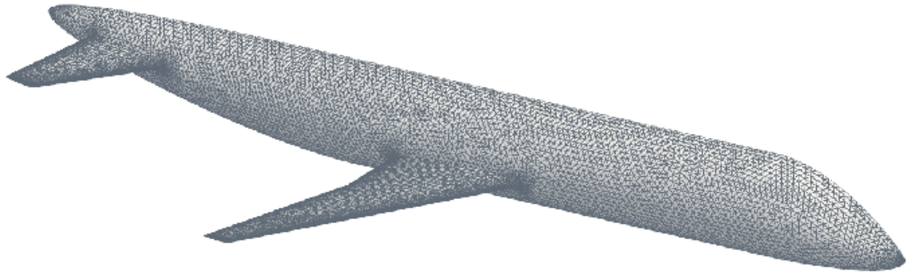
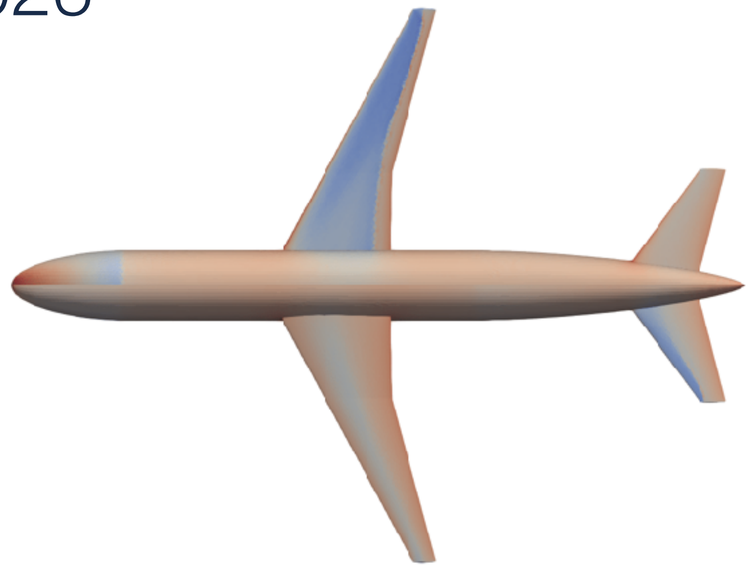
Differentiable Surface Mesh Deformation for Large-Scale MDO of Aircraft Concepts

Advanced Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, August 20th, 2026





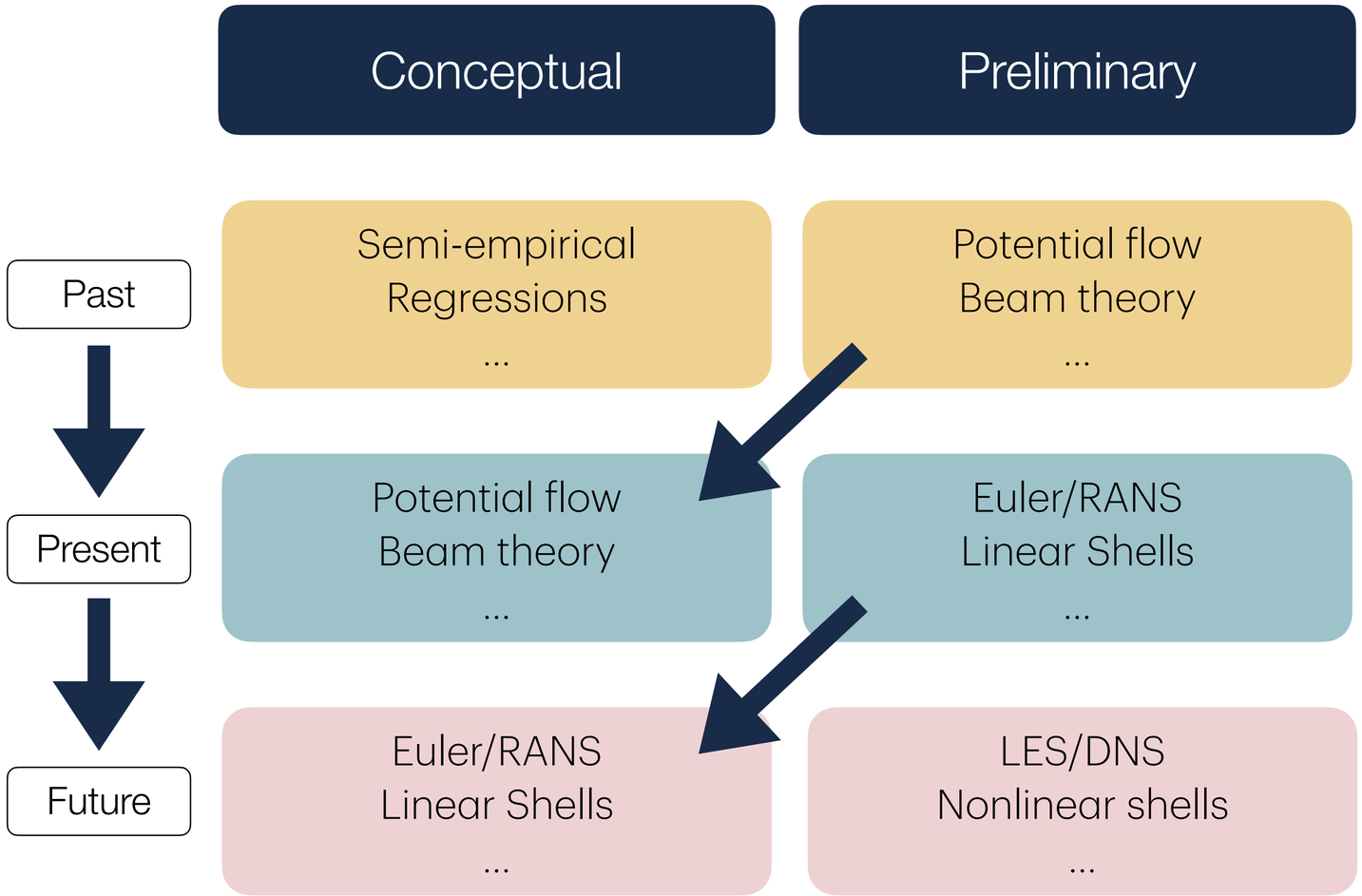
Marius Ruh, PhD Candidate
UC San Diego

Agenda

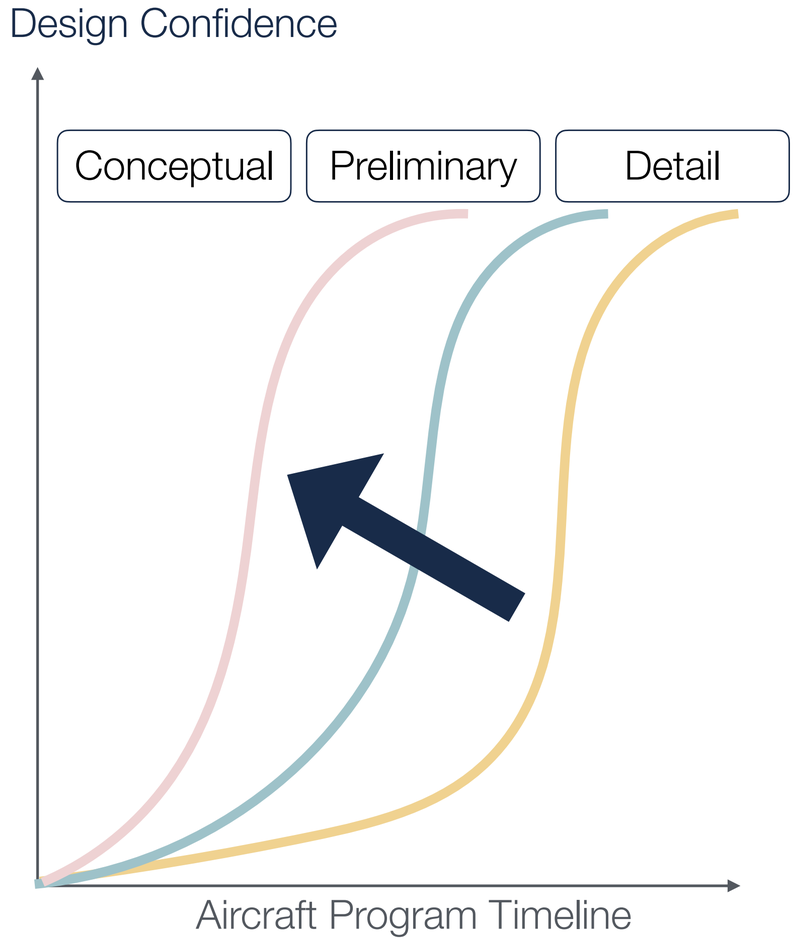
- An Academic Perspective on MDAO in Modern Aircraft Conceptual Design
- Why Surface Mesh Deformation Can Be a Bottleneck
- A Novel Differentiable Surface Mesh Deformation Algorithm
- Mesh Deformation Results for Conceptual-Level Design Changes
- MDO Demonstration Results
- Conclusion & Future Outlook

An Academic Perspective on MDAO in Modern Aircraft Conceptual Design

Advances in modeling, simulation & computing hardware have accelerated aircraft design

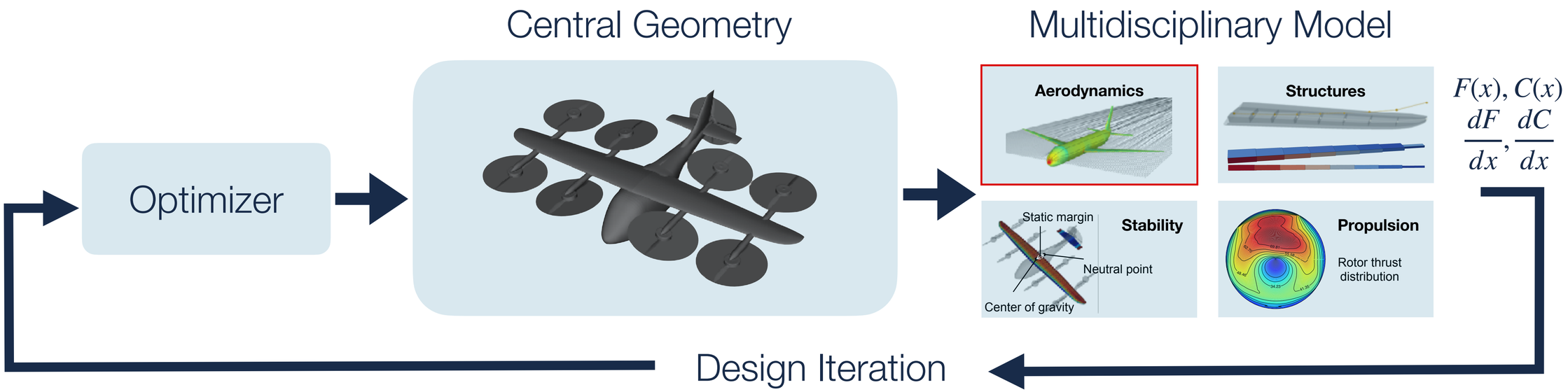


Notional diagram showing increasing fidelity at the conceptual level as computational methods advance



Multidisciplinary design analysis and optimization can enhance conceptual design

- Exploiting gradient information allows for efficient exploration of high-dimensional design spaces.
- Adjoint-based sensitivity analysis is crucial for solving large-scale MDO problems.
- MDO methods can help conceptual design engineers to rapidly evaluate the potential of many different configurations.



