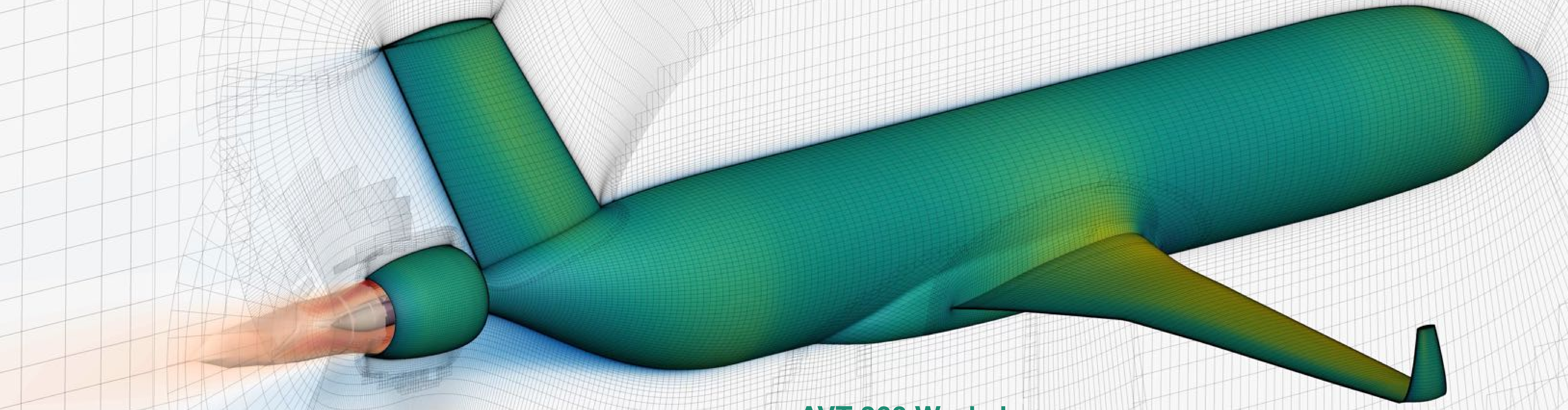
CFD-based Aircraft Design Optimization

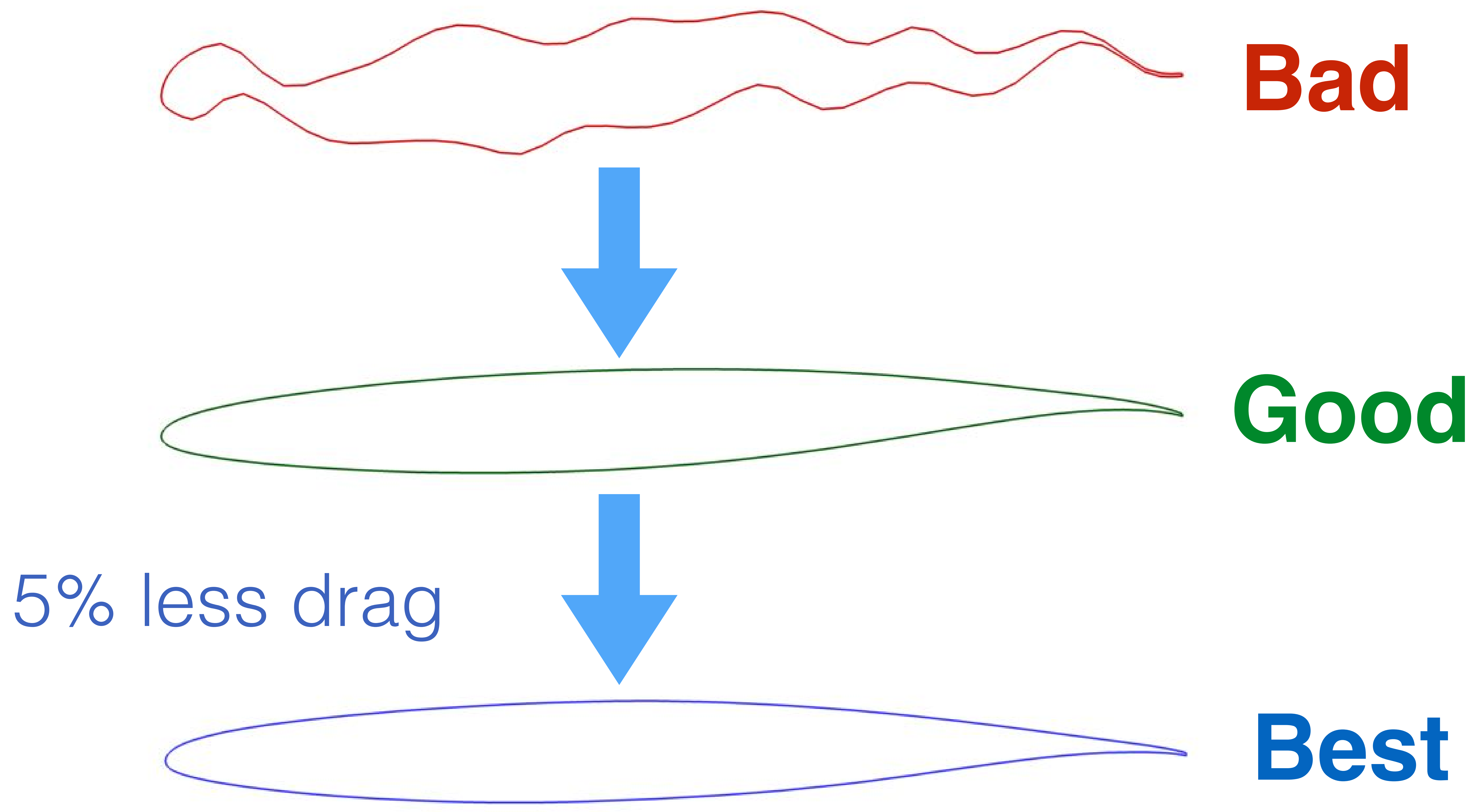


AEROSPACE ENGINEERING
UNIVERSITY of MICHIGAN
http://mdolab.engin.umich.edu

AVT-366 Workshop
May 2022
Remote Presentation

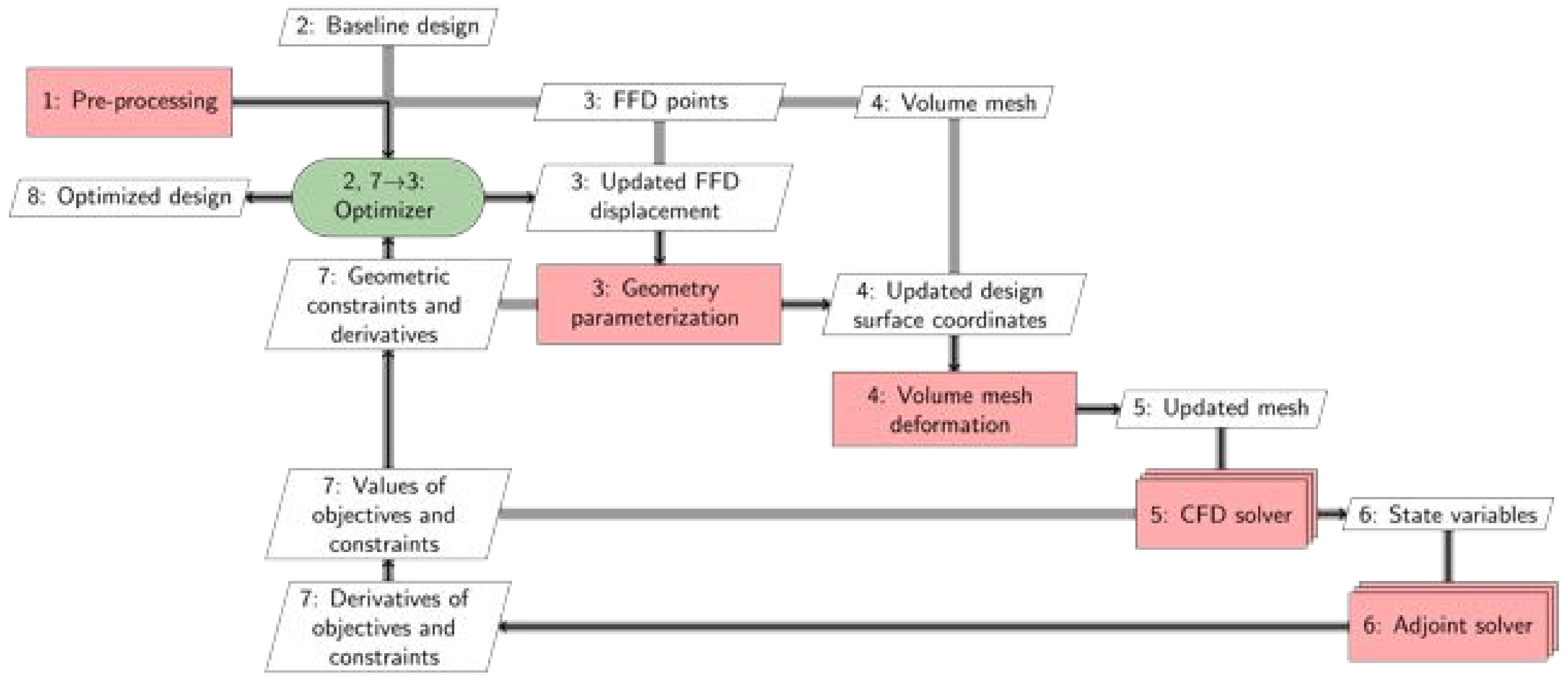
Joaquim R. R. A. Martins
with contributions from Timothy Brooks, Justin Gray, Ping He, John Hwang, John Jasa, Gaetan Kenway, Graeme Kennedy, Zhoujie Lyu, Charles Mader, Ney Secco, and Anil Yildirim

How can we find the *best* shapes?



Research in the Multidisciplinary Design Optimization Laboratory has two complementary thrusts

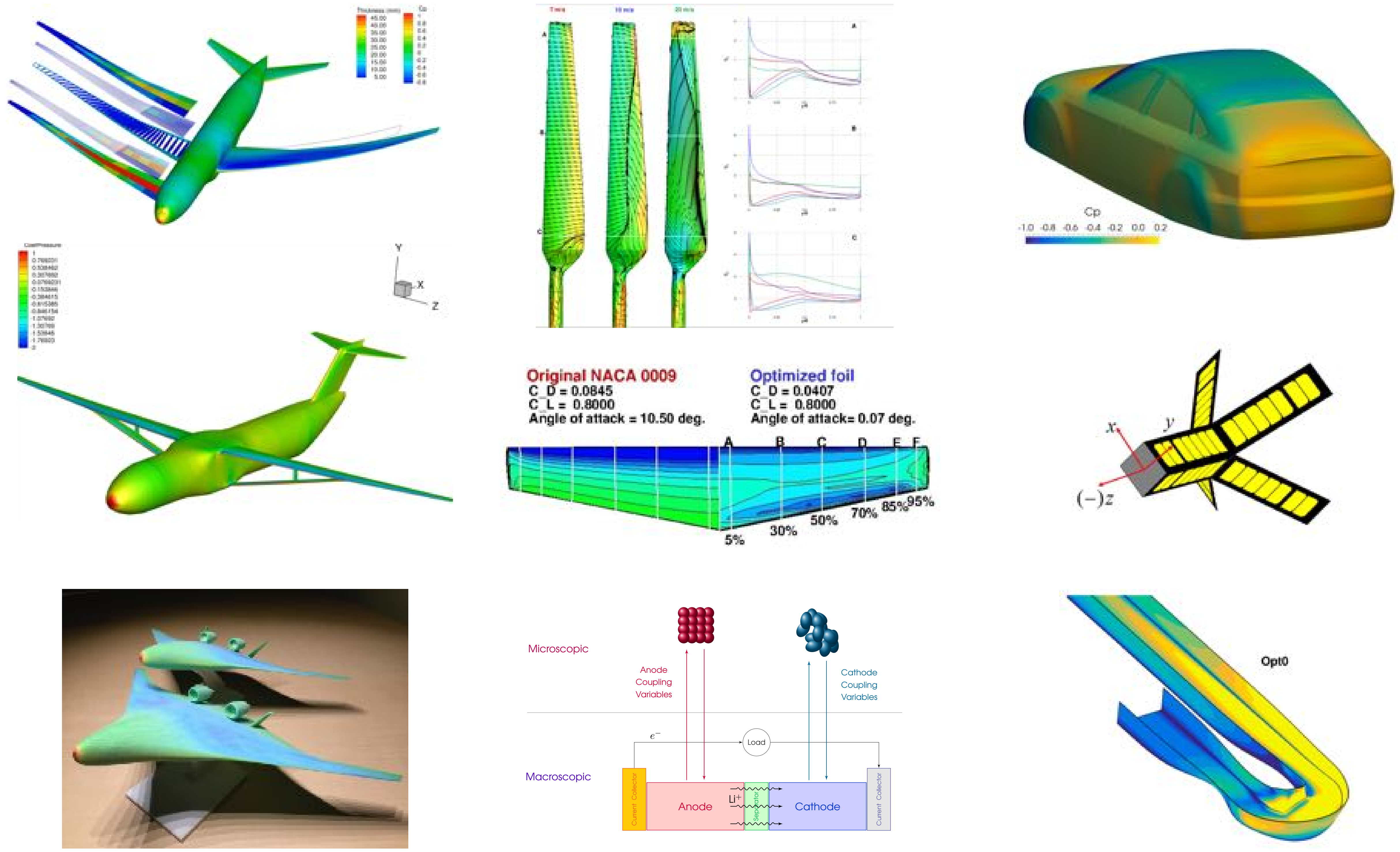
MDO algorithms



$$\frac{\partial R}{\partial u}\frac{du}{dr} = \mathcal{I} = \left[\frac{\partial R}{\partial u}\right]^T \left[\frac{du}{dr}\right]^T$$

