircraft. Within this context, the results show that the proposed algorithm can support full-configuration aerodynamic shape optimization with a conforming surface mesh while allowing substantial geometry changes during optimization.

This work addresses a geometric bottleneck that limits the use of higher-fidelity aerodynamic models in conceptual aircraft MDO. By enabling conforming full-configuration surface meshes to deform robustly and differentiably under large shape changes, the proposed method provides a step toward using more informative aerodynamic models earlier in the design process. Such capability can reduce conceptual-design uncertainty, improve the interpretability of optimization results, and help identify geometry-driven design issues before later, more expensive stages of aircraft development.

Several limitations remain. The current implementation computes component SDF values using projection-based distance evaluations, which can become expensive for large CFD meshes or repeated union-SDF calls. In addition, to evaluate total derivatives of the system (outer) model, the final SDF projection also requires derivatives of SDF gradients with respect to geometry variables; for projection-defined SDFs, this introduces second-order adjoint or mixed-derivative requirements that may become a computational bottleneck. A promising direction is to train SDF surrogates for repeated evaluations and use an analytic or projection-based correction only near the final zero-level projection. The soft-bracketed seam search also assumes that the deformed intersection remains inside the stored bracket and that the relevant target components remain in the candidate set. If large geometry changes violate these assumptions, the setup data must be regenerated.

Future work should extend the demonstration to more complex configurations, including novel concepts such as eVTOL aircraft with booms and multiple interacting lifting surfaces. We also plan to test topology-changing cases in which local intersection ownership changes during optimization, since the smooth SDF-union formulation supports such changes in principle. Finally, the method should be coupled to higher-fidelity viscous aerodynamic, structural, aeroelastic, propulsion, and multi-point mission models. These extensions would reduce the modeling assumptions used in the present demonstration and move the approach closer in scope to system-level optimization studies.

Appendix

A. Supplemental Figures for Mesh Quality Metrics

Figure 12 shows several mesh quality metrics for the deformation case shown in Fig. 6. The results suggest that even under extreme deformation, the overall mesh quality remains acceptable.

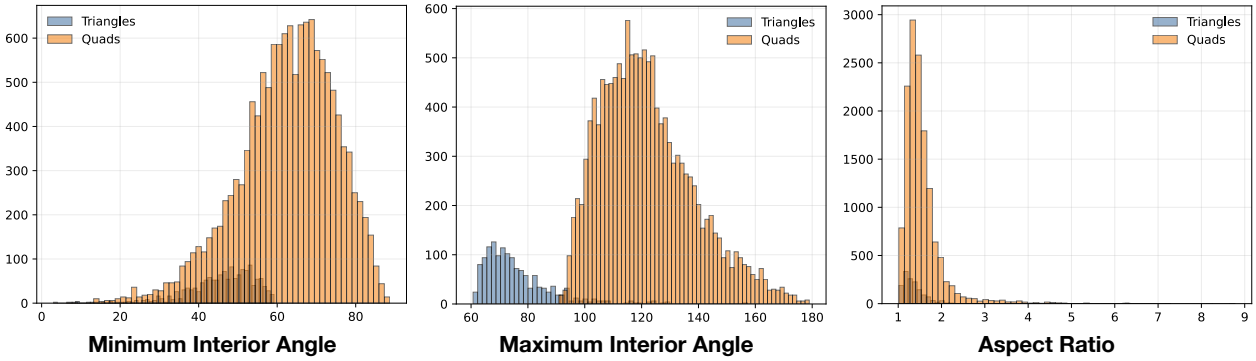
Acknowledgments

We thank Luca Scotzniovsky for his help in connecting the panel code to the mesh deformation algorithm. We also acknowledge the use of generative AI tools in preparation of this work. ChatGPT (OpenAI) assisted in proofreading and language editing, and Codex (OpenAI) assisted in implementing portions of the research code used to generate the results. The development and design of the algorithm, formulation of the test problem, and interpretation of the results were done entirely by the authors.