MDO research can be grouped by problem scale, scope, and modeling fidelity

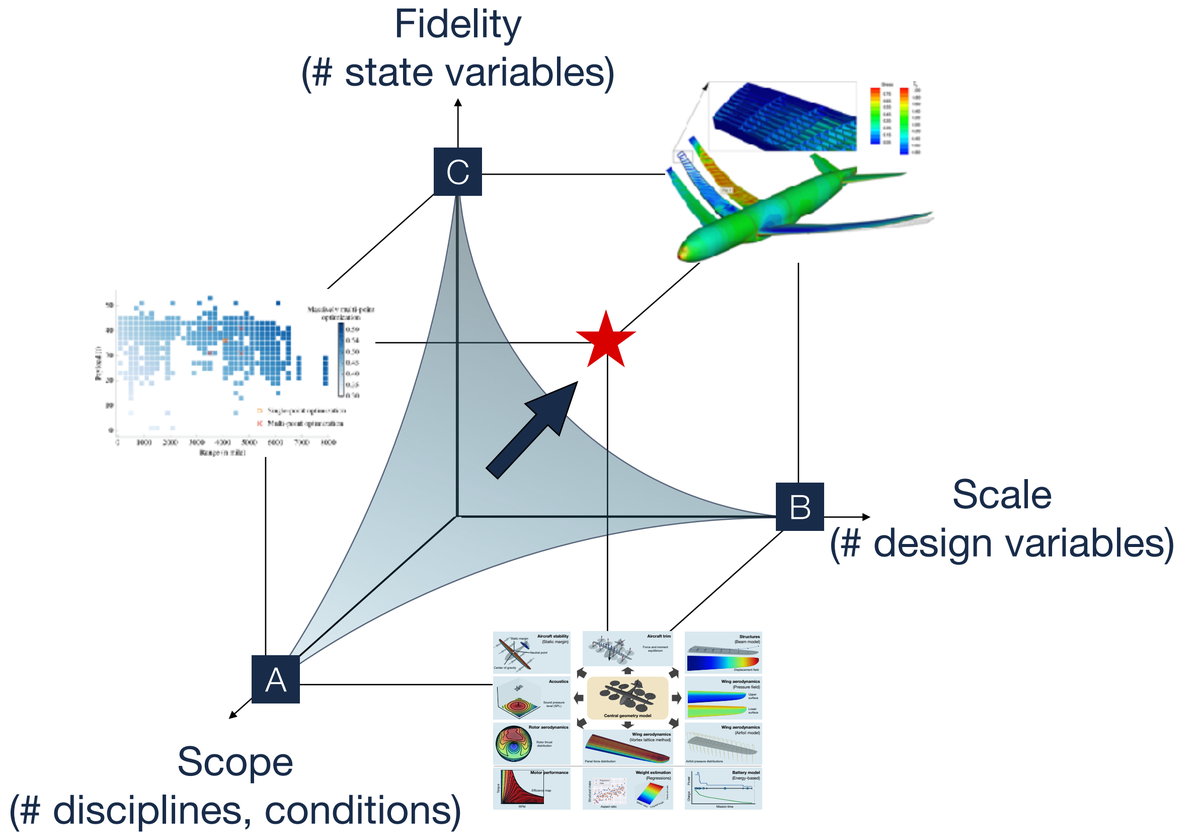
- The MDO community is moving toward solving problems of industrial complexity (★).

  A. **Scope**: all major disciplines with 10s-100s of design conditions

  B. **Scale**: 100s-1,000s of design variables representative of all major subsystems

  C. **Fidelity**: increasing number of state variables to capture complex physics

- My recent research focuses on enabling higher-fidelity aerodynamic analysis for comprehensive, system-level problems.



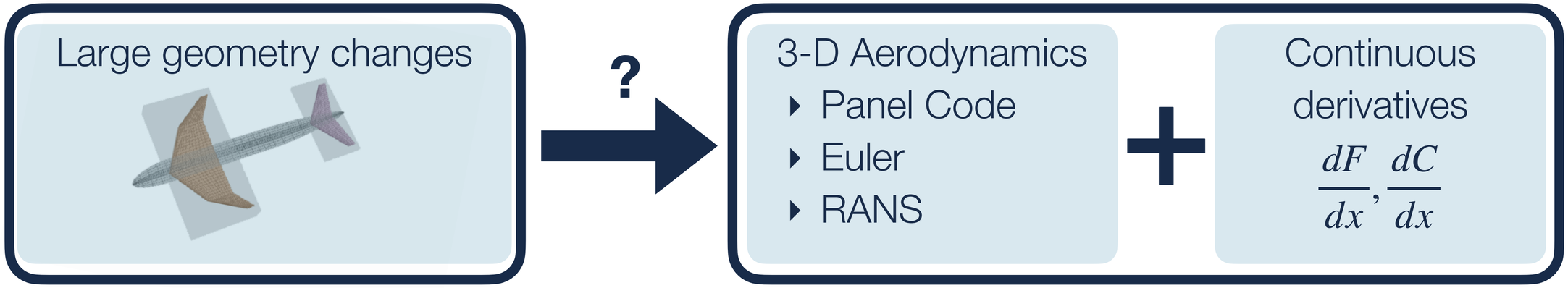
A–B: Ruh, Marius L., et al. "System-Level, Large-Scale Multidisciplinary Design Optimization of an Air Taxi Concept." Journal of Aircraft (2026): 1-24.

B–C: Kenway, Gaetan, Graeme Kennedy, and Joaquim RRA Martins. "A CAD-free approach to high-fidelity aerostructural optimization." 13th AIAA/ISSMO multidisciplinary analysis optimization conference. 2010.

C–A: Li, Jichao, and Jinsheng Cai. "Massively multipoint aerodynamic shape design via surrogate-assisted gradient-based optimization." AIAA Journal 58.5 (2020): 1949-1963.

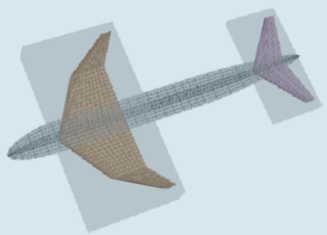
This work aims to increase the aerodynamic modeling fidelity at the conceptual level

- At the conceptual level, studying the effect of large design changes is important.
- This is straightforward for semi-empirical and low-fidelity methods (e.g., vortex lattice), which only require simple solver discretizations. However, they neglect important flow physics.
- Many higher-fidelity, 3-D aerodynamic models require conforming surface meshes, making it challenging to explore large geometric design changes.



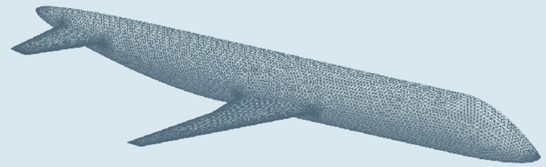
The objective is to develop a surface mesh deformation algorithm for conceptual design

Large Geometry Changes
+
Simple 2-D, component-level solver discretizations



?

Large Geometry Changes
+
Full-configuration, 3-D body-fitted solver discretizations



Why Surface Mesh Deformation Can Be a Bottleneck

Surface mesh deformation is a bottleneck for using 3-D aerodynamics for conceptual design

Geometry

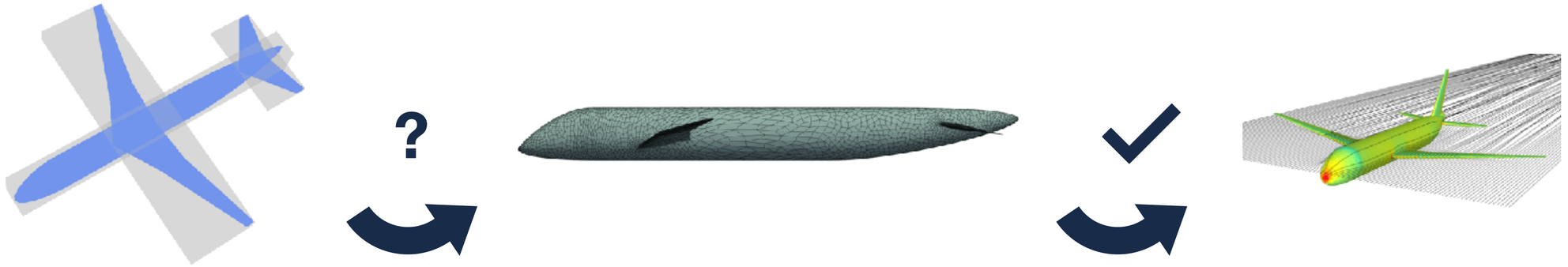
- **Independently parameterized component geometries** are advantageous at the conceptual level
- Well-established approaches (e.g., CAD surfaces + FFD) and mature tools (e.g., OpenVSP, ESP, …)
- This work uses B-spline surfaces embedded in component FFD blocks, automatically differentiated via CSDL*

Surface Mesh Deformation

- A conforming surface mesh of multi-component geometries **requires computing intersections**
- CAD and meshing tools can provide an initial mesh of the intersected, undeformed geometry
- But: *For large geometry changes, how to deform the surface mesh differentiably (i.e., w/o remeshing)?*

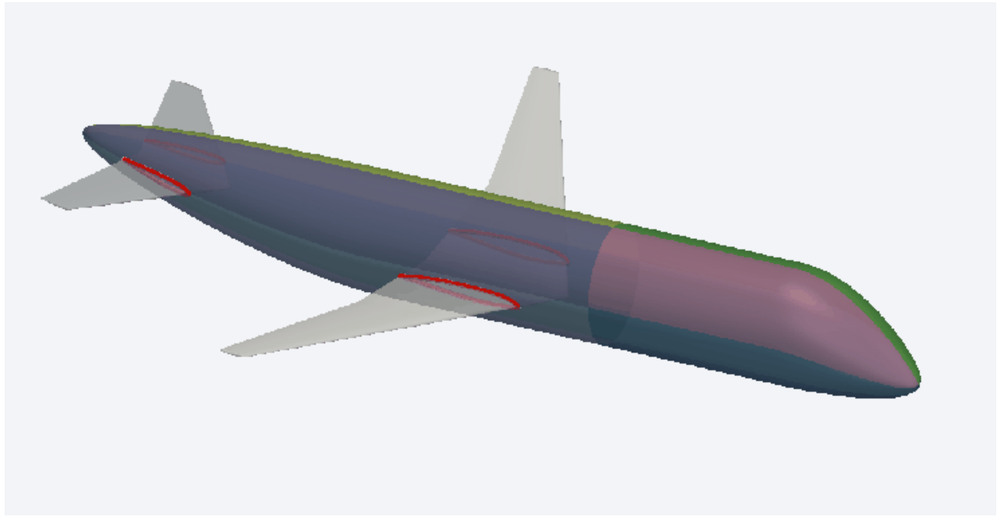
3-D Aerodynamics

- Well-established methods and software across the fidelity spectrum
- Low: Panel codes (Mach Line, VortexAD)
- Mid-fidelity: Euler (OpenFOAM, ADFlow, SU2, …)
- High-fidelity: RANS
- Most of these code bases **provide analytical sensitivities**



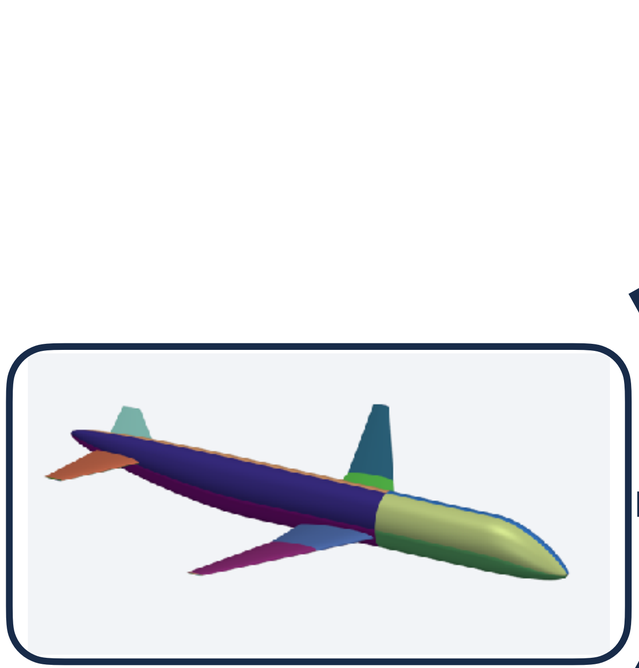
A key challenge is to accurately (re)-compute the motion of component intersections

- Surface meshes must conform to how intersections change across design iterations.
- Several challenges must be addressed for conceptual design:
  - **Robustness**: intersections may cross parametric surface patches
  - **Efficiency**: computation time should be negligible compared to the aerodynamic solver
  - **Differentiability**: gradients should be smooth
- Prior work does not simultaneously address all three challenges.

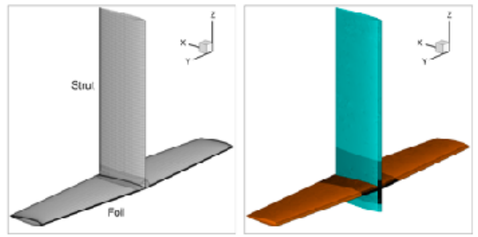


The computation of component intersections must be robust for a wide range of geometric changes, including across parametric surface boundaries.