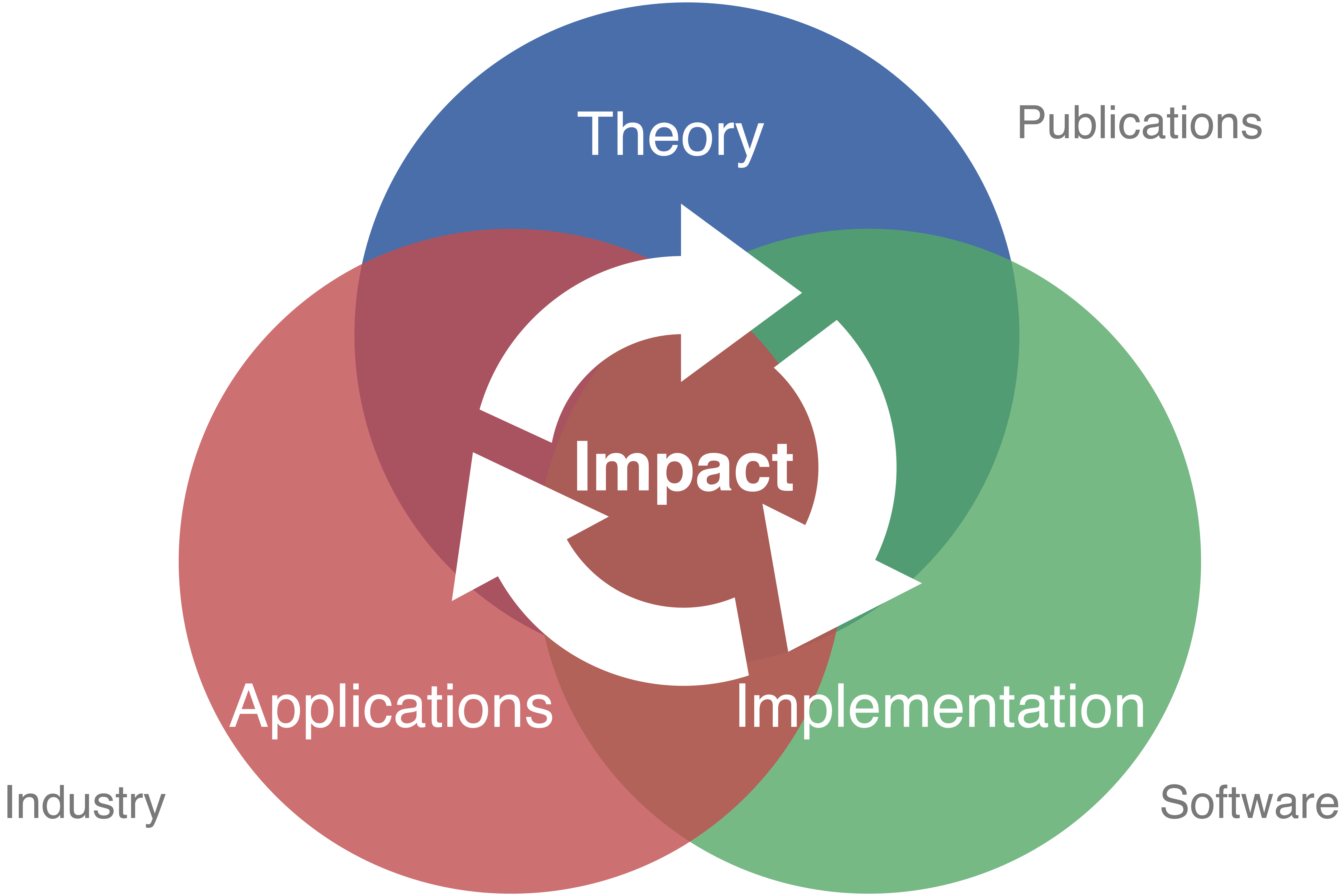
openMDAO

Applications of MDO



http://mdolab.engin.umich.edu

Theoretical developments need to be implemented and applied in industry for impact and to inform research needs

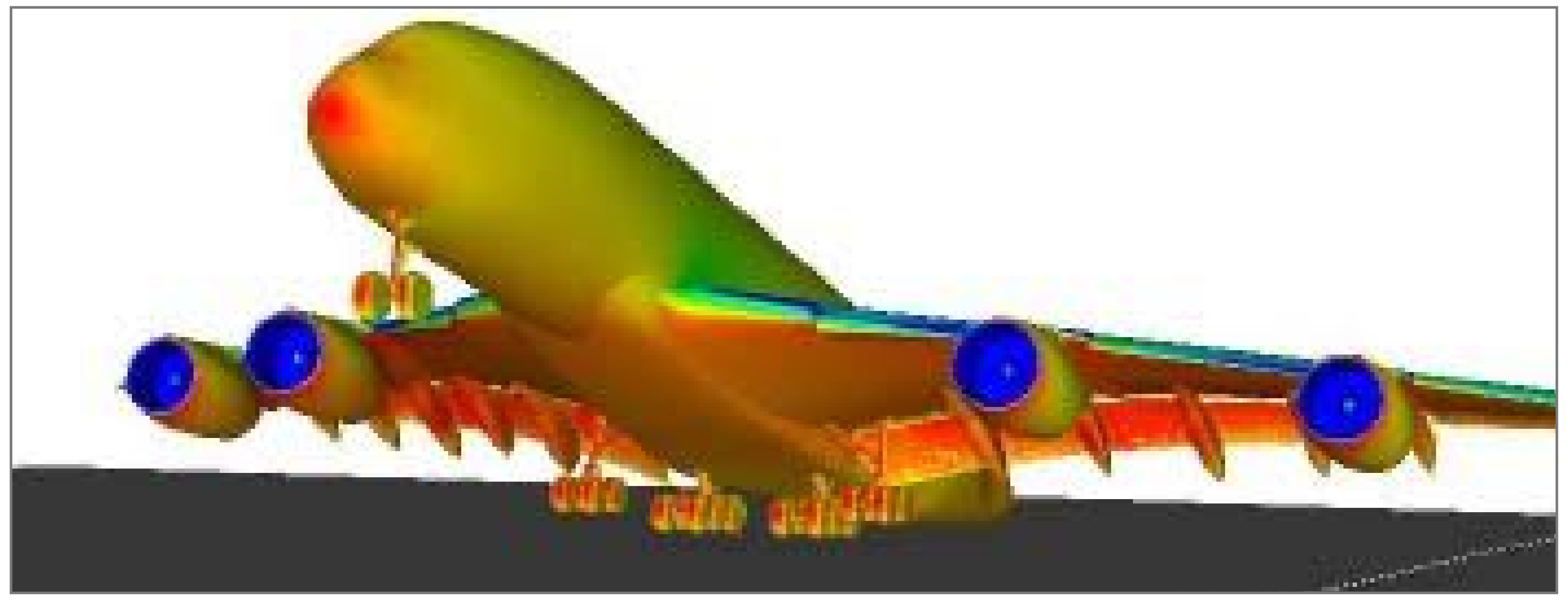


Numerical methods have been playing an increasing role in engineering simulations