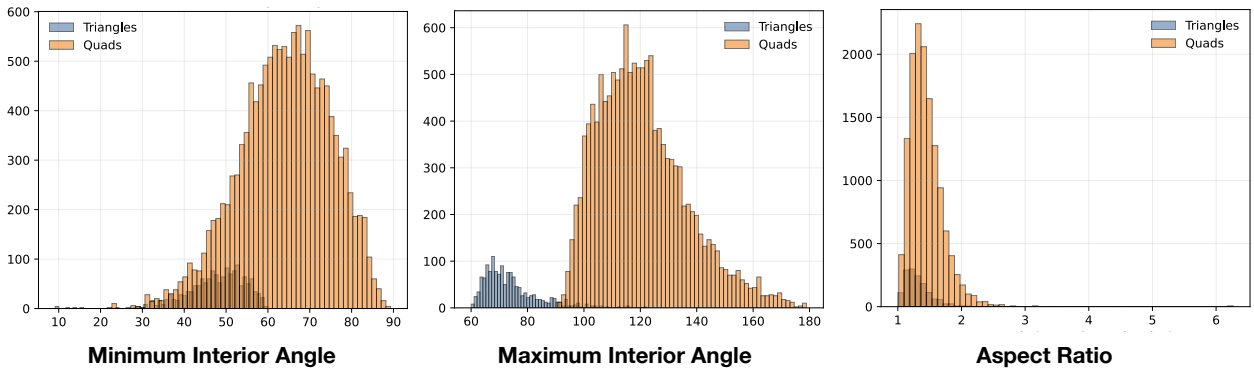
The second author was supported by NASA under award No. 80NSSC23M0217.

References

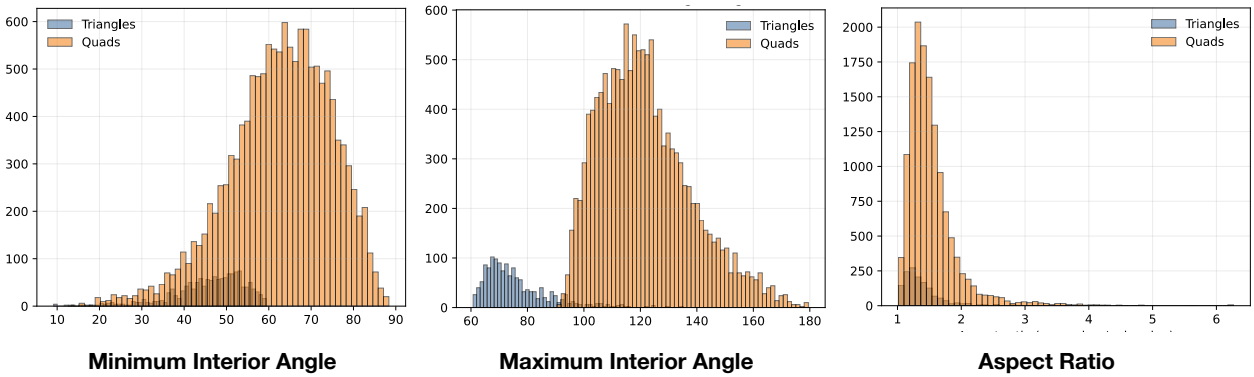
[1] Gandarillas, V., Joshy, A. J., Sperry, M. Z., Ivanov, A. K., and Hwang, J. T., "A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis," *Structural and Multidisciplinary Optimization*, Vol. 67, No. 5, 2024. https://doi.org/10.1007/s00158-024-03792-0.



(a) Extreme deformation 1: Wing translation: -1.5 MAC, wing rotation: $-10°$, tail rotation: $+10°$



(b) Undeformed mesh



(c) Extreme deformation 2: Wing translation: +1.5 MAC, wing rotation: $+10°$, tail rotation: $-10°$

Fig. 12 Mesh quality metrics for extremely warped and baseline meshes.

[2] Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., and Naylor, B. A., "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104.

[3] Kenway, G., Kennedy, G., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, American Institute of Aeronautics and Astronautics, 2010.

[4] Kenway, G. K. W., and Martins, J. R. R. A., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, 2014, p. 144–160. https://doi.org/10.2514/1.c032150.

[5] Gray, A. C., Riso, C., Jonsson, E., Martins, J. R. R. A., and Cesnik, C. E. S., "High-Fidelity Aerostructural Optimization with a Geometrically Nonlinear Flutter Constraint," *AIAA Journal*, Vol. 61, No. 6, 2023, p. 2430–2443. https://doi.org/10.2514/1.j062127.

[6] Mader, C. A., and Martins, J. R. R. A., "Stability-Constrained Aerodynamic Shape Optimization of Flying Wings," *Journal of Aircraft*, Vol. 50, No. 5, 2013, p. 1431–1449. https://doi.org/10.2514/1.c031956.

[7] Gagnon, H., and Zingg, D. W., "High-fidelity Aerodynamic Shape Optimization of Unconventional Aircraft through Axial Deformation," *52nd Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, 2014. https://doi.org/10.2514/6.2014-0908.