Prior work on component intersections focuses on the computation of 1-D closed curves

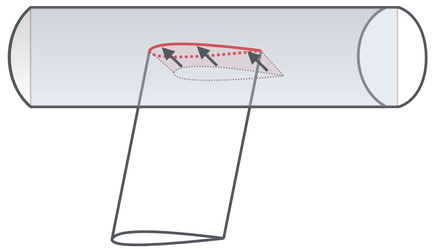


CAD geometry of independently parameterized components



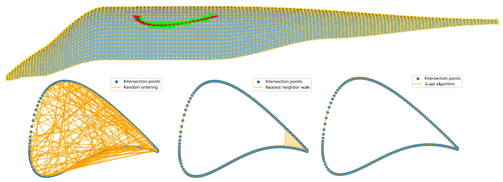
Triangulation of components with high-resolution near intersections

[1]



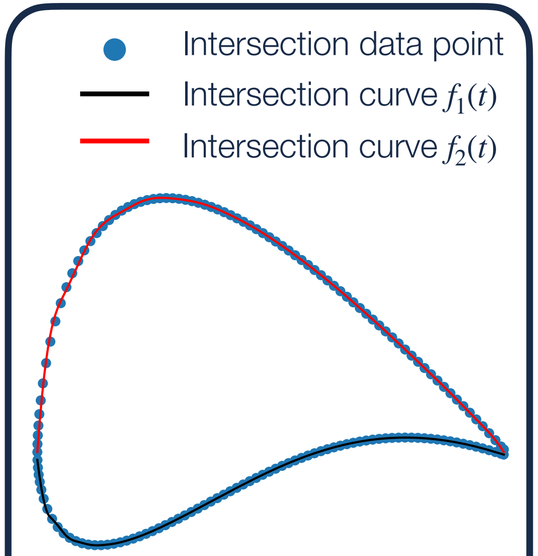
Minimization of virtual projection-error area

[2]



Intersection point cloud + ordering heuristics

[3]



Functional representation of 1-D, closed intersection curves

[1] Yildirim, A., Mader, C. A., & Martins, J. R. (2022). A surface mesh deformation method near component intersections for high-fidelity design optimization. *Engineering with Computers*, *38*(Suppl 2), 1393-1425.

[2] Ruh, M. L., Warner, M., & Hwang, J. T. (2025). Toward large-scale multidisciplinary design optimization of aircraft with CFD and mixed-fidelity system models. In *AIAA SCITECH 2025 Forum* (p. 0371).

[3] Ruh, M. L., & Hwang, J. T. (2025). Differentiable Surface Mesh Deformation for Non-Conformal, Independently Parameterized Components with Large Shape Changes. In *AIAA AVIATION FORUM AND ASCEND 2025* (p. 3683).

Computation of intersection curves introduces many design choices and potential bottlenecks

- To define an intersection as a function, some important choices are:
  - **Function space**: piecewise linear, B-splines, CST, …
  - **Number of functions**: one for the entire intersection, upper and lower, …
  - **Number of data points** to fit the function
  - **Consistent ordering** of the data points
  - Mapping of intersection mesh nodes to the function domain (physical to parameter space)
- These choices directly affect accuracy, robustness, and differentiability
- **How can we handle intersections without recomputing intersection curves?**

Existing, well-established mesh deformation methods remain important

- Once intersection nodes are updated, the next step is to compute the displacement field:

$$\Delta \mathbf{x}^i = \mathbf{x}^i - \mathbf{x}_0^i$$

- The motion of the intersection mesh nodes can be viewed as prescribed boundary conditions.
- Established deformation methods update the remaining vertices.

Physics-based

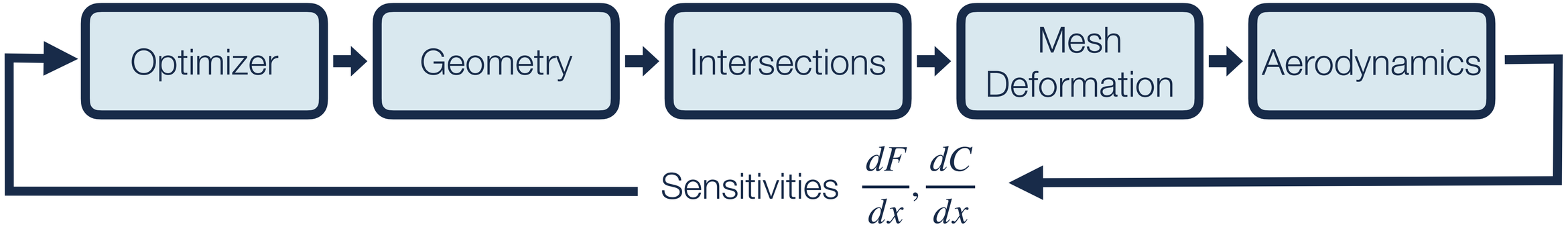
- Directly operate on mesh by solving a PDE
- Displacement from PDE solution
- Ex: Laplacian, elasticity-based, etc

Interpolation-based

- Fitting a displacement surrogate model
- Radial basis functions (RBF), Inverse distance weighting (IDW)

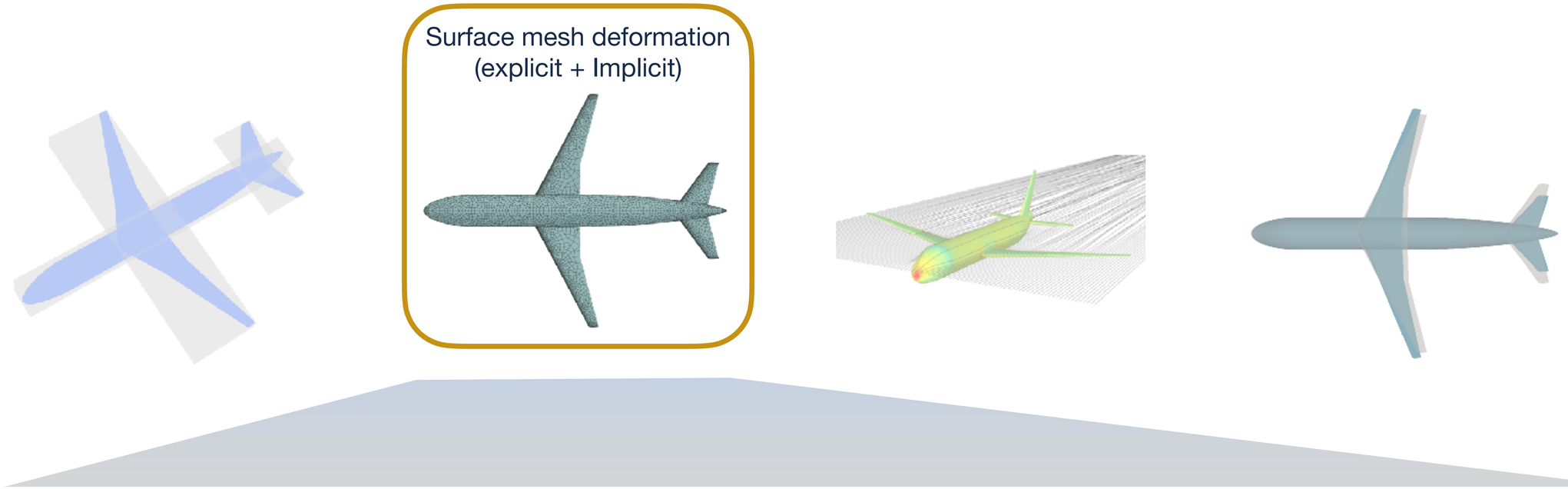
We contribute and demonstrate a methodology for differentiable surface mesh deformation

- **Differentiability**: compatible with adjoint-based workflows
- **Quality preservation**: maintains acceptable mesh quality even under extreme deformations
- **Robustness**: tolerates large translations, rotations, and scaling
- **OML conformity**: deformed mesh stays on the OML
- **Efficiency**: deformation time is manageable compared to the aerodynamic solver



A Novel, Differentiable Surface Mesh Deformation Algorithm

The proposed mesh deformation methodology has three main steps



1) Computation of discrete intersection vertices (not curves)

2) Mesh-based, quality-guided deformation of (free) mesh nodes

3) Projection of deformed mesh nodes onto CAD geometry

At setup time, we generate a conforming surface mesh and project it onto the OML

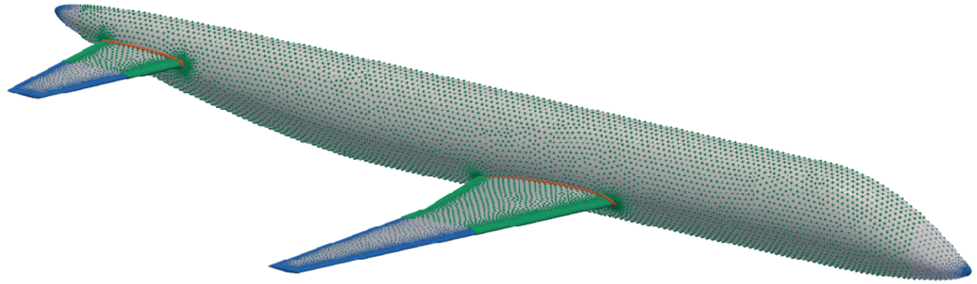
- The projection operation maps every mesh node $\mathbf{x}^j \in \mathbb{R}^3$ from physical space to parametric space (i, u, v), where i denotes the parametric surface index.

- The motion of mesh nodes can be:

  - Prescribed: the updated mesh position is known through the intersection computation ($\cdot$) or re-evaluation of parametric coordinates ($\cdot$)

  - Free: computed via a mesh-based deformation method (Graph—Laplacian) ($\cdot$)

- The number of free and prescribed (parametric) mesh nodes is a user choice.



- Prescribed: through intersection computation
- Prescribed: re-evaluated from parametric space
- Free: computed through bulk deformation

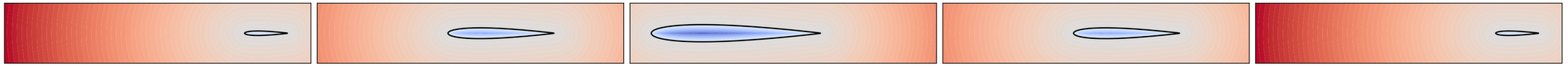
Signed distance functions can implicitly define complex geometries

- A signed distance function (SDF) $\phi(\boldsymbol{x})$ is a scalar function that computes the orthogonal distance of a point $\boldsymbol{x}$ to the boundary $\partial\Omega$ of a set Ω

$$\phi(\boldsymbol{x}) = \begin{cases} -d(\boldsymbol{x}, \partial\Omega) & \boldsymbol{x} \in \Omega \\ d(\boldsymbol{x}, \partial\Omega) & \boldsymbol{x} \notin \Omega \\ 0 & \boldsymbol{x} \in \partial\Omega \end{cases} \qquad d(\boldsymbol{x}, \partial\Omega) = \inf_{\boldsymbol{y} \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{y}\|_2$$

- A boundary surface can be represented by the zero-level set of the SDF, where the gradient $\nabla\phi(\boldsymbol{x})$ equals the surface normal vector

- For CAD geometries, we compute the SDF using closest-point projections, where the sign is determined using the dot product of the projected point with the surface normal vector



Signed-distance field of a swept wing at different spanwise sections

Using signed distance functions, a 1-D root-finding method computes intersection vertices

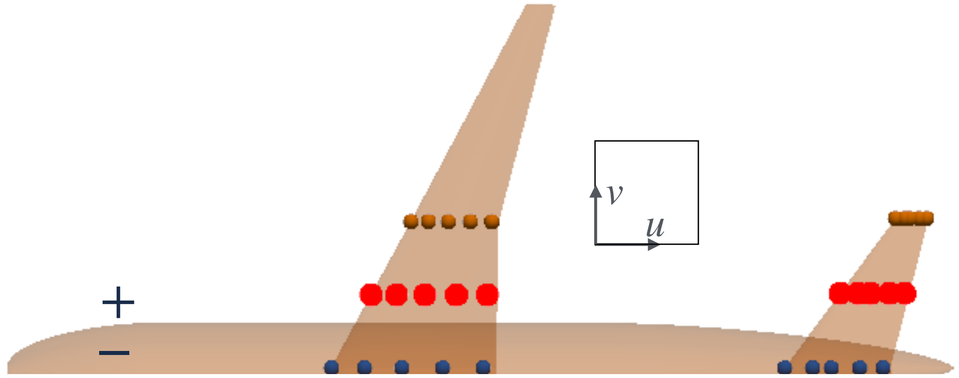
- At setup time, each intersection mesh node shared between two components is assigned to a “driving” component, from which it inherits parametric coordinates. E.g., for a wing-fuselage intersection, the mesh nodes at the intersection are assigned to the wing's parametric space.

- The parametric domain along one direction (e.g., $v = [0,1]$) contains a bracket wrt the sign of the SDF value of the other intersecting component, i.e.,

$$\phi_{\text{fuse}}\left(S_{\text{wing}}(u, v = 1)\right) > 0 \qquad \phi_{\text{fuse}}\left(S_{\text{wing}}(u, v = 0)\right) < 0$$

- A 1-D root-finding algorithm robustly finds the exact intersection nodes.

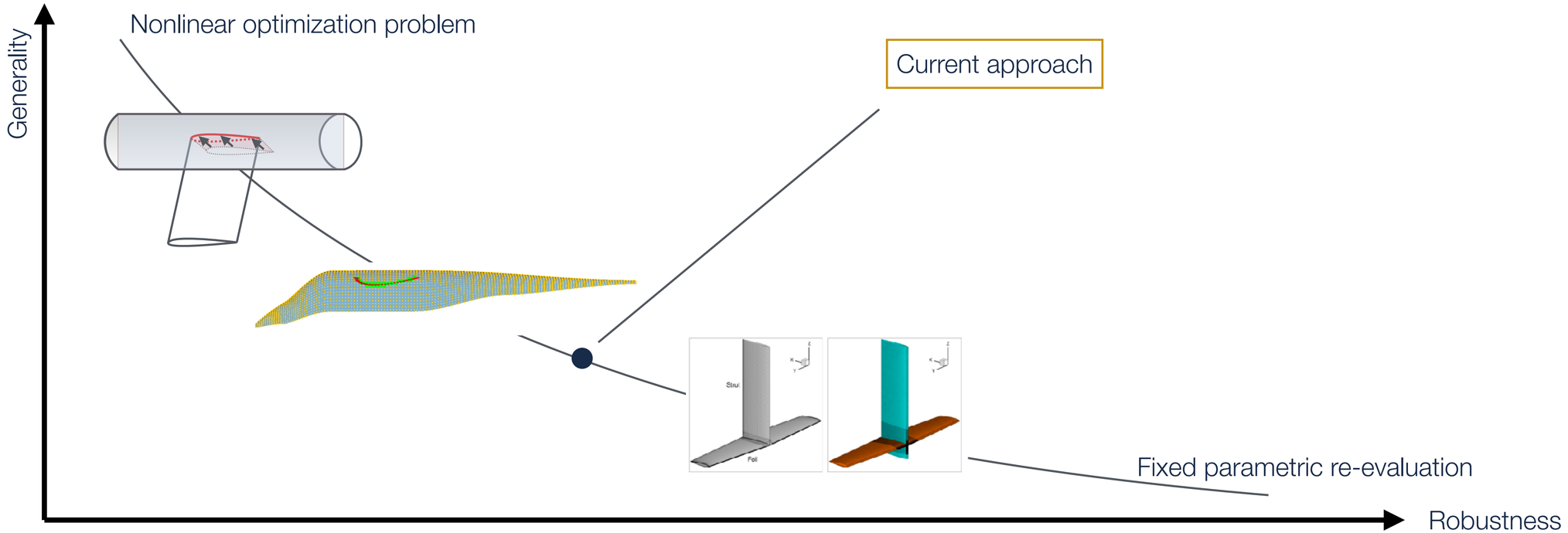
- This approach avoids computing intersection curves and naturally preserves mesh node spacing near component intersections.



Computation of intersection vertices

Our approach for computing intersections lies on a Pareto front of generality vs. robustness

The approach works well for many practical component intersections at the conceptual level. However, more general methods may be necessary, e.g., if the intersection topology changes.



Notional diagram showing the generality-robustness tradeoff of computing component intersections

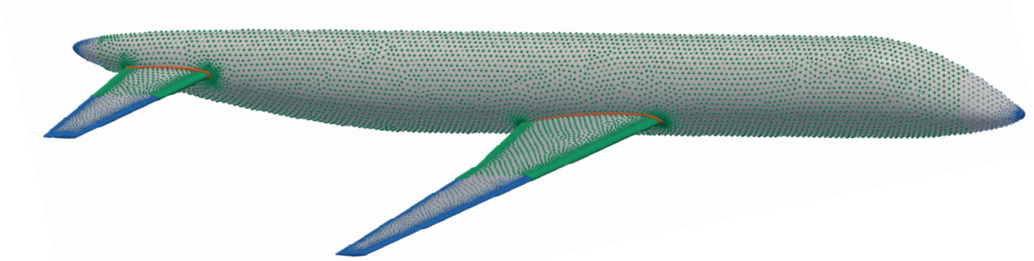
A regularized graph-Laplacian formulation deforms the bulk of the remaining vertices

- Treating the mesh as a graph $\mathcal{G}(\mathbf{X}, \mathbf{E})$, with vertices $\mathbf{X}$ and edges $\mathbf{E}$, we solve the following regularized least-squares problem (Dirichlet energy minimization)

$$\min_{\mathbf{X}_f} \frac{1}{2} \left\| \mathbf{W}^{\frac{1}{2}}\mathbf{B}\left(\mathbf{X} - \mathbf{X}_{ref}\right) \right\|_F^2 + \frac{\alpha}{2} \left\| R\underbrace{\left(\boldsymbol{X} - \boldsymbol{X}_{ref}\right)}_{} \right\|_F^2 \quad \text{subject to} \quad \mathbf{X}_p = \overline{\mathbf{X}}_p$$

edge weighting | edge-adjacency | component motion | regularization (penalizing non-convex polygons)

- For each edge, the weight w_{ij} is computed from the inverse area of its surrounding faces.
- Subtracting $\mathbf{X}_{ref}$ improves the representation of known rigid and affine component motion.



- $\overline{\mathbf{X}}_p$ Prescribed: through intersection computation
- $\overline{\mathbf{X}}_p$ Prescribed: re-evaluated from parametric space
- $\mathbf{X}_f$ Free: computed through bulk deformation

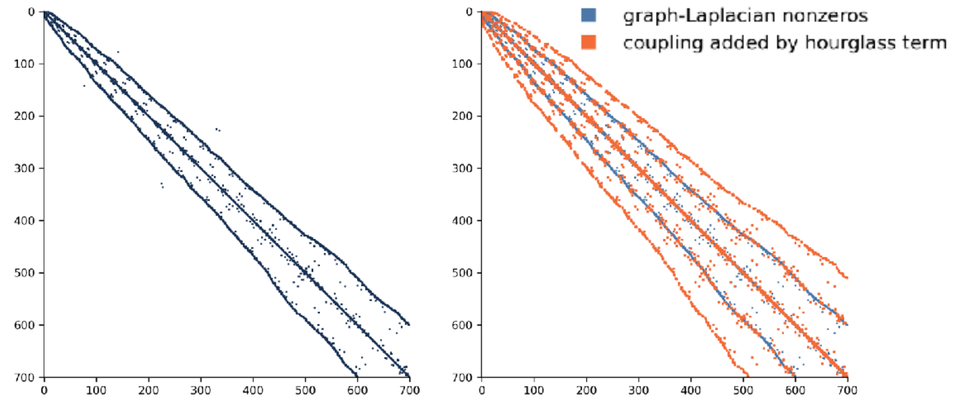
Exploiting sparsity and symmetry to solve the graph-Laplacian mesh deformation problem

- Computing the solution to the previously mentioned least-squares problem requires solving:

$$\left(\mathbf{L}_{ff} + \alpha\mathbf{K}_{ff}\right)\mathbf{Z}_f = -\left(\mathbf{L}_{fp} + \alpha\mathbf{K}_{fp}\right)\mathbf{Z}_p \qquad \mathbf{L} = \mathbf{B}^T\mathbf{W}\mathbf{B}, \quad \mathbf{K} = \mathbf{R}^T\mathbf{R}, \quad \mathbf{Z} = \mathbf{X} - \mathbf{X}_{ref}$$

- Since the matrix $\mathbf{L}_{ff} + \alpha\mathbf{K}_{ff}$ is SPD and sparse, we can leverage a sparse Cholesky factorization algorithm (CHOLMOD) for an efficient solution.

- For fixed setup time weights $\mathbf{W}$, we only need to perform one factorization, unless the mesh deformation is performed over multiple load steps.



Stiffness matrix sparsity: comparing the effect of adding regularization

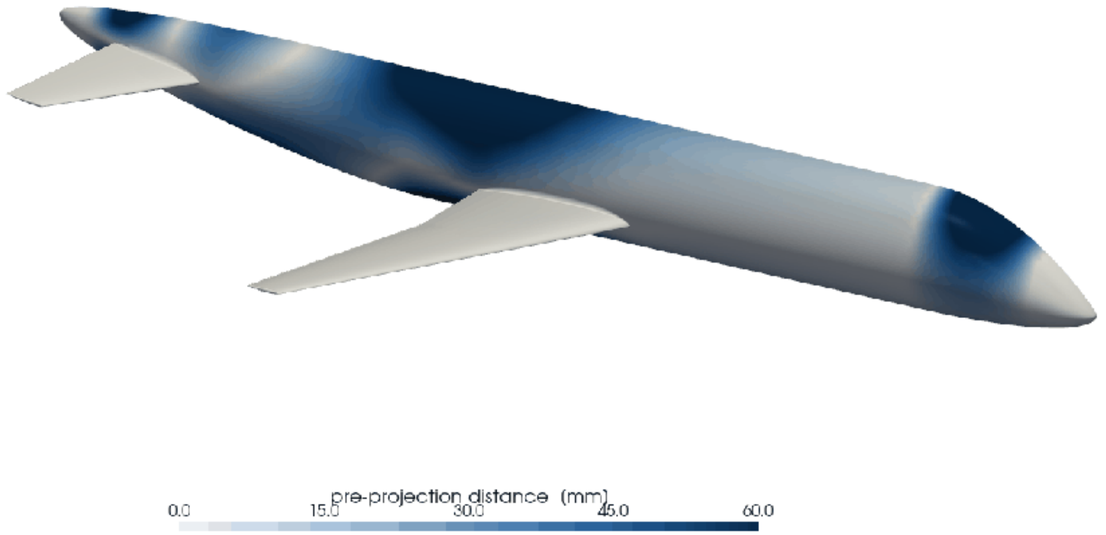
Since vertices can move off the OML, a projection step is necessary

- We project the deformed free vertices onto the CAD geometry since they may no longer lie exactly on the CAD geometry, depending on the magnitude of the geometry change.

- To maximize robustness, we seed the nonlinear solver with the result of a triangulation-based projection by triangulating the CAD geometry.

- For sensitive geometric features, we restrict the parametric surfaces that are projection-eligible.

- Standard implicit differentiation of the nonlinear residual provides the sensitivities.

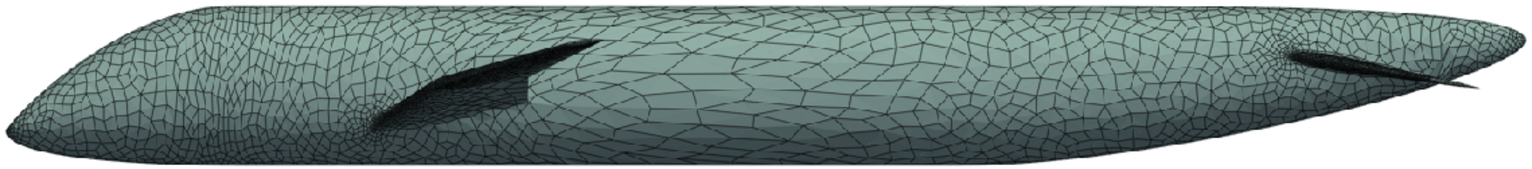


Methodology summary

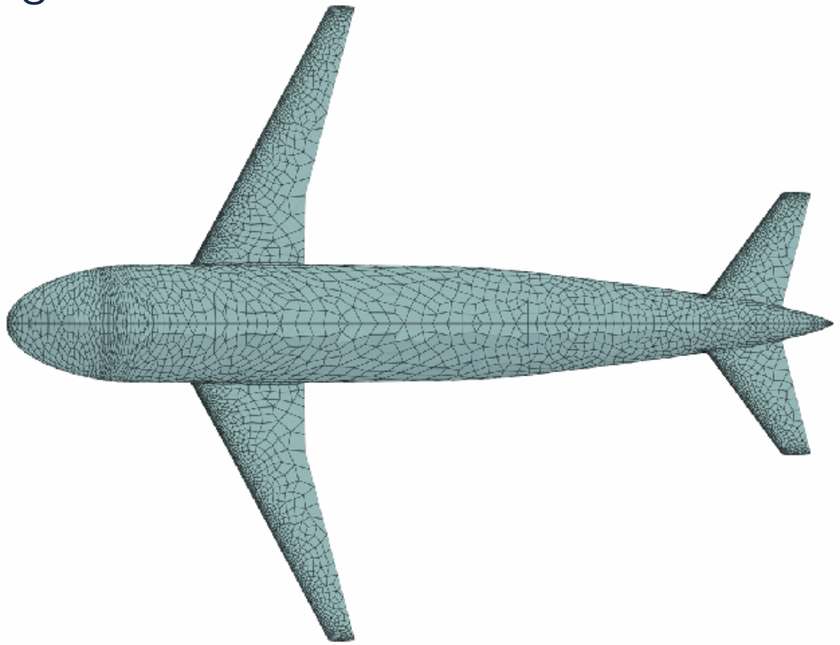
Steps	Residuals
1. **Setup time**: project mesh onto OML and assign free and prescribed vertices	None
2. **Intersection computation**: using SDFs, solve 1-D root-finding problem	Two nonlinear residuals: 1-D root-finding with nested projection-based SDF evaluations
3. **Bulk mesh deformation**: solve sparse, regularized least-squares system	One linear residual: standard least squares
4. **CAD OML projection**: project free mesh vertices onto CAD geometry	One nonlinear residual: for global continuity, requires $\mathscr{C}^1$ CAD geometry