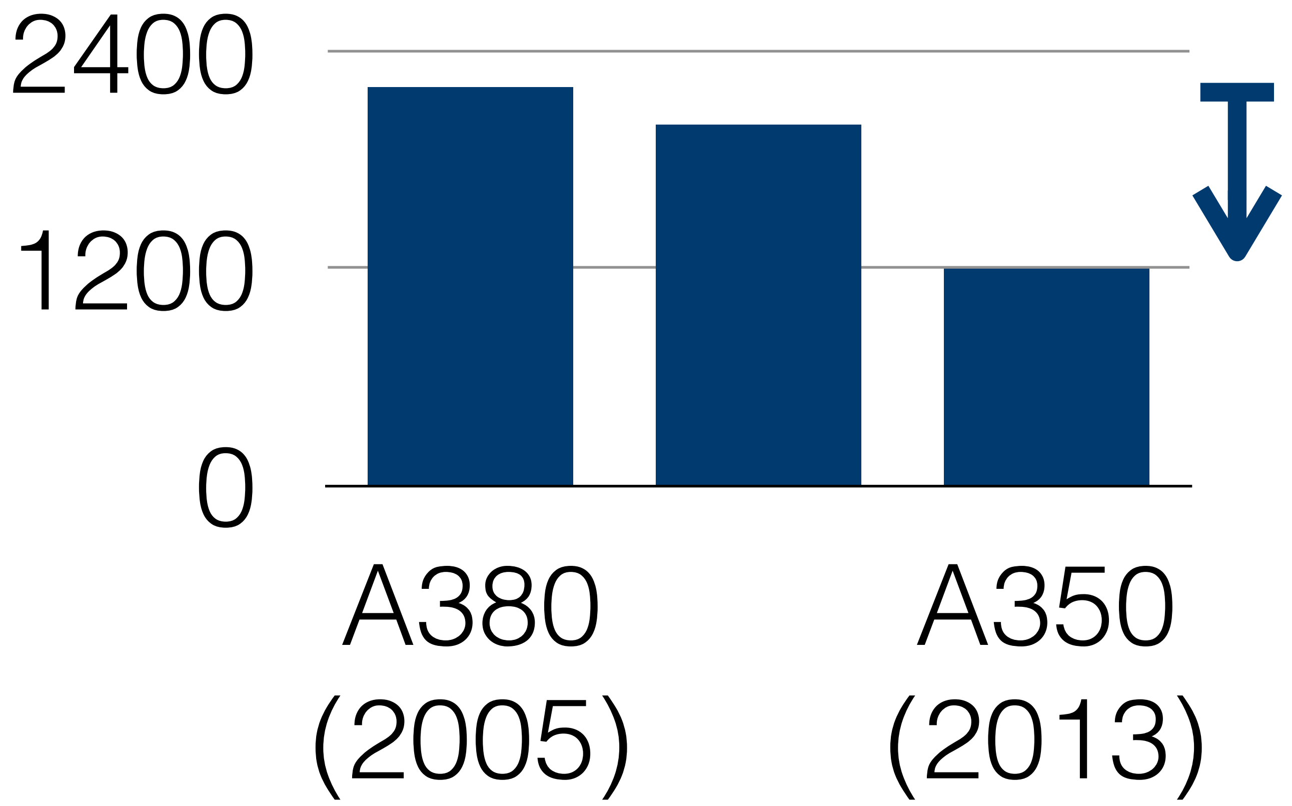
Experiments

▼

Numerical simulations



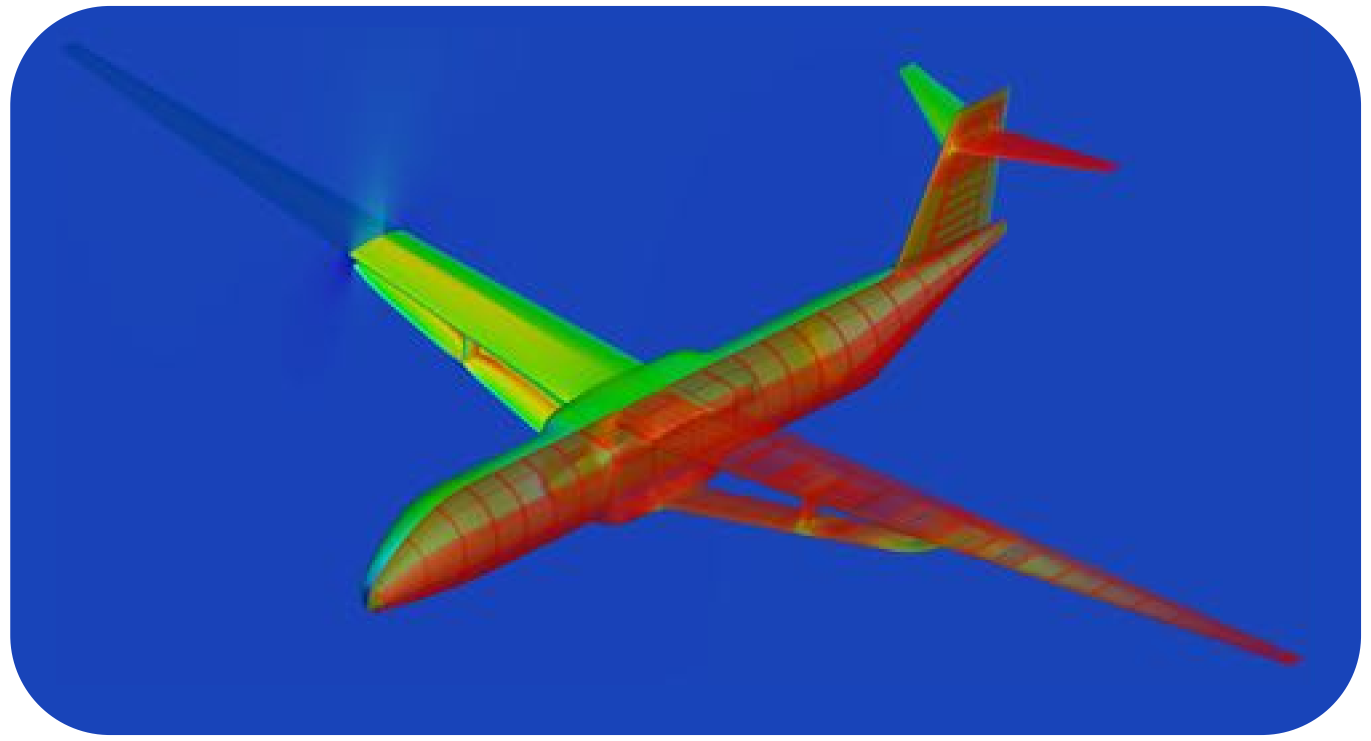
[Source: Airbus A380 - RAe Hamburg & VDI January 2008]



40% fewer wind tunnel days

Numerical optimization provides a way to fully automate the design process

Wing span
Airfoil shapes
Structural sizing



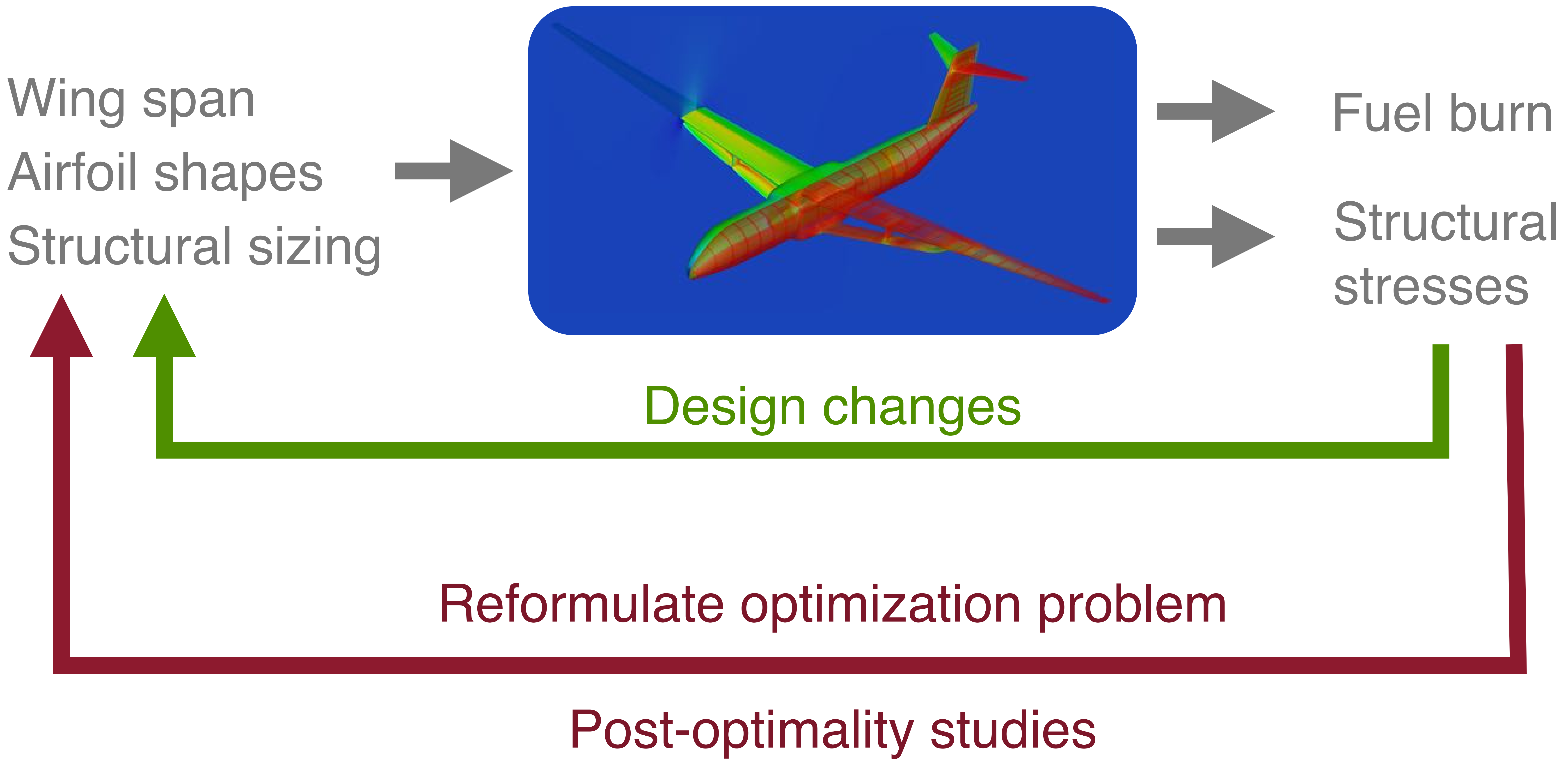
Fuel burn
Structural stresses

Design changes

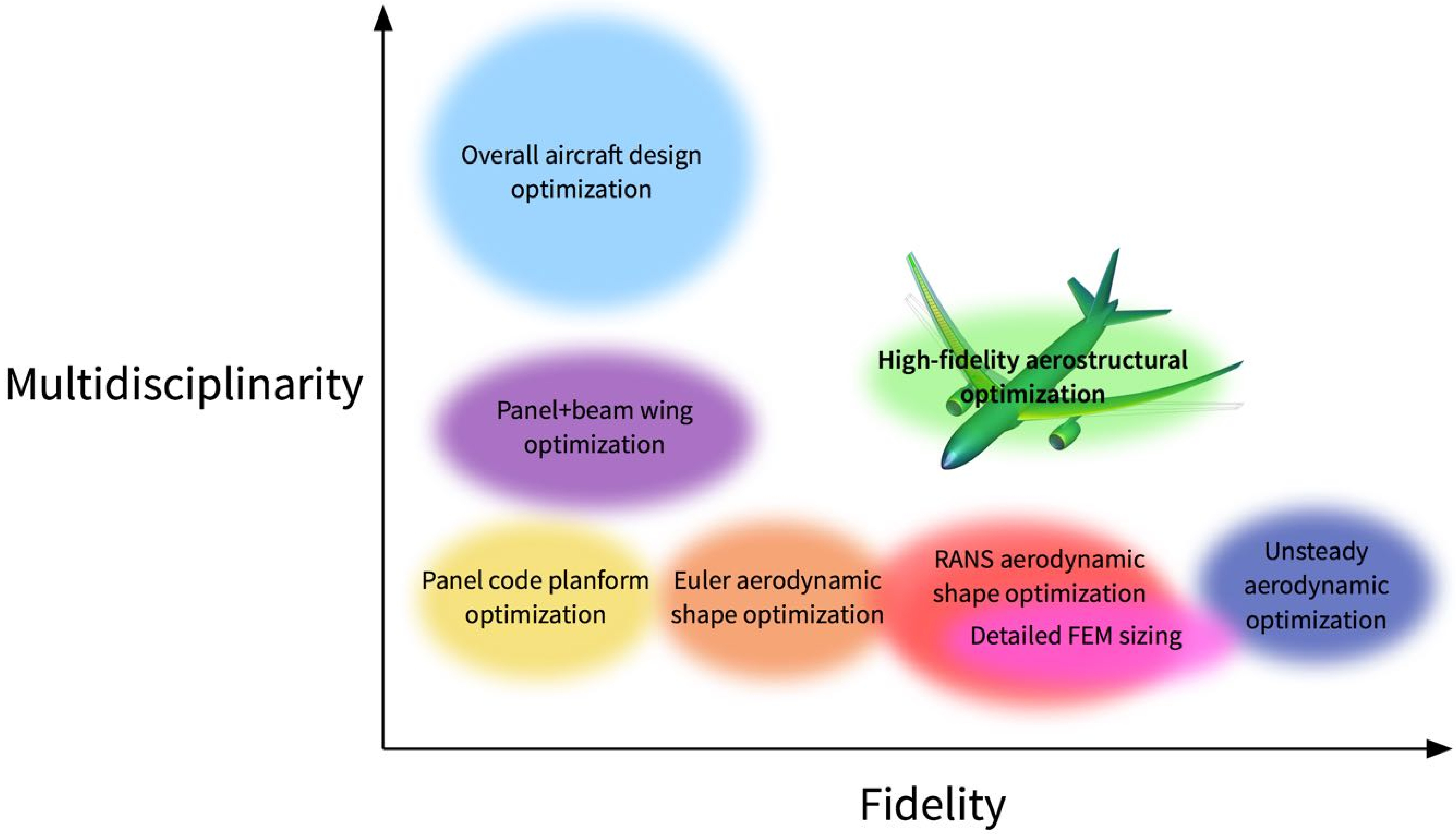
Design optimization problem:

minimize	$f(x)$	objective
with respect to	x	design variables
subject to	$c(x) \leq 0$	constraints

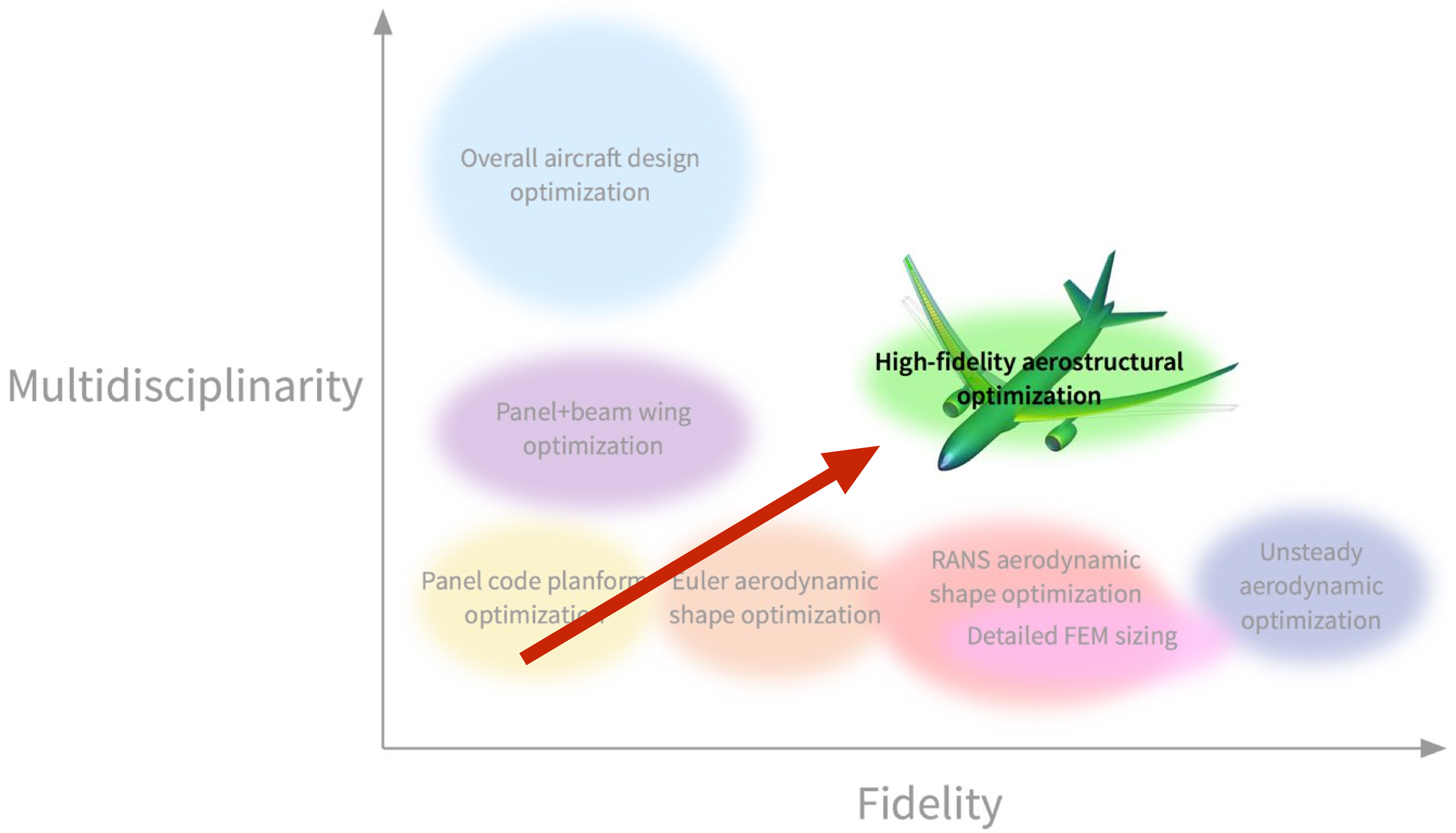
In practice, there is another outer loop where the designer reformulates the optimization problem



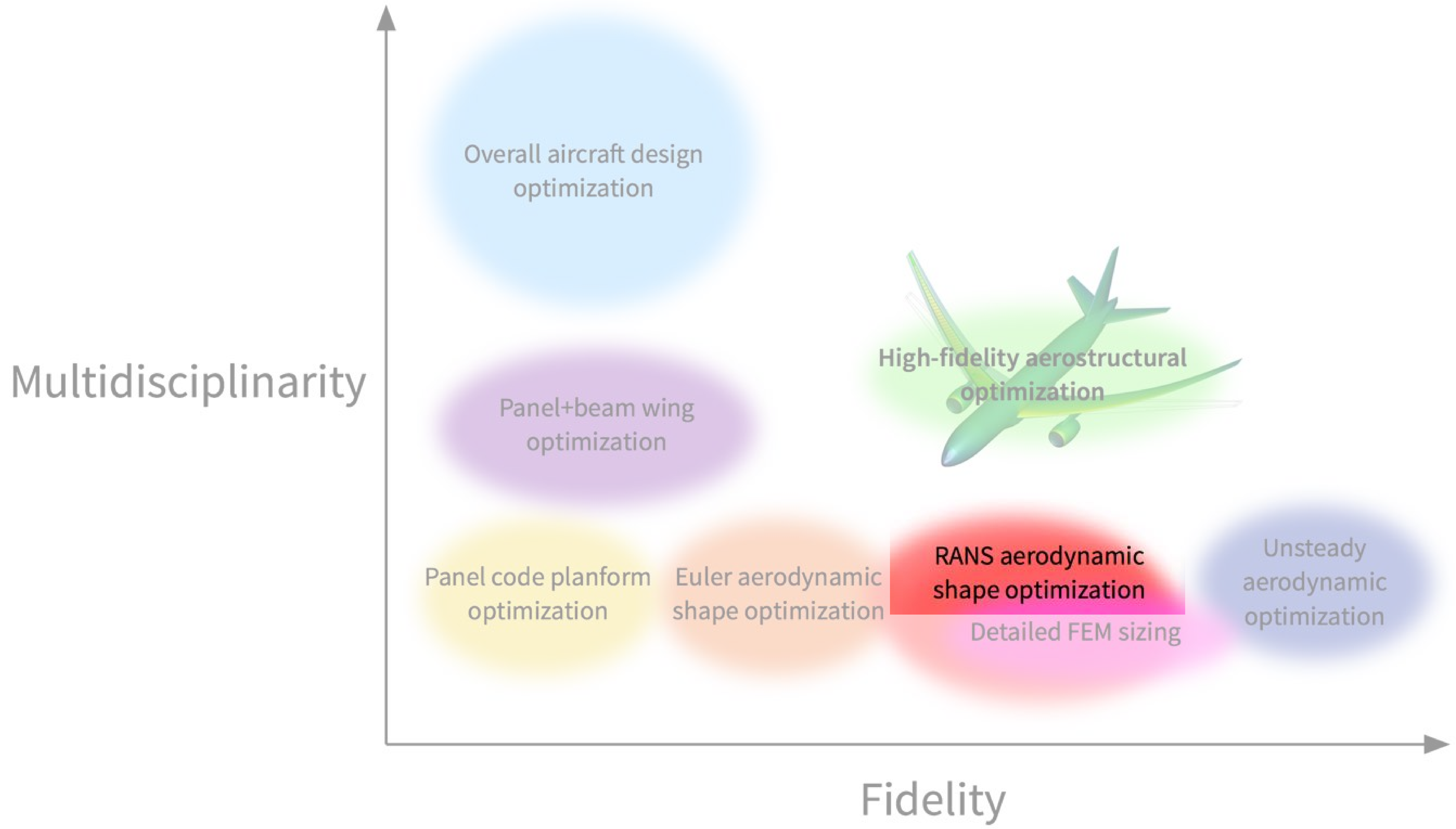
State of the art in aircraft MDO is many disciplines with low fidelity, or one or two with high fidelity



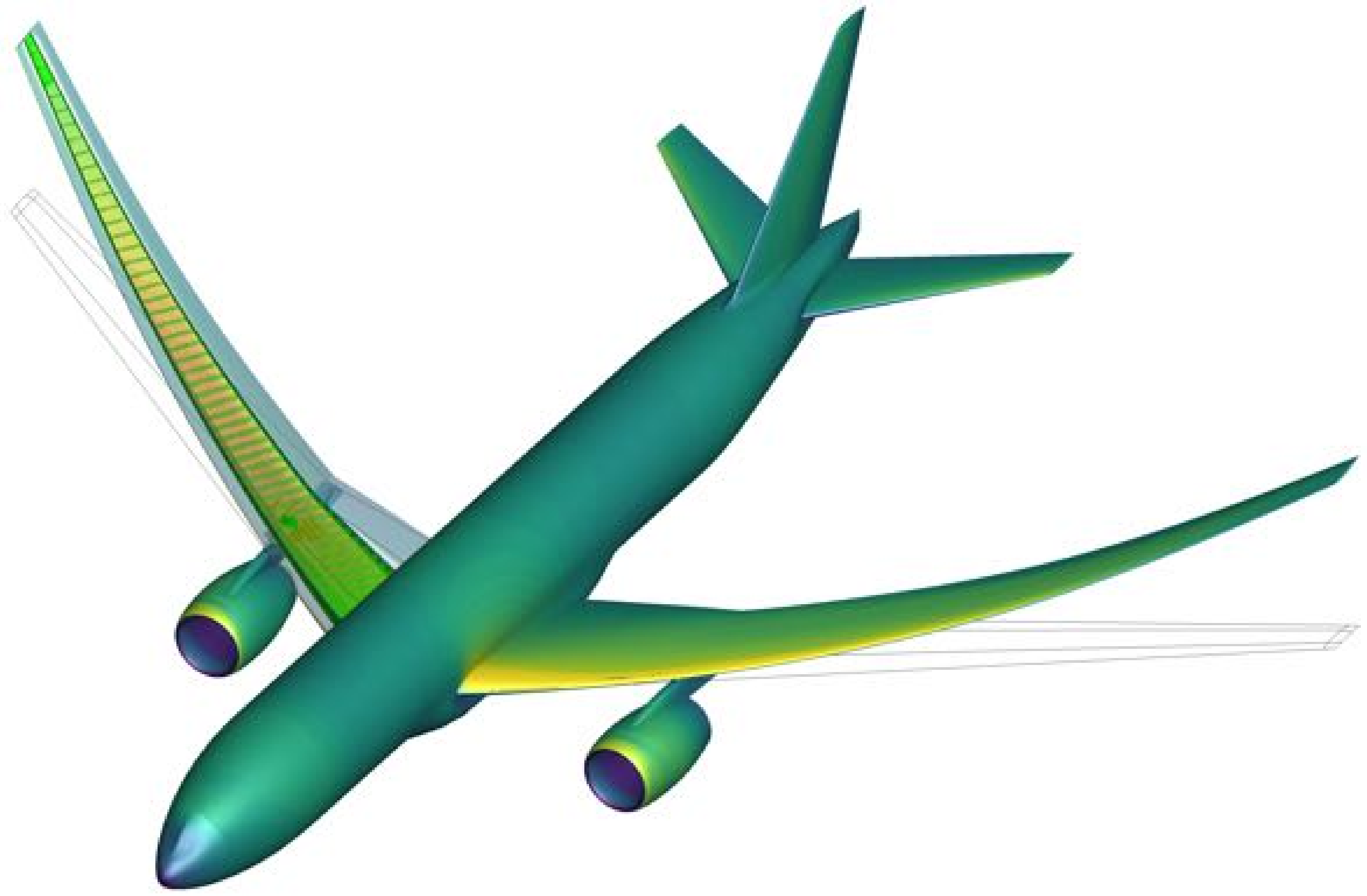
Want to do MDO with two or more high-fidelity disciplines, starting with aerodynamics and structures

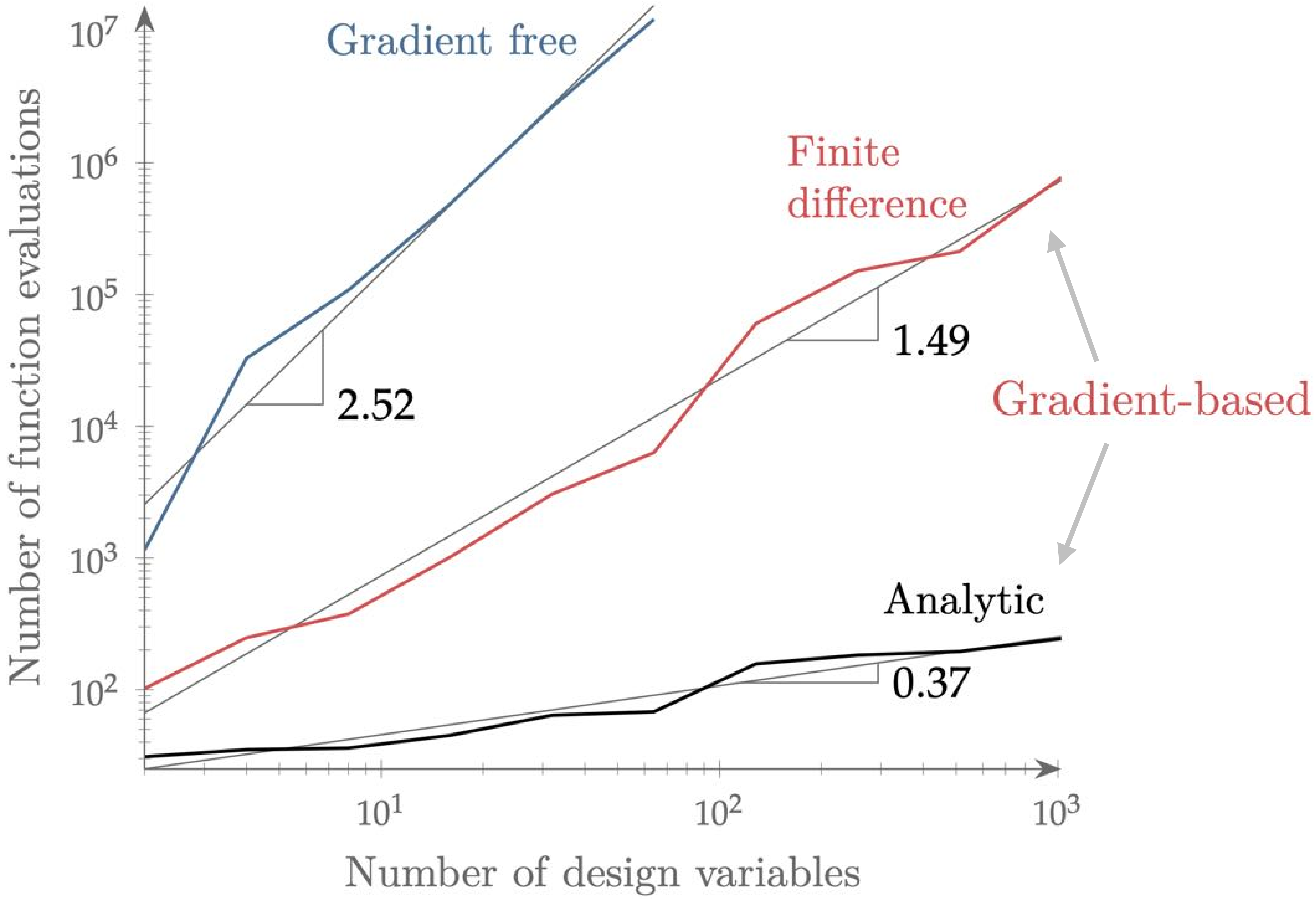


Before doing high-fidelity aerostructural optimization well, we need to develop robust aerodynamic shape optimization capability



Gradient-based optimization is the only hope for large numbers of design variables

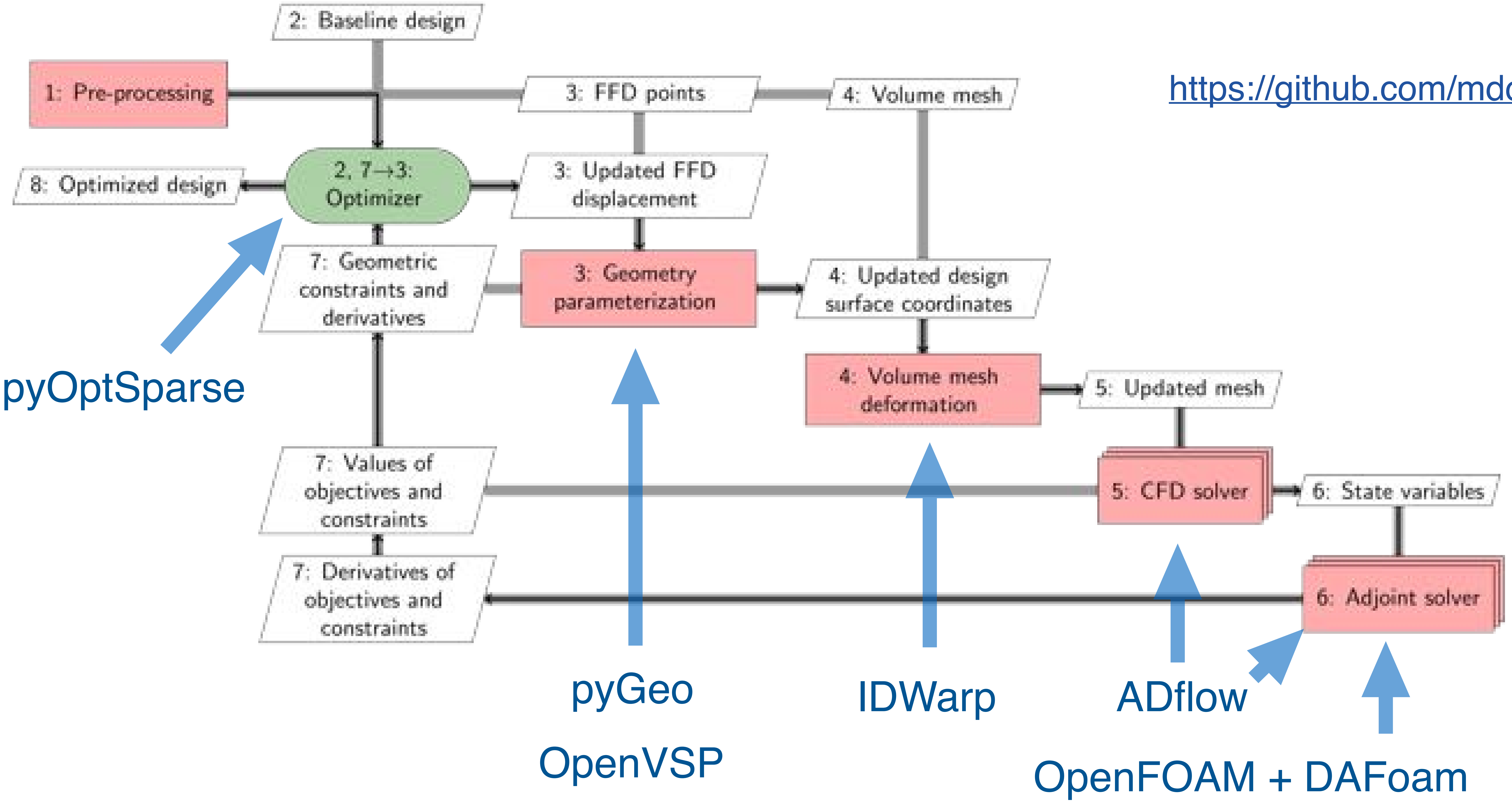




Martins and Ning. **Engineering Design Optimization.** Cambridge University Press, 2021.

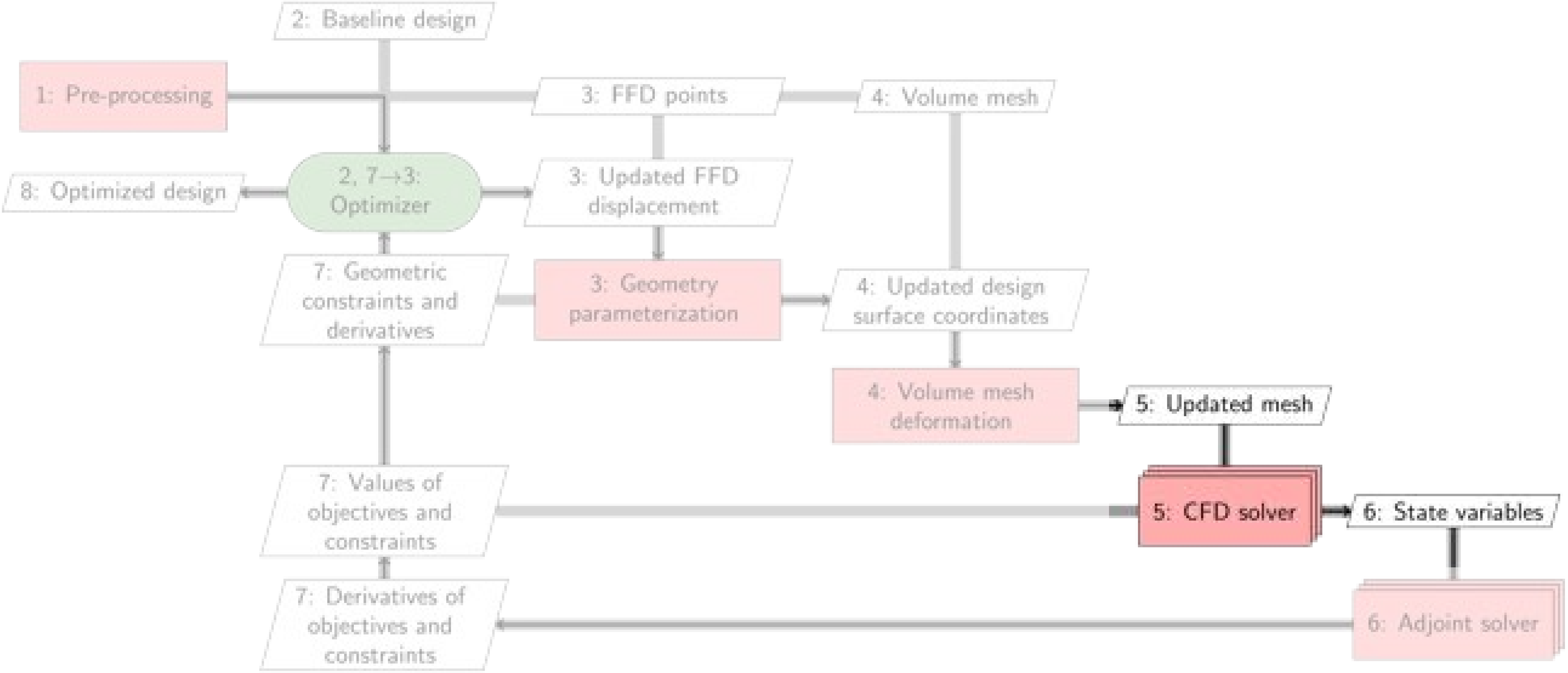
MACH-Aero is an open-source framework with all the tools required for aerodynamic design optimization