[8] Ruh, M. L., Warner, M. A. P., Scotzniovsky, L., Fletcher, A. H., Sperry, M. Z., and Hwang, J. T., "System-Level, Large-Scale Multidisciplinary Design Optimization of an Air Taxi Concept," *Journal of Aircraft*, 2026, p. 1–24. https://doi.org/10.2514/1.c038533.

[9] Critchfield, T., and Ning, A., "Mission-Focused Multidisciplinary Design Optimization of Tilt-Rotor eVTOL Propulsion System," *Journal of Aircraft*, 2026, p. 1–16. https://doi.org/10.2514/1.c038445.

[10] Venkataraman, S., and Haftka, R., "Structural optimization complexity: what has Moore's law done for us?" *Structural and Multidisciplinary Optimization*, Vol. 28, No. 6, 2004, p. 375–387. https://doi.org/10.1007/s00158-004-0415-y.

[11] Martins, J. R. R. A., and Kennedy, G. J., "Enabling large-scale multidisciplinary design optimization through adjoint sensitivity analysis," *Structural and Multidisciplinary Optimization*, Vol. 64, No. 5, 2021, p. 2959–2974. https://doi.org/10.1007/s00158-021-03067-y.

[12] Raymer, D. P., *Aircraft Design: A Conceptual Approach*, 6th ed., American Institute of Aeronautics and Astronautics, 2018.

[13] Roskam, J., *Airplane Design: Part I–VIII*, DARcorporation, 2010.

[14] Anderson, J. D., *Fundamentals of Aerodynamics*, 5th ed., McGraw-Hill, 2010.

[15] Katz, J., and Plotkin, A., *Low-Speed Aerodynamics*, 2nd ed., Cambridge University Press, 2001.

[16] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, Vol. 54, No. 3, 2016, p. 828–846. https://doi.org/10.2514/1.j053813.

[17] Field, D. A., "Laplacian smoothing and Delaunay triangulations," *Communications in Applied Numerical Methods*, Vol. 4, No. 6, 1988, pp. 709–712.

[18] Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28, No. 8, 1990, pp. 1381–1388. https://doi.org/10.2514/3.25229.

[19] Rendall, T. C. S., and Allen, C. B., "An efficient radial basis function method for mesh deformation within aerodynamic optimization," *International Journal for Numerical Methods in Engineering*, Vol. 74, No. 2, 2008, pp. 245–265.

[20] Haimes, R., and Dannenhoffer, J., "The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry," *21st AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, 2013. https://doi.org/10.2514/6.2013-3073.

[21] McDonald, R. A., and Gloudemans, J. R., "Open Vehicle Sketch Pad: An Open Source Parametric Geometry and Analysis Tool for Conceptual Aircraft Design," *AIAA SCITECH 2022 Forum*, American Institute of Aeronautics and Astronautics, 2022. https://doi.org/10.2514/6.2022-0004.

[22] Hwang, J., and Martins, J. R. R. A., "GeoMACH: Geometry-Centric MDAO of Aircraft Configurations with High Fidelity," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 2012. https://doi.org/10.2514/6.2012-5605, URL http://dx.doi.org/10.2514/6.2012-5605.

[23] Hajdik, H. M., Yildirim, A., Wu, E., Brelje, B. J., Seraj, S., Mangano, M., Anibal, J. L., Jonsson, E., Adler, E. J., Mader, C. A., Kenway, G. K. W., and Martins, J. R. R. A., "pyGeo: A geometry package for multidisciplinary design optimization," *Journal of Open Source Software*, Vol. 8, No. 87, 2023, p. 5319. https://doi.org/10.21105/joss.05319.

[24] Fletcher, A. H., and Hwang, J. T., "Implicit Nonlinear Geometry Parameterization for Multidisciplinary Design Optimization," *Proceedings of the ASME 2026 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE2026*, 2026. Accepted.

[25] Vasu, S., Talabot, N., Lukoianov, A., Baqué, P., Donier, J., and Fua, P., "HybridSDF: Combining Deep Implicit Shapes and Geometric Primitives for 3D Shape Representation and Manipulation," *2022 International Conference on 3D Vision (3DV)*, IEEE, 2022, p. 617–626. https://doi.org/10.1109/3dv57658.2022.00072, URL http://dx.doi.org/10.1109/3dv57658.2022.00072.