Mesh Deformation Results for Conceptual-Level Design Changes

The methodology enables mesh deformation for practical conceptual-level geometry changes



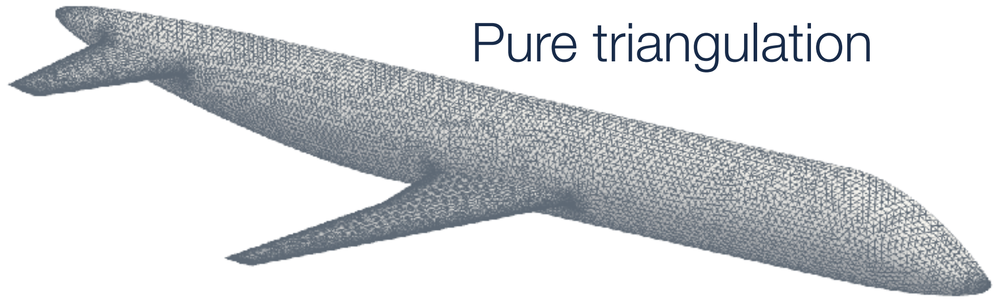
Wing rotation and translation + tail rotation



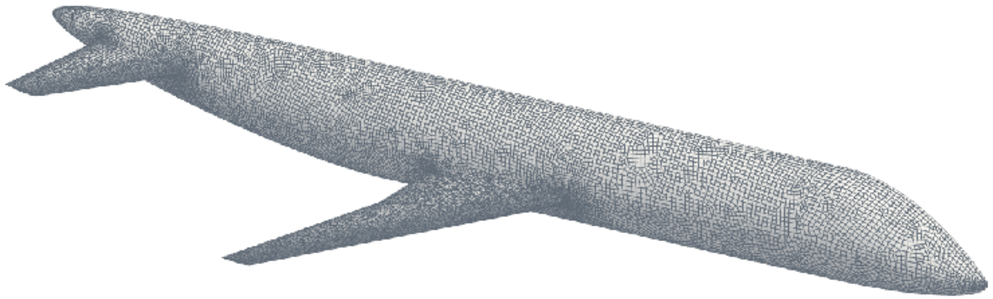
Wing translation + fuselage diameter scaling

Different surface mesh element types

Pure triangulation



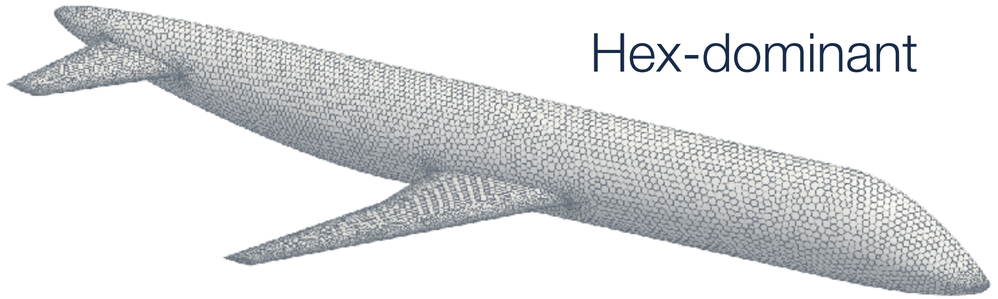
```
folded cells              0
min scaled Jacobian   0.161
max area ratio         1.59x
```



Quad-dominant

```
folded cells              0
min scaled Jacobian   0.162
max area ratio         1.71x
```

Hex-dominant



```
folded cells              0
min scaled Jacobian  -0.141
max area ratio         1.72x
```

Negative means slightly non-convex polygons (baseline mesh contains borderline elements)

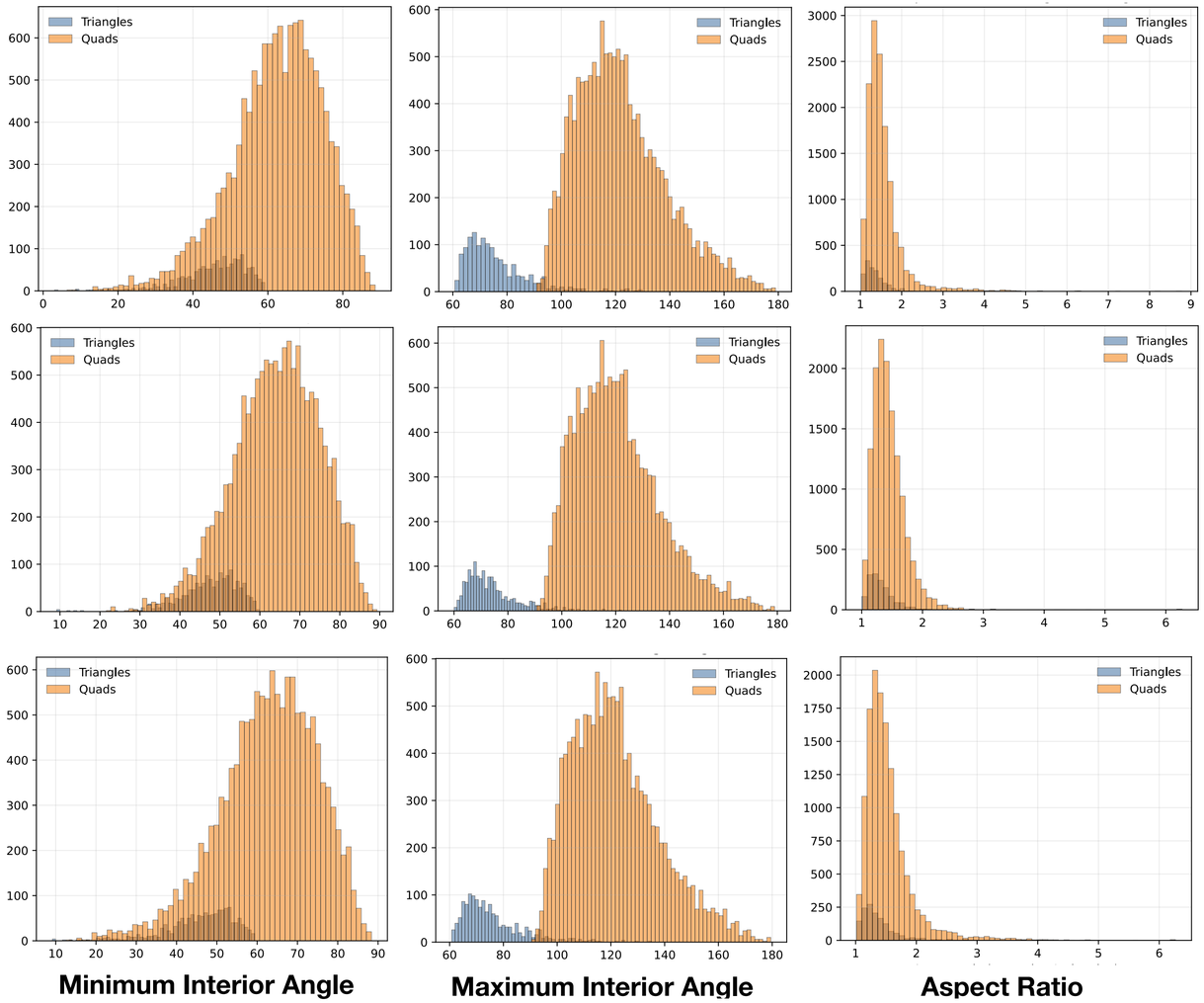
Mesh quality under extreme deformations

For a 15k element panel code mesh, less than 0.14% of elements exceed an aspect ratio of 5

Wing Translation: -1.5 MAC
Wing Rotation: -10˚
Tail Rotation: +10˚

Baseline Mesh

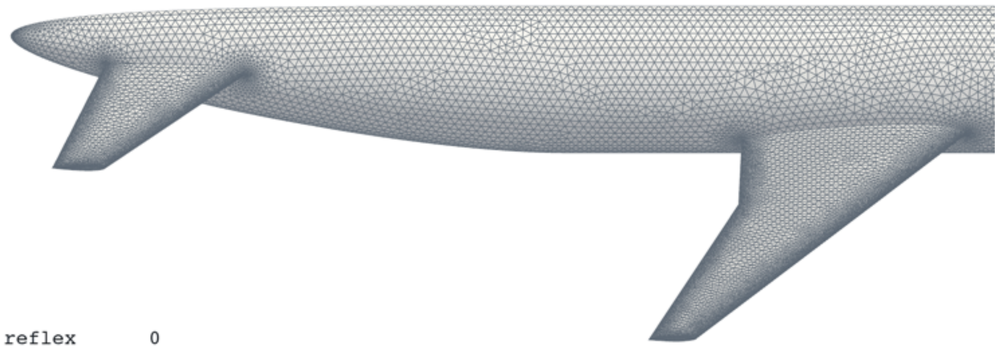
Wing Translation: +1.5 MAC
Wing Rotation: +10˚
Tail Rotation: -10˚



When does the methodology fail?

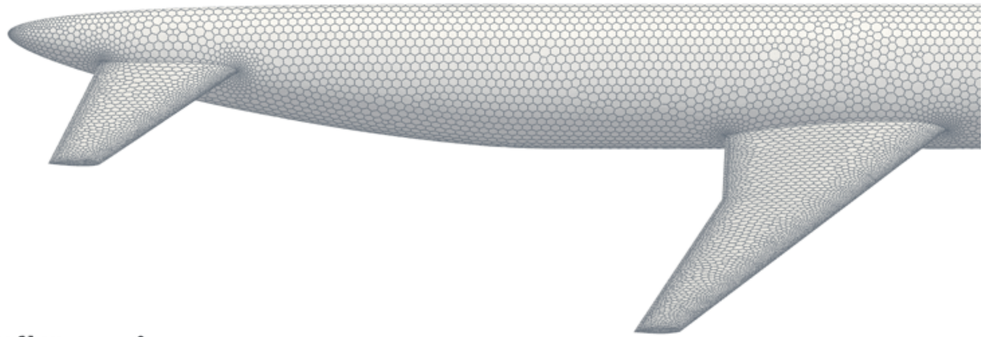
- Extreme deformations may require multiple load steps to avoid element inversion and degradation.
- Removing the quadratic penalty on the least-squares objective (for polygons with 4 or more sides) makes element inversion more likely.
- If the intersection topology changes:
  - Closed loop to an open loop → deformation bounds must be carefully chosen!
  - Merging components

Pure triangulation



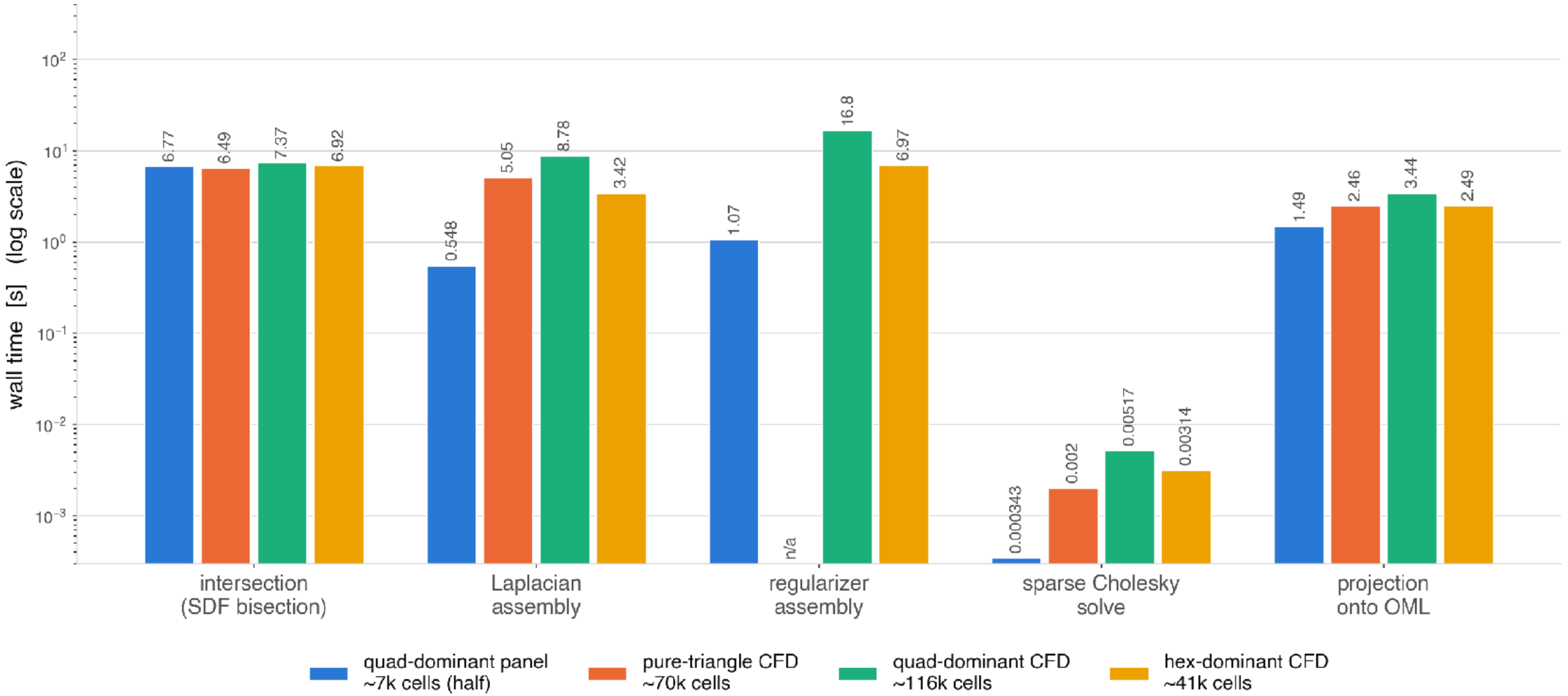
```
reflex      0
folded      0
```

Hex-dominant

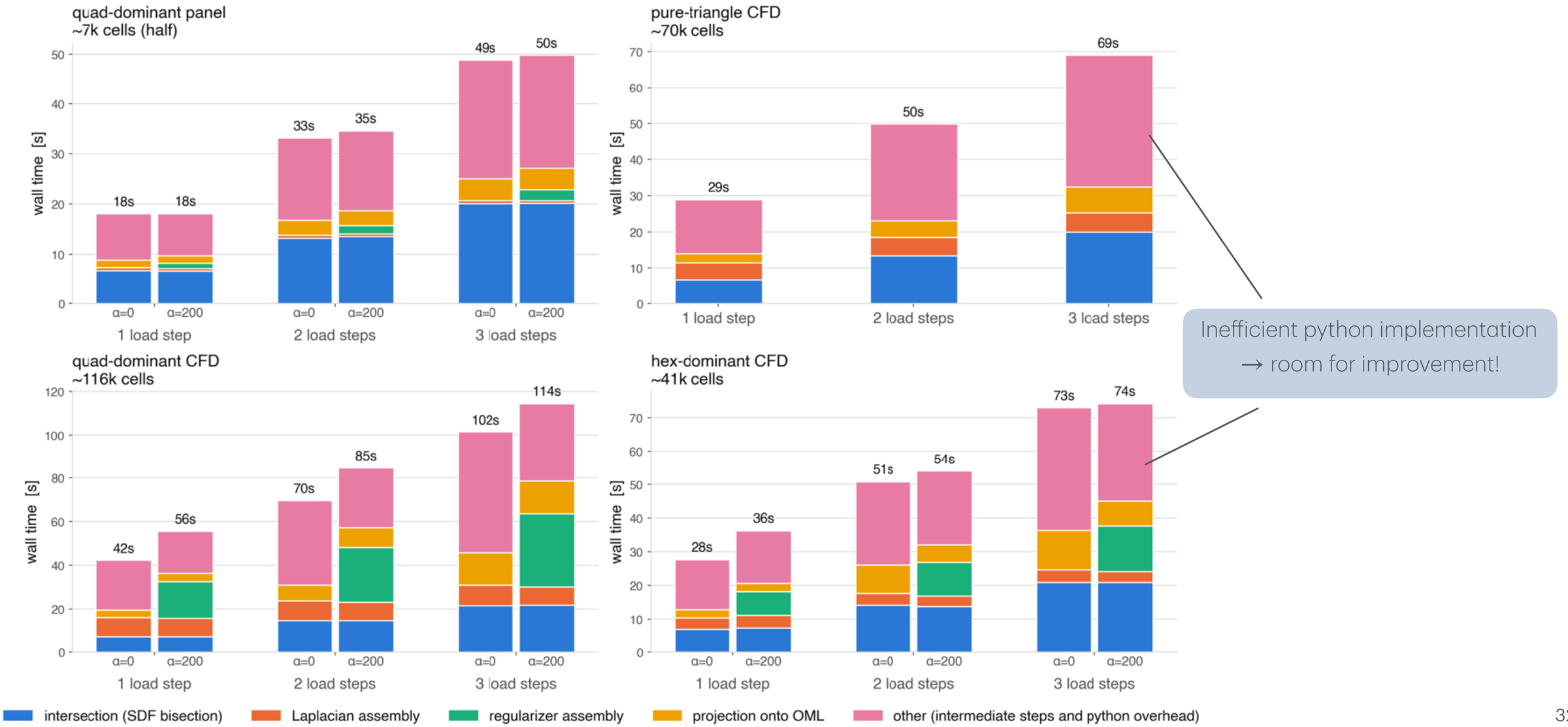


```
reflex      0
folded      0
```

What is the computation time of the methodology’s major steps?



What is the end-to-end computation time?



Inefficient python implementation
→ room for improvement!

MDO Demonstration Results

Fuel-burn minimization of an Embraer 175 regional jet using a 3-D panel code

- Predictive models
  - VortexAD: 3-D panel code with a quad-dominant mesh with ~15k elements
    - Skin friction correction (Raymer)
    - Compressibility: Prandtl–Glauert
  - Component weight build-up equations
  - Breguet range equation
- Single subsonic cruise condition: M=0.7
- Static margin computed from finite differences

	Variable	Quantity
min.	fuel burn	1
w.r.t.	Planform	4
	Wing twist	8
	Wing camber	30
	Wing translation	1
	Tail rotation	1
	Fuselage Diameter	1
subject to	Trim	2
	Wing loading	1
	Tail volume coeff.	1
	Min. Payload	1
	Static margin	1

Fuel-burn minimization planform comparison: with and without static margin

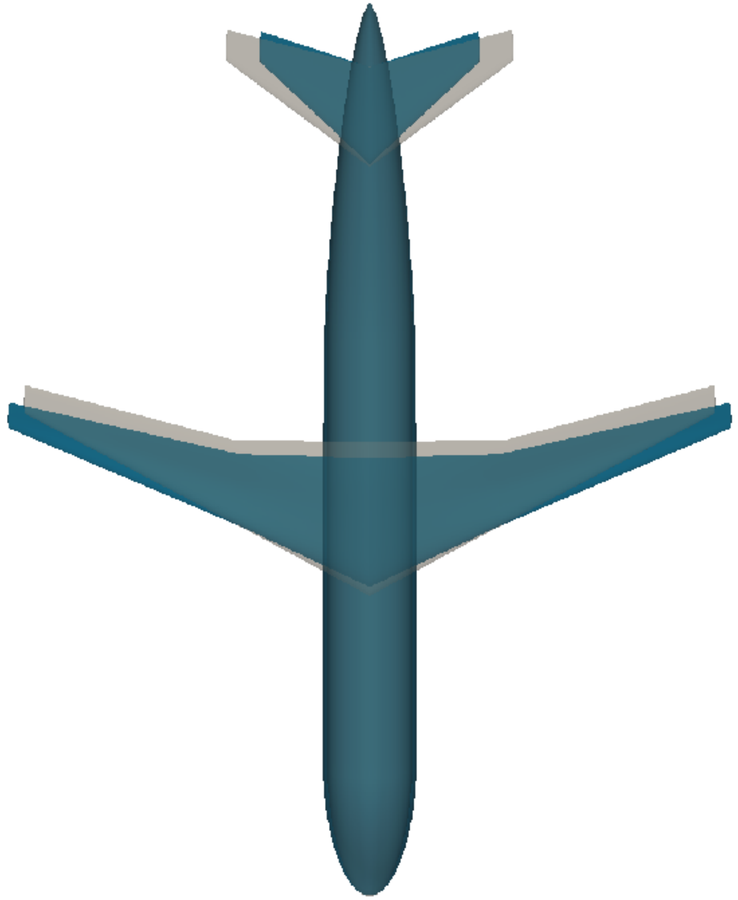
Enforcing a static margin constraint moves the wing further back.



Baseline

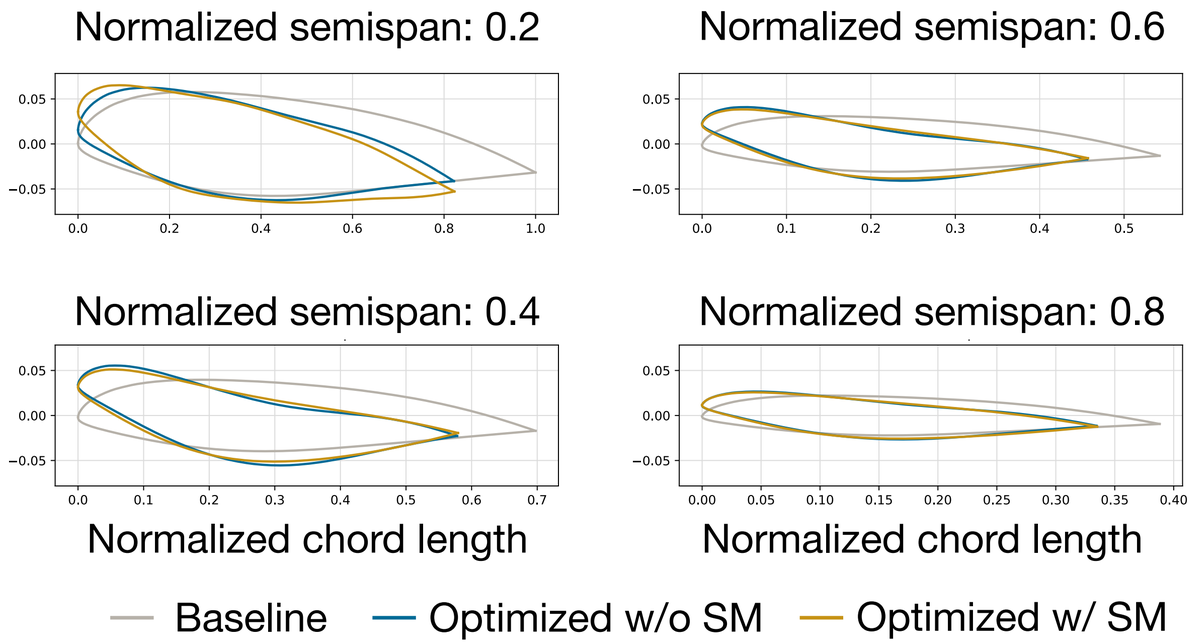
Optimized w/o SM constraint

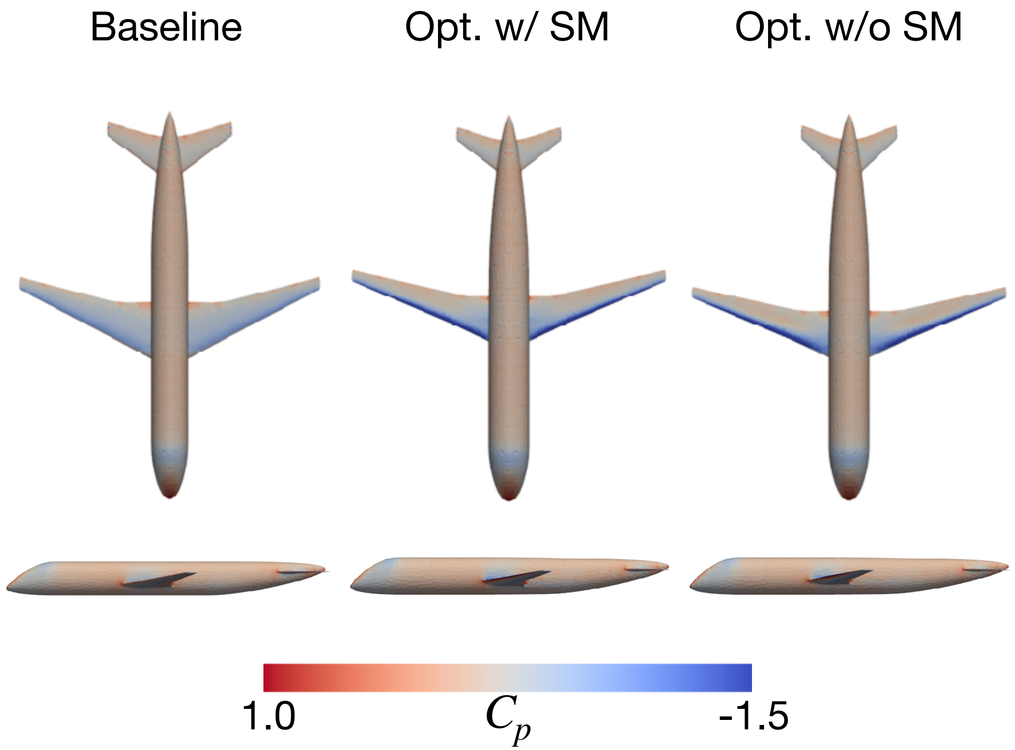
Optimized w/ SM constraint



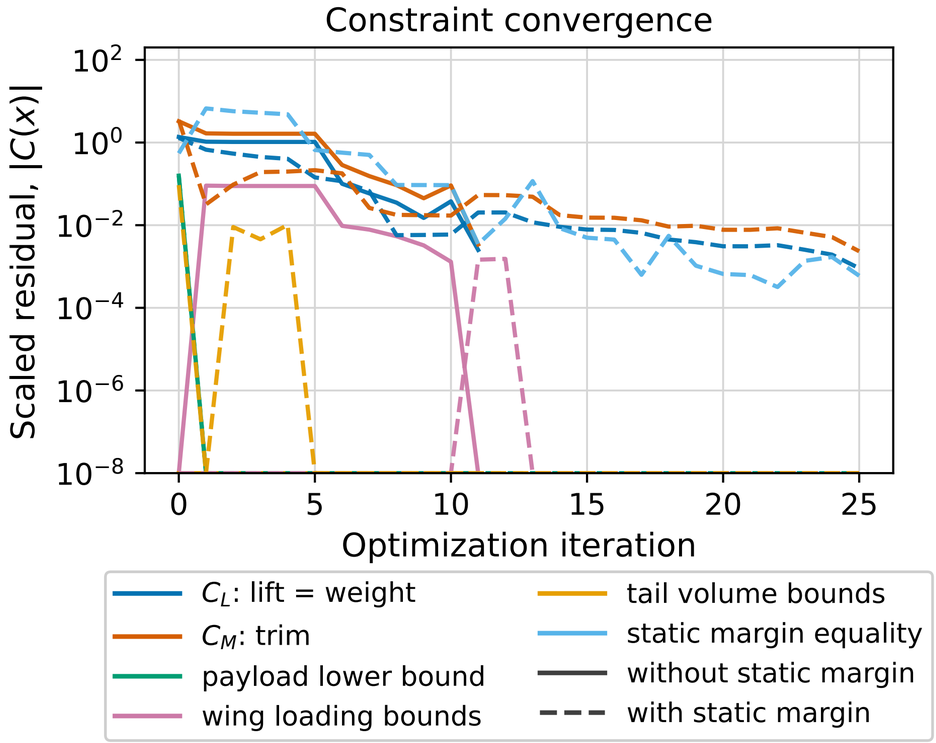
Fuel-burn minimization airfoil shape and pressure comparisons

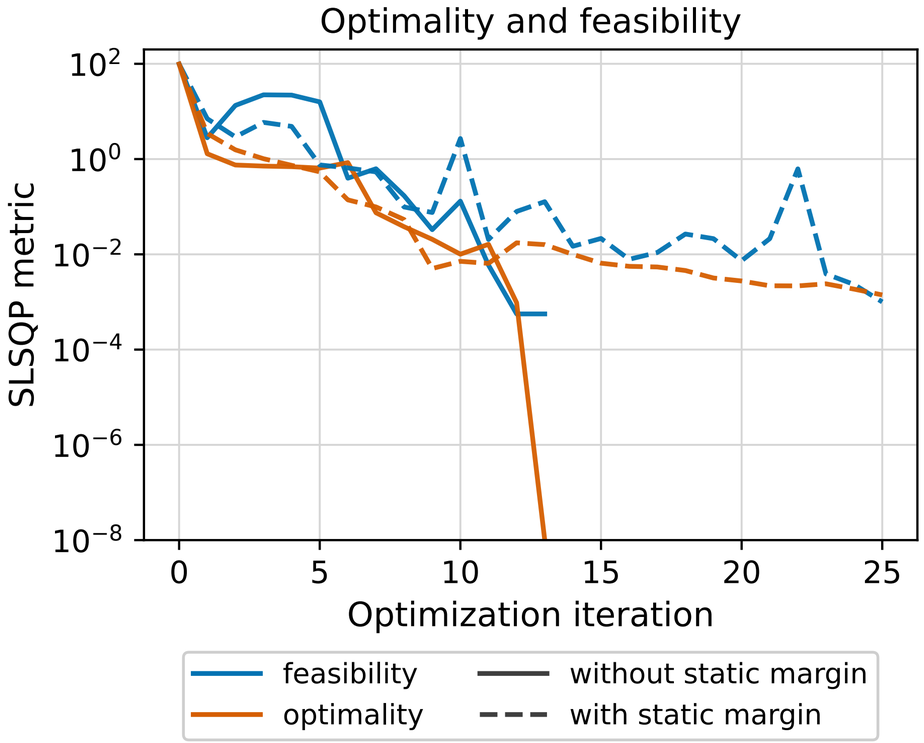
The **optimizer exploits** the **modeling assumptions** of the panel method, producing airfoil shapes with negative camber, without incurring a drag penalty.





Convergence history of the fuel-burn minimization





Lift-constrained drag minimization using an Euler CFD solver

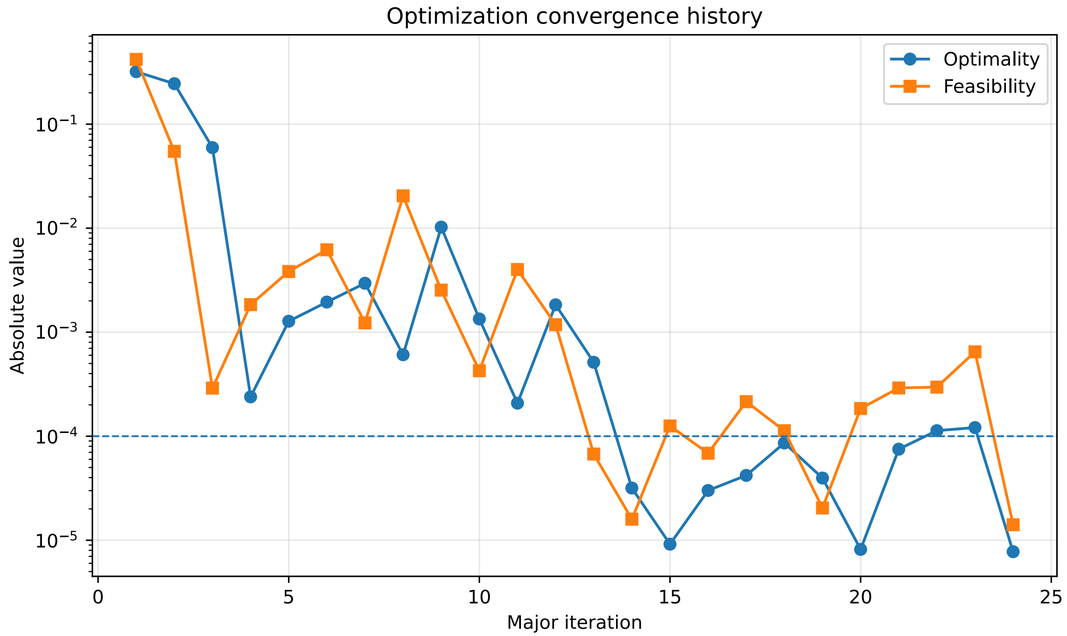
- Predictive models
  - OpenFOAM (no viscous corrections)
    - Sensitivities via DAFoam
    - **~100K surface elements** (triangles)
    - **~1.7M volume cells** (tetrahedra)
  - Component weight build-up equations
- Single subsonic cruise condition: **M=0.82**
- Elasticity-based volume mesh deformation

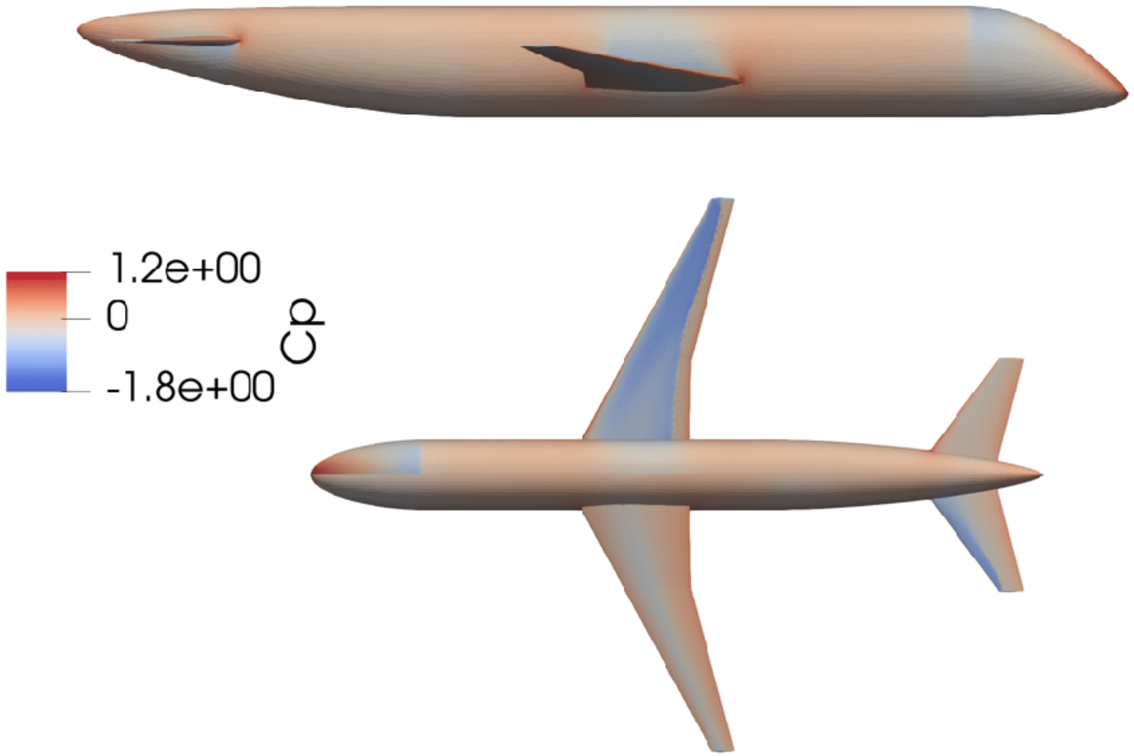
	Variable	**Quantity**
min.	Drag	1
w.r.t.	Planform	6
	Wing twist	3
	Wing camber	30
	Wing translation	1
	Wing mounting angle	1
	Tail rotation	1
subject to*	Force balance	1
	Moment balance	1

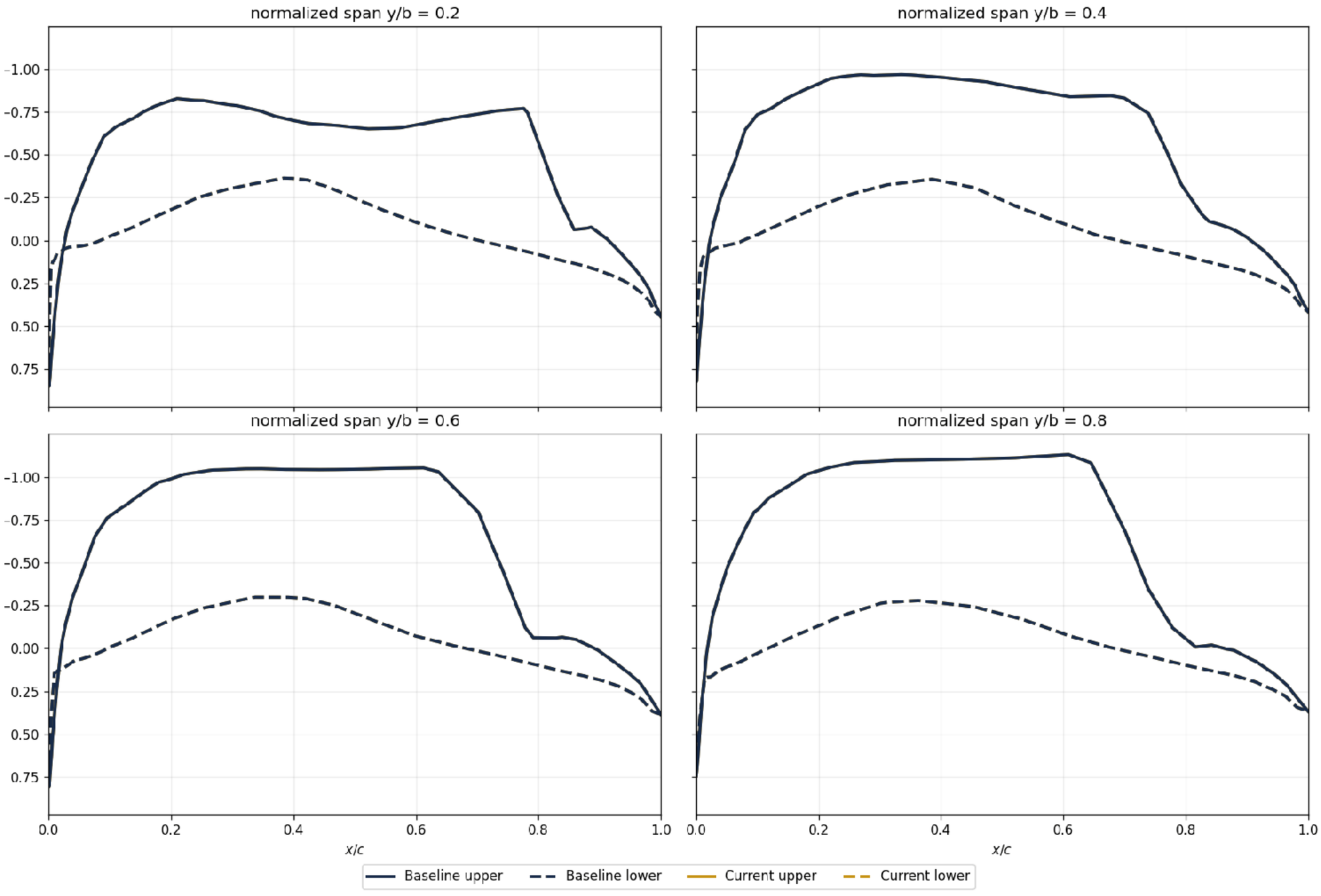
*For this demonstration, additional constraints such as wing loading or static margin were omitted to reduce computation time (i.e., to limit the number of expensive DAFoam's adjoint calls)

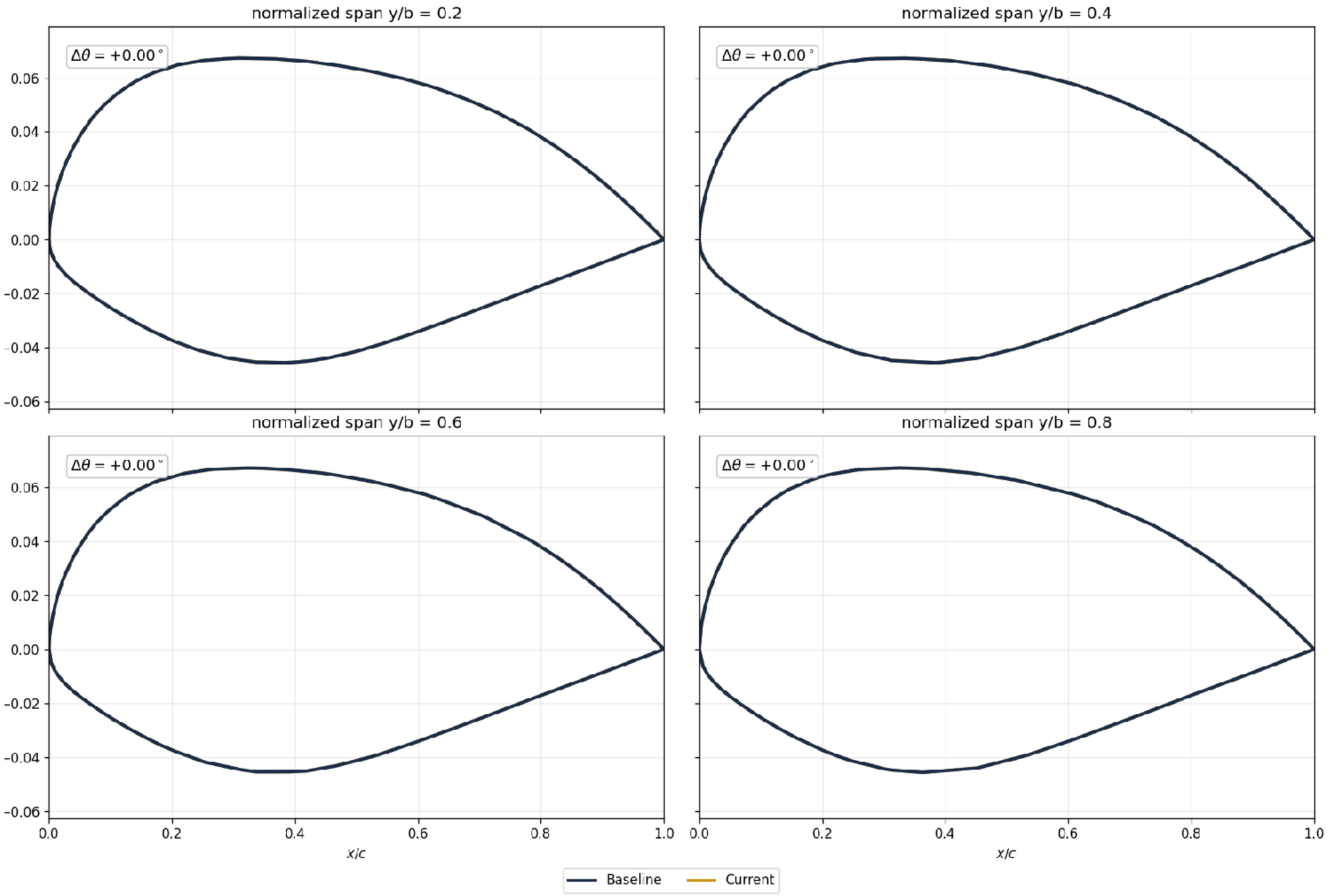
Using high-performance computing, the optimizer converges within 24 major iterations

- HPC setup: 4 nodes with 28 cores each (Intel Xeon 6724P)
- Total wall time: about 8 hours






Baseline | $C_{D,\mathrm{ref}}$ = 0.02151872 | Lift bal. = +2.902e-02 | Moment bal. = -1.186e-01
normalized span y/b = 0.2
normalized span y/b = 0.4
normalized span y/b = 0.6
normalized span y/b = 0.8
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Baseline upper
Baseline lower
Current upper
Current lower


Baseline | $C_{D,\mathrm{ref}}$ = 0.02151872 | Lift bal. = +2.902e-02 | Moment bal. = -1.186e-01
normalized span y/b = 0.2
normalized span y/b = 0.4
normalized span y/b = 0.6
normalized span y/b = 0.8
Δθ = +0.00°
0.06
0.04
0.02
0.00
-0.02
-0.04
-0.06
0.0
0.2
0.4
0.6
0.8
1.0
x/c
Baseline
Current

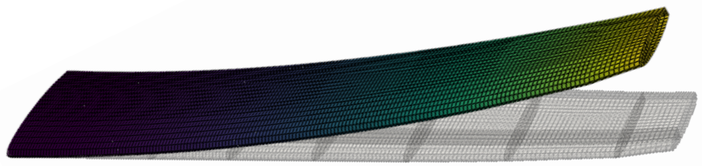
Conclusion and Future Outlook

Differentiable surface mesh deformation for 3-D aerodynamic analysis in conceptual design

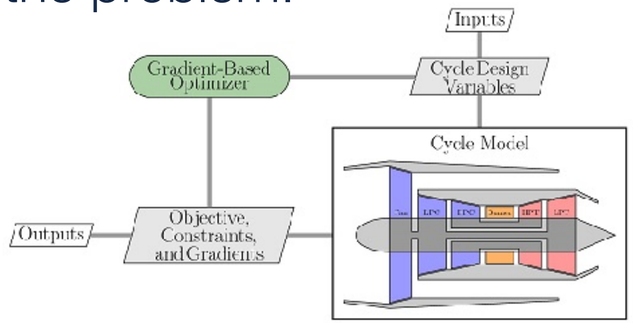
- Recent advances in modeling and simulation have enabled **increasingly complex large-scale** aircraft **design studies** with many design conditions and coupled disciplines.
- In conceptual design, however, many full-configuration **3-D aerodynamic solvers require conforming surface meshes**, creating a geometric bottleneck under large deformations.
- To address this challenge, we developed a **differentiable surface mesh deformation** methodology that computes **intersection** vertices using signed distance functions and **one-dimensional root finding**, deforms the bulk mesh using a regularized **graph-Laplacian** approach, and projects the mesh back onto the CAD geometry.
- Demonstrations using a **3-D panel code** and an **Euler CFD** solver achieved **robust optimization convergence**, requiring only a few hours on a local computer for the panel code and about eight hours on HPC for the Euler solver.

Future work will focus on integrating RANS and other discipline physics-based models

- Turbulence modeling is essential in capturing important flow physics and in improving the interpretability of the optimization results.
- Robust and efficient boundary layer cell deformation for large geometric changes may require special treatment → potential for novel research.
- Instead of empirical component weight build-up equations, a physics-based structural model (beams or shells) will capture aero-structural effects more accurately.
- An engine model such as pyCycle will add further realism to the problem.



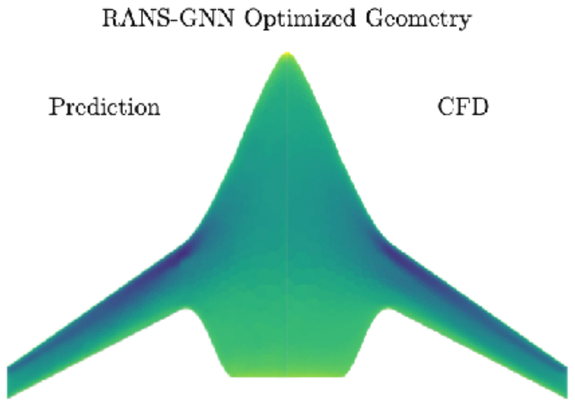
Displacement predicted by a structural shell model



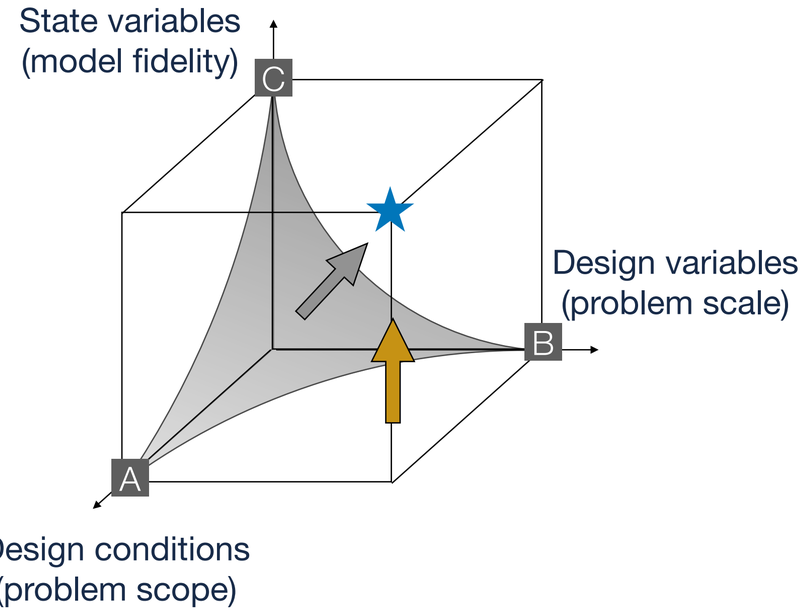
Hendricks, E. S., & Gray, J. S. (2019). pyCycle: A tool for efficient optimization of gas turbine engine cycles. Aerospace, 6(8), 87.

Ultimately, the goal is to integrate the methodology with system-level MDO studies

- The MDO results enabled by the surface mesh deformation algorithm suggest that higher-fidelity RANS-based analysis may be within reach for aircraft conceptual design, using adjoint-based MDO methods.
- Paired with recent work in multi-fidelity aerodynamic modeling, e.g., using graph neural networks (GNNs), the proposed surface mesh deformation algorithm could be a useful building block for significantly increasing the fidelity level at the conceptual design stage.



Sperry, Mark, and John T. Hwang. "Multifidelity Surrogate Modeling for 3D Aerodynamic Flow Field Prediction Using Graph Neural Networks." AIAA AVIATION 2026 Forum. 2026.



Thank you for listening

Contact: mruh@ucsd.edu