https://github.com/mdolab/MACH-Aero



All modules have a Python interface, which is used to couple them

CFD Solvers: ADflow and OpenFOAM

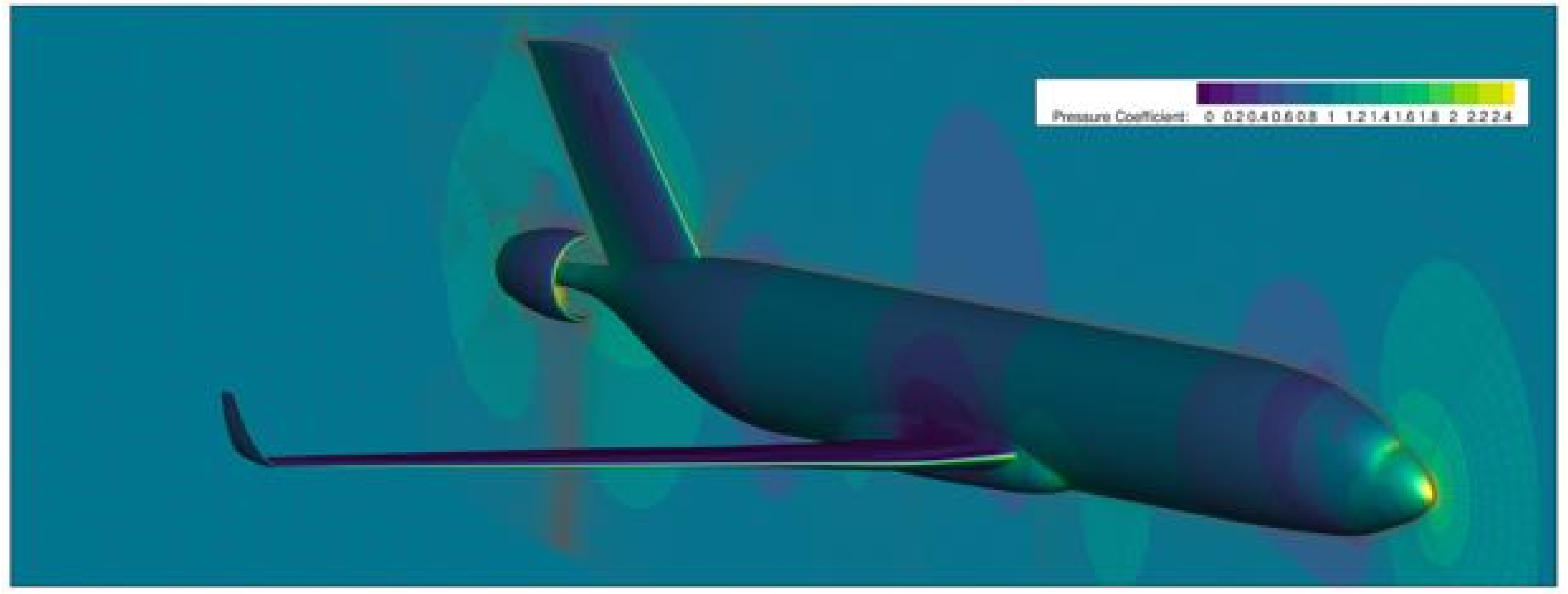


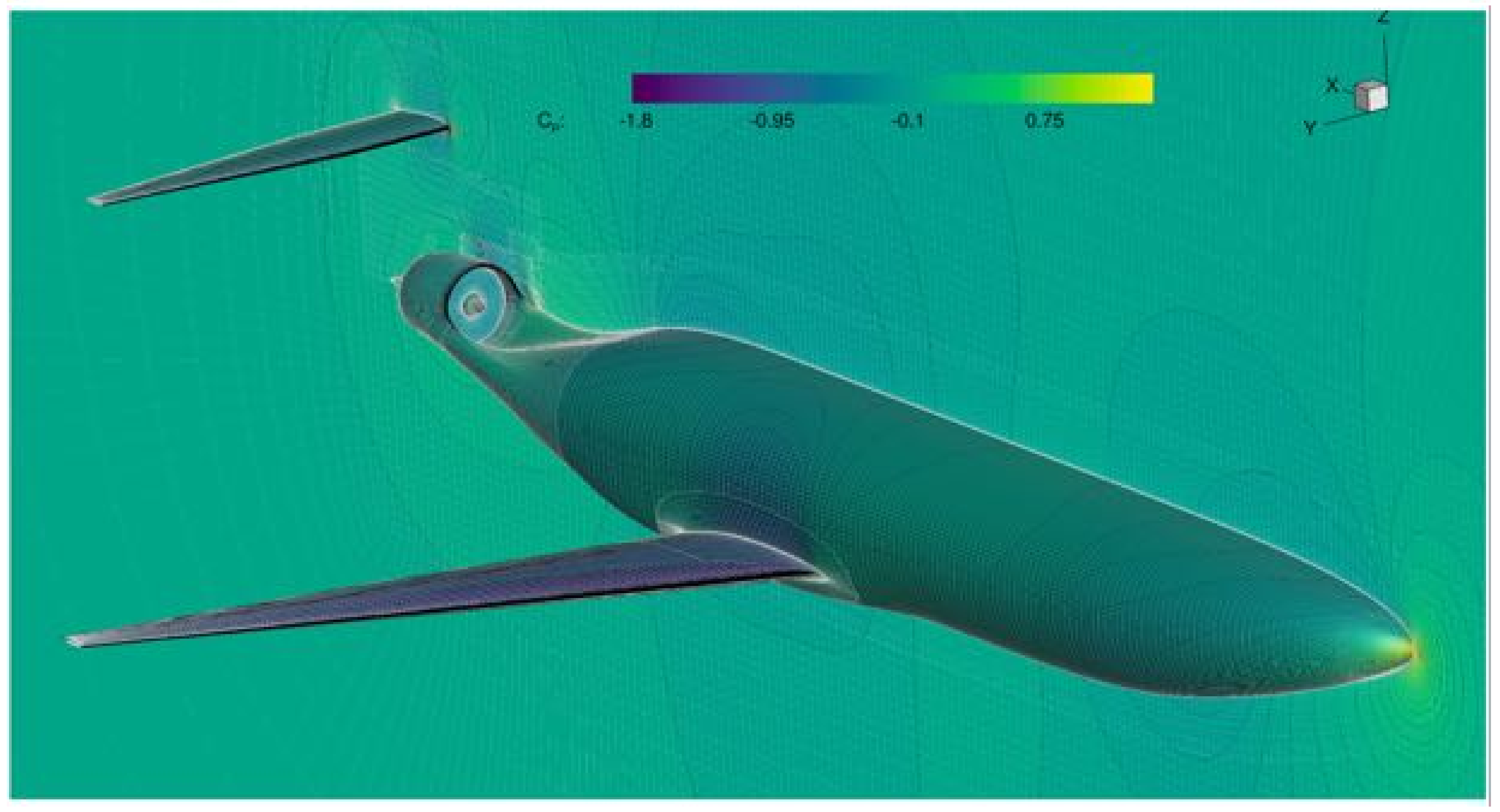
Both of these CFD solvers are open source

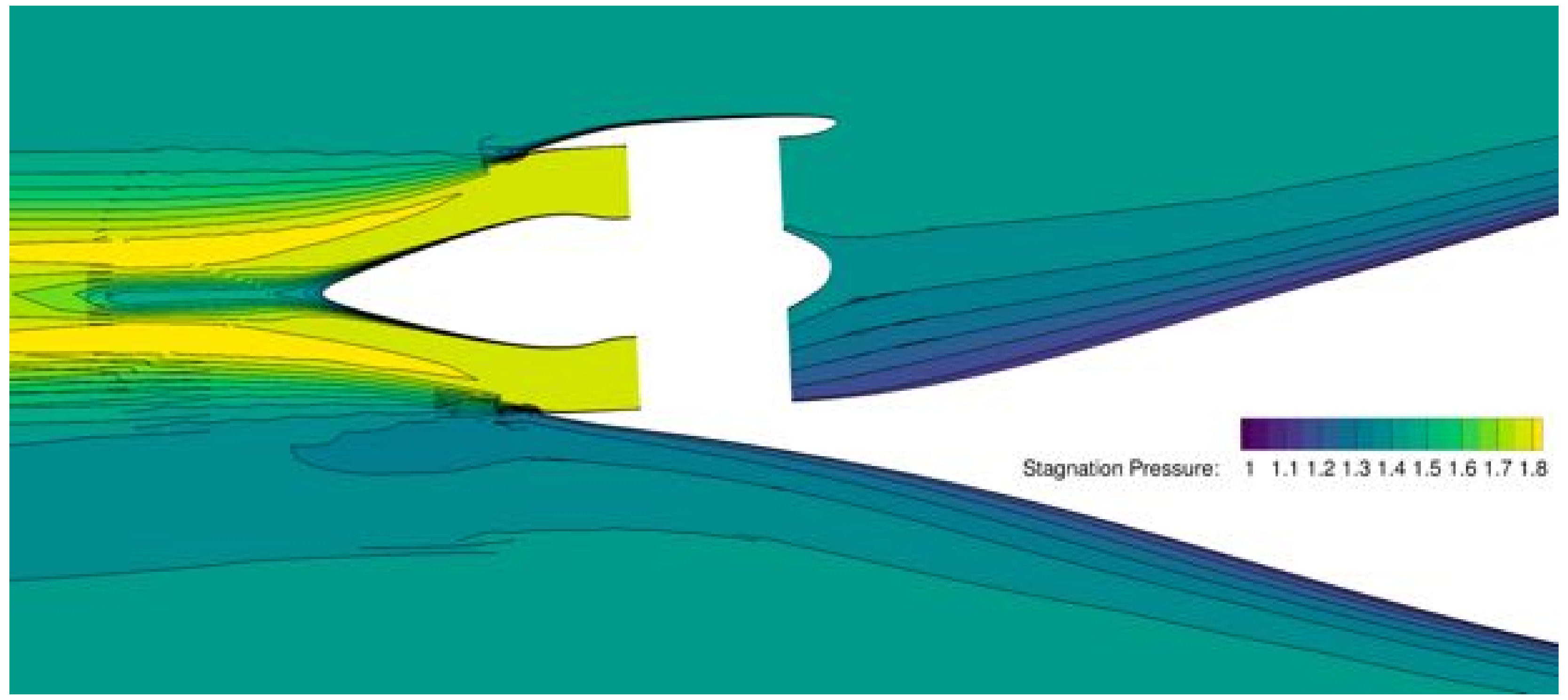
https://github.com/mdolab/adflow

ADflow is a RANS solver that includes an adjoint method for efficient derivative computation

- Parallel, finite-volume, cell-centered, overset, solver for RANS equations
- Approximate Newton–Krylov method for speed and robustness
- Spalart–Allmaras turbulence model
- Discrete adjoint developed using automatic differentiation (AD) to evaluate partial derivatives
- Full-turbulence adjoint







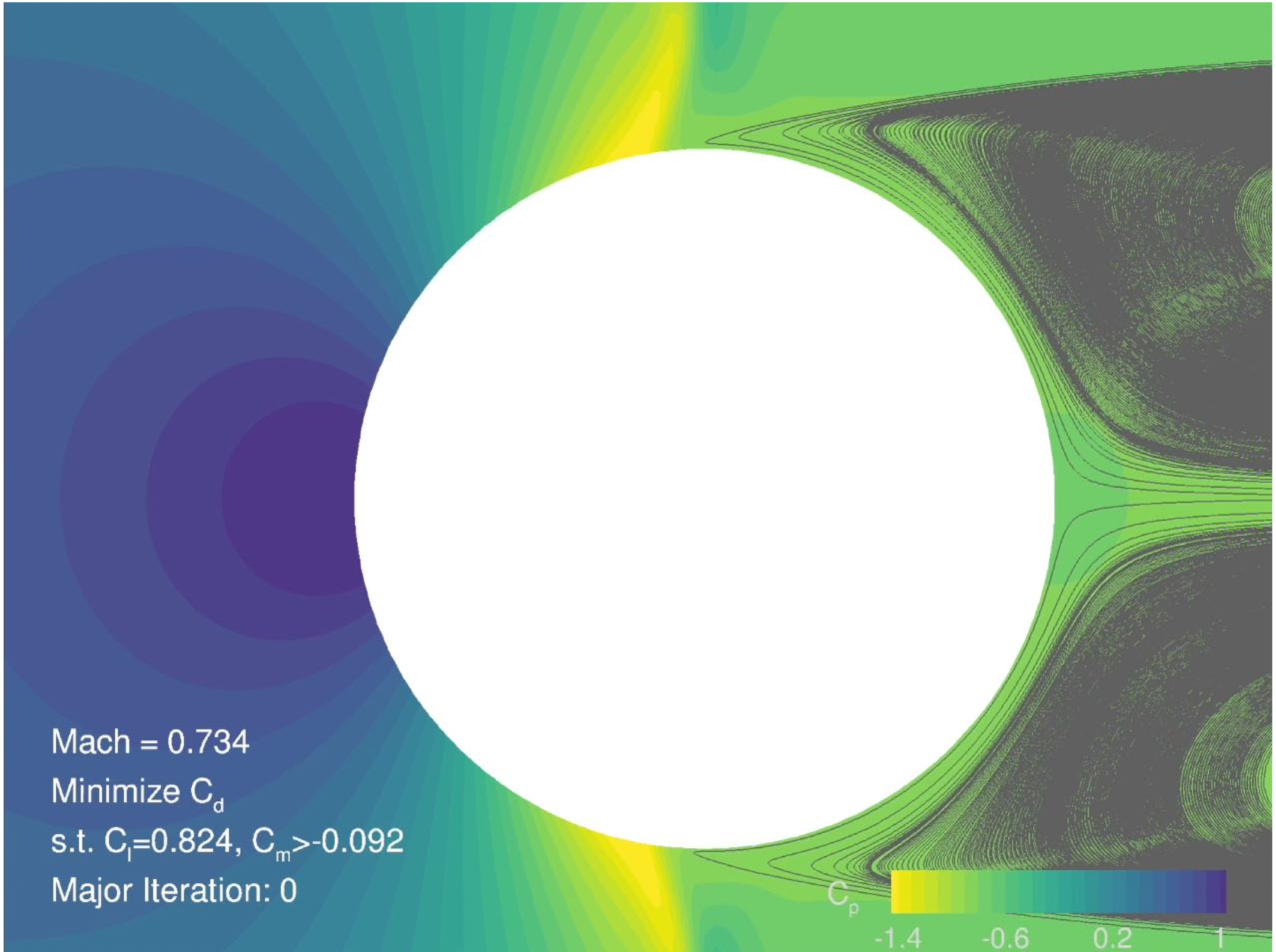
Yildirim, Kenway, Mader, and Martins. **A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations**. *Journal of Computational Physics*, 2019

ANK is extremely robust

CRM at 90 deg and M=0.85

This 8 million cell M6 mesh converges in about 14 minutes with 120 processors

Optimizing an airfoil starting from a circle is not a need…

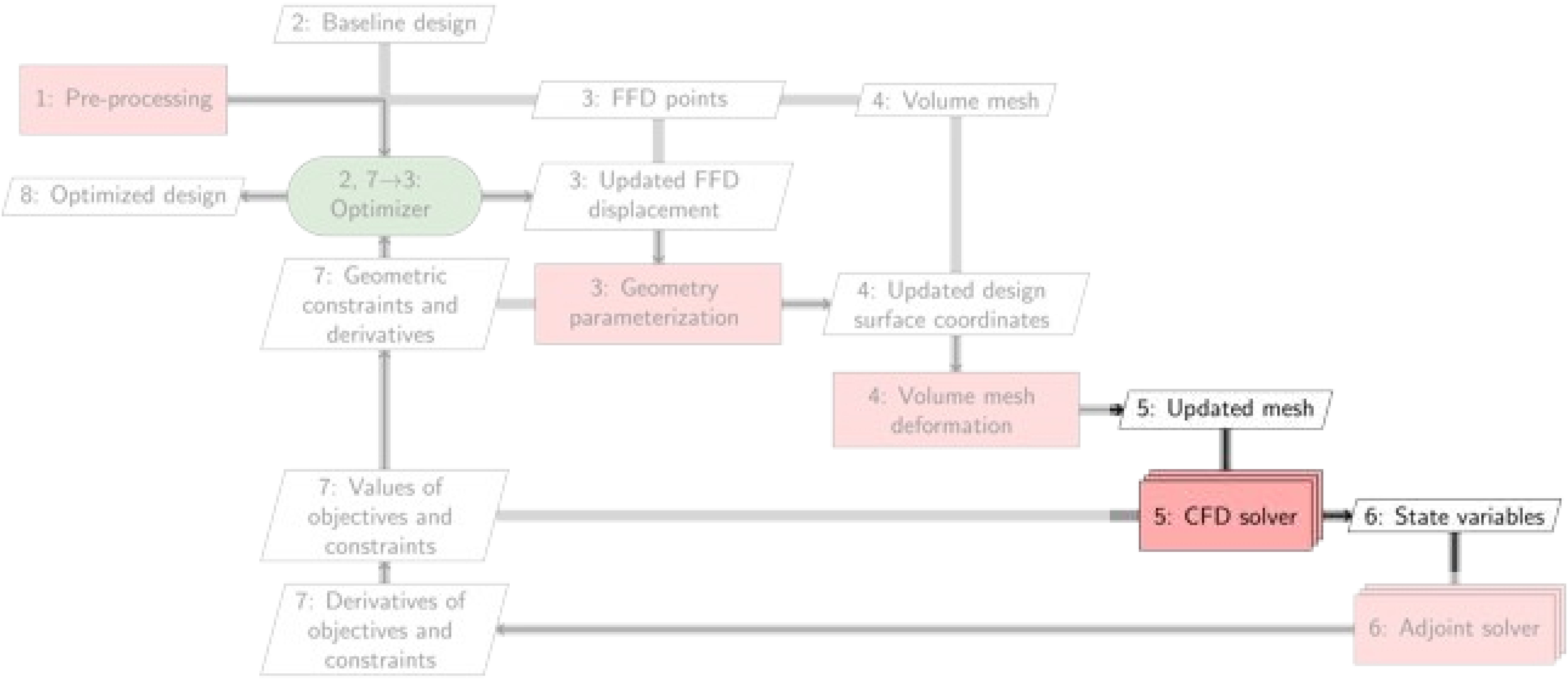


He, Li, Mader, Yildirim, Martins. **Robust aerodynamic shape optimization—from a circle to an airfoil**. *Aerospace Science and Technology*, 2019

…but optimizations does sometimes try intermediate crazy designs

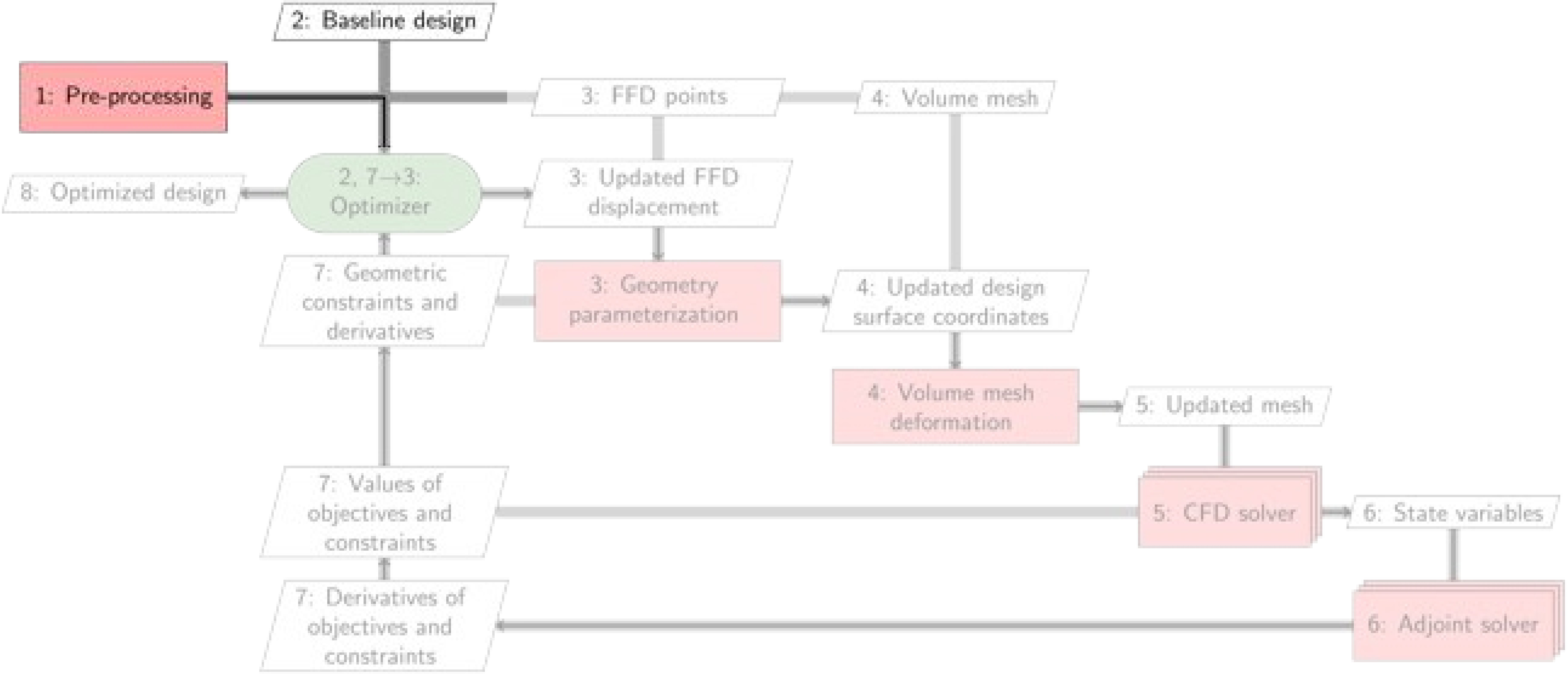
He, Li, Mader, Yildirim, Martins. **Robust aerodynamic shape optimization—from a circle to an airfoil**. *Aerospace Science and Technology*, 2019

OpenFOAM can be used interchangeably in MACH-Aero using the same interface



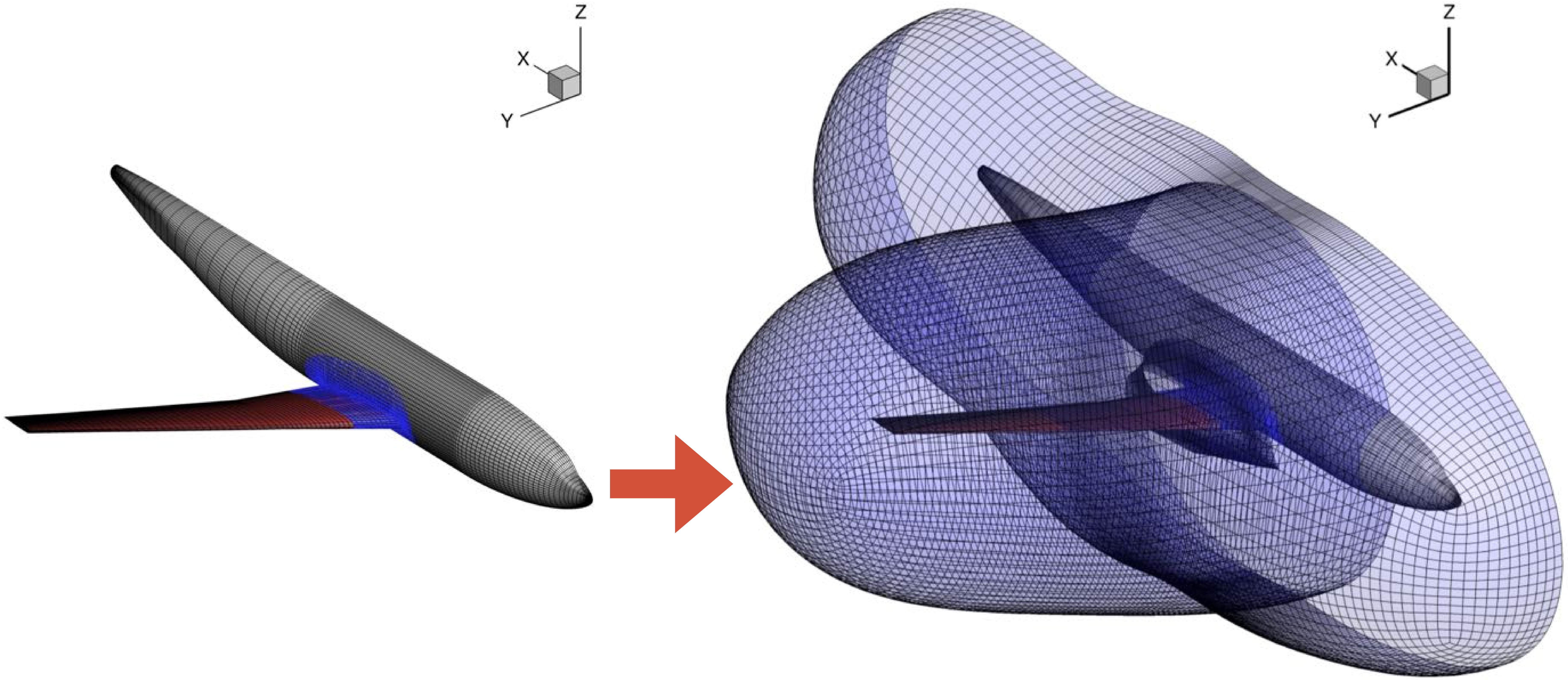
- Pressure based solver better suited for low speed applications
- Can handle unstructured meshes
- Slower than ADflow, but still fast enough for optimization

Mesh Generation: pyHyp



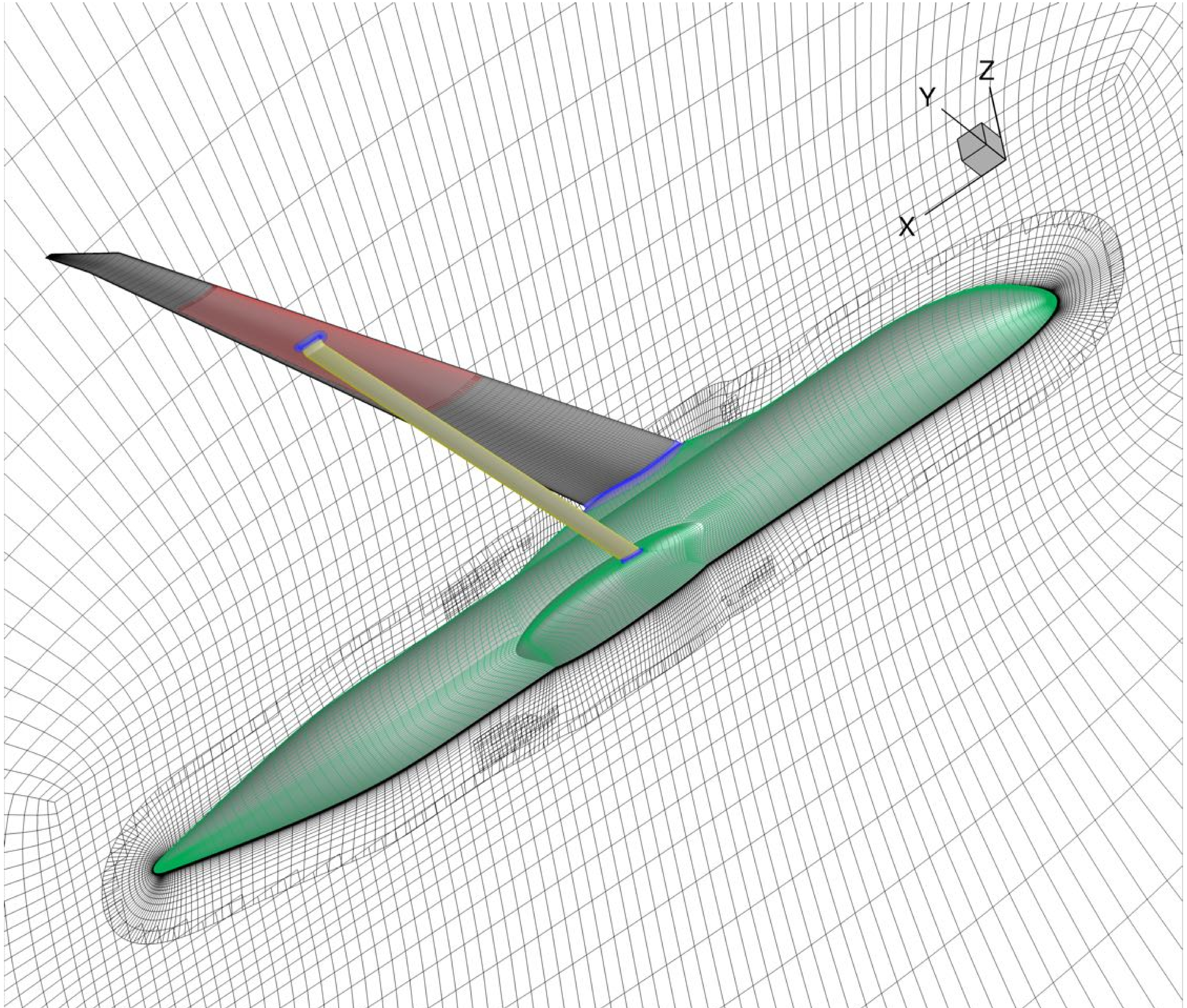
https://github.com/mdolab/pyhyp

pyHyp extrudes the surface meshes to make volume meshes

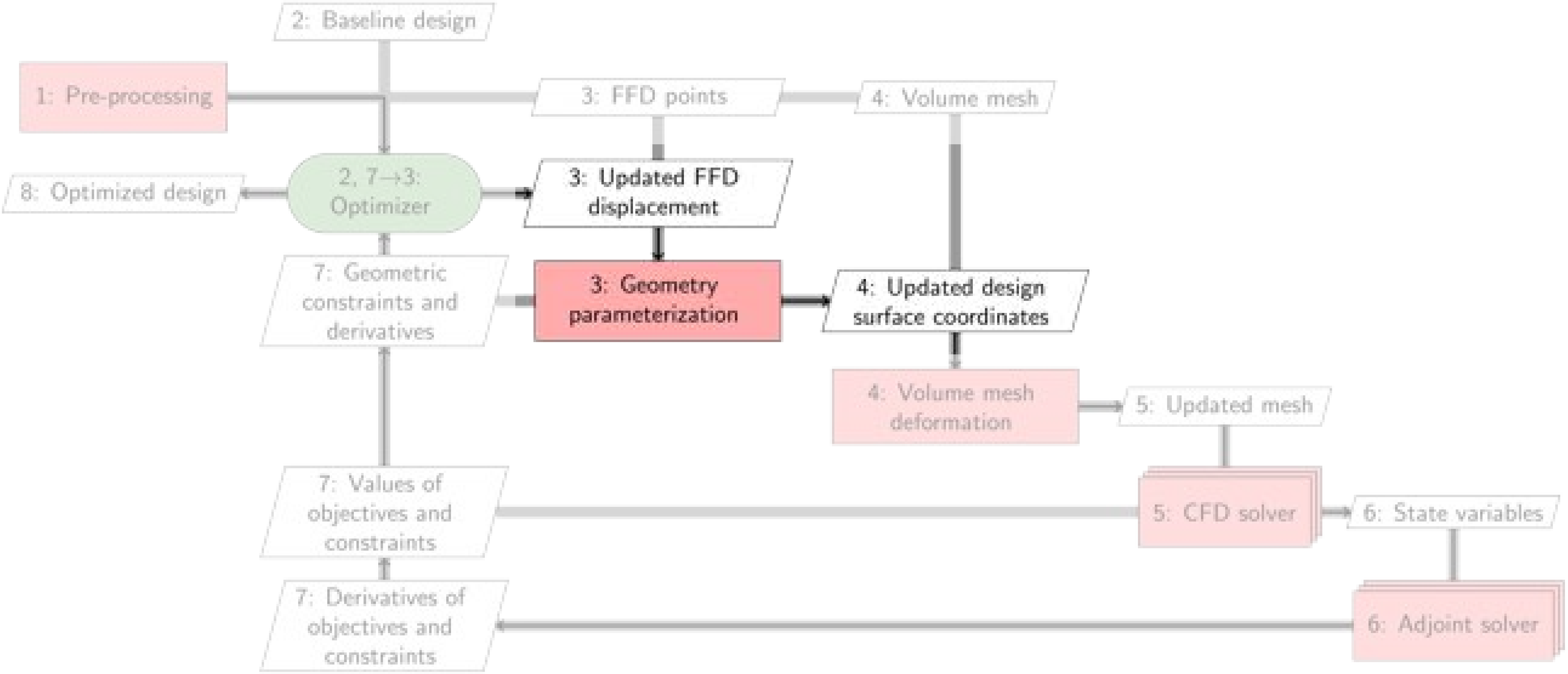


- Uses hyperbolic mesh generation algorithms
- High-quality mesh in terms of stretch ratio and orthogonality
- Can handle collar meshes

We overset the meshes to obtain the full configuration mesh

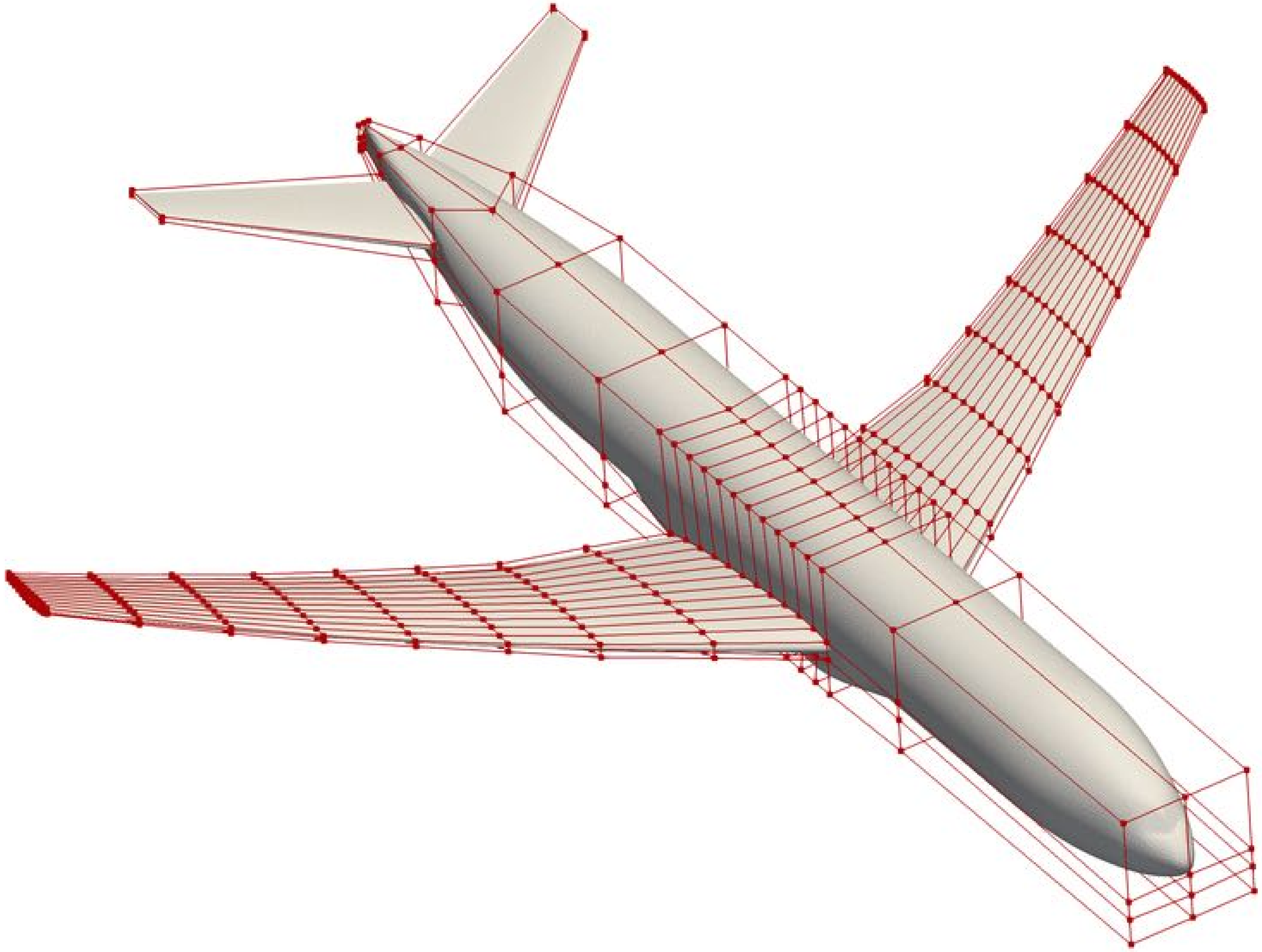


Geometry Parametrization: pyGeo or OpenVSP

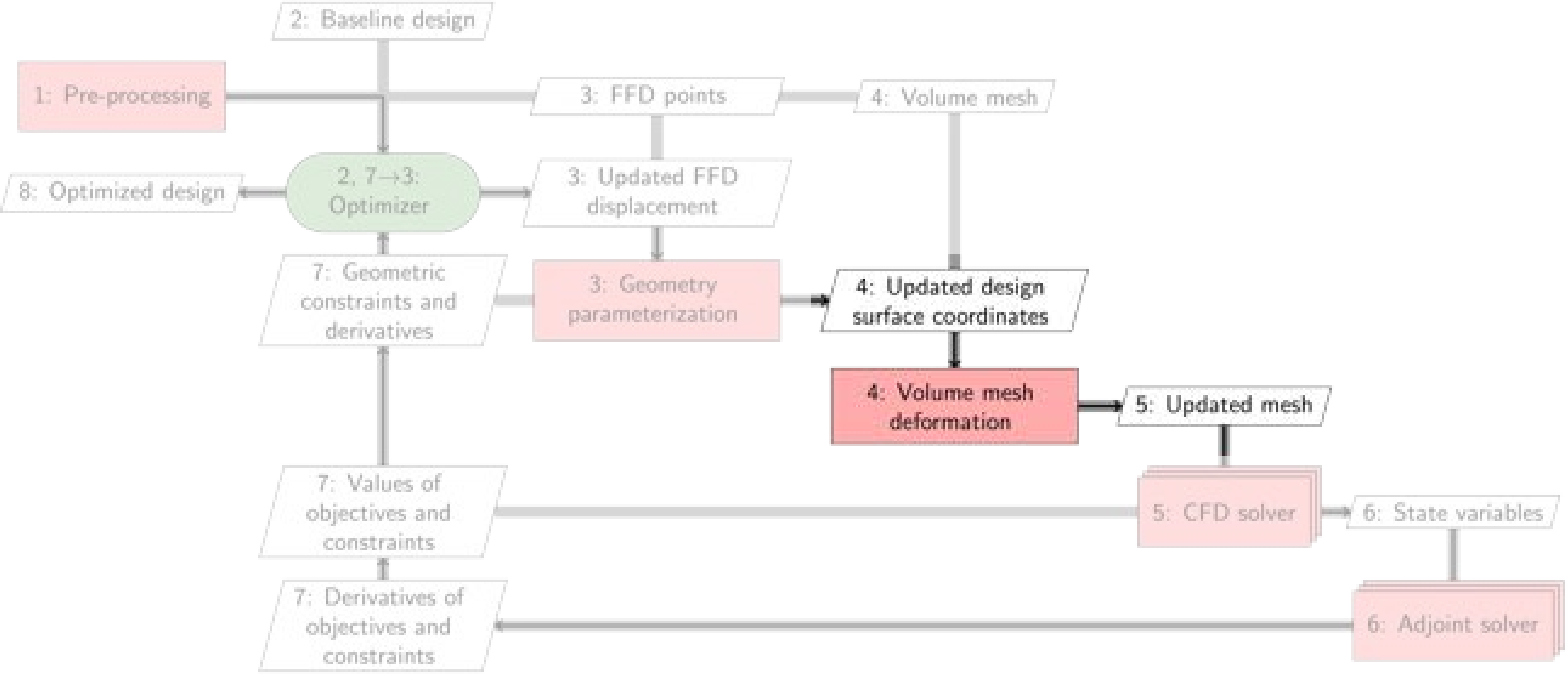


https://github.com/mdolab/pygeo

pyGeo parametrizes geometries using free-form deformation volumes