[26] Marschner, Z., Sellan, S., Liu, H.-T. D., and Jacobson, A., "Constructive Solid Geometry on Neural Signed Distance Fields," *SIGGRAPH Asia 2023 Conference Papers*, ACM, 2023, p. 1–12. https://doi.org/10.1145/3610548.3618170, URL http://dx.doi.org/10.1145/3610548.3618170.

[27] Ericson, C., *Real-Time Collision Detection*, CRC Press, 2004. https://doi.org/10.1201/b14581, URL http://dx.doi.org/10.1201/b14581.

[28] Blom, F. J., "Considerations on the Spring Analogy," *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 6, 2000, pp. 647–668. https://doi.org/10.1002/(SICI)1097-0363(20000330)32:6<647::AID-FLD979>3.0.CO;2-K.

[29] Zeng, D., and Ethier, C. R., "A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains," *Finite Elements in Analysis and Design*, Vol. 41, No. 11-12, 2005, p. 1118–1139. https://doi.org/10.1016/j.finel.2005.01.003.

[30] Field, D. A., "Laplacian Smoothing and Delaunay Triangulations," *Communications in Applied Numerical Methods*, Vol. 4, No. 6, 1988, pp. 709–712.

[31] Karman, S. L., Anderson, W. K., and Sahasrabudhe, M., "Mesh Generation Using Unstructured Computational Meshes and Elliptic Partial Differential Equation Smoothing," *AIAA Journal*, Vol. 44, No. 6, 2006, pp. 1277–1286.

[32] Stein, K., Tezduyar, T. E., and Benney, R., "Automatic Mesh Update with the Solid-Extension Mesh Moving Technique," *Computer Methods in Applied Mechanics and Engineering*, Vol. 193, No. 21–22, 2004, pp. 2019–2032.

[33] Tonon, P., Sanches, R. A. K., Takizawa, K., and Tezduyar, T. E., "A Linear-Elasticity-Based Mesh Moving Method with No Cycle-to-Cycle Accumulated Distortion," *Computational Mechanics*, Vol. 67, 2021, pp. 413–434.

[34] de Boer, A., van der Schoot, M. S., and Bijl, H., "Mesh Deformation Based on Radial Basis Function Interpolation," *Computers & Structures*, Vol. 85, No. 11–14, 2007, pp. 784–795. https://doi.org/10.1016/j.compstruc.2007.01.013.

[35] Rendall, T. C. S., and Allen, C. B., "Efficient Mesh Motion Using Radial Basis Functions with Data Reduction Algorithms," *Journal of Computational Physics*, Vol. 228, No. 17, 2009, pp. 6231–6249. https://doi.org/10.1016/j.jcp.2009.05.013.

[36] Yildirim, A., Mader, C. A., and Martins, J. R. R. A., "A Surface Mesh Deformation Method Near Component Intersections for High-Fidelity Design Optimization," *Engineering with Computers*, 2021. https://doi.org/10.1007/s00366-020-01247-w.

[37] Ruh, M. L., and Hwang, J. T., "Differentiable Surface Mesh Deformation for Non-Conformal, Independently Parameterized Components with Large Shape Changes," *AIAA AVIATION FORUM AND ASCEND 2025*, American Institute of Aeronautics and Astronautics, 2025. https://doi.org/10.2514/6.2025-3683.

[38] Ruh, M. L., Warner, M. A. P., and Hwang, J. T., "Toward Large-Scale Multidisciplinary Design Optimization of Aircraft With CFD and Mixed-Fidelity System Models," *AIAA SCITECH 2025 Forum*, American Institute of Aeronautics and Astronautics, 2025.

[39] Scotzniovsky, L., and Hwang, J. T., "A Fast, Memory-Efficient Panel Method for Large-Scale Multidisciplinary Design Optimization Under Uncertainty Using Graph-Based Modeling," *AIAA AVIATION FORUM AND ASCEND 2025*, American Institute of Aeronautics and Astronautics, 2025. https://doi.org/10.2514/6.2025-3021.

[40] Raymer, D., *Aircraft Design: A Conceptual Approach, Fifth Edition*, American Institute of Aeronautics and Astronautics, Inc., 2012. https://doi.org/10.2514/4.869112, URL http://dx.doi.org/10.2514/4.869112.

[41] Joshy, A. J., and Hwang, J. T., "PySLSQP: A transparent Python package for the SLSQP optimization algorithm modernized with utilities for visualization and post-processing," *Journal of Open Source Software*, Vol. 9, No. 103, 2024, p. 7246. https://doi.org/10.21105/joss.07246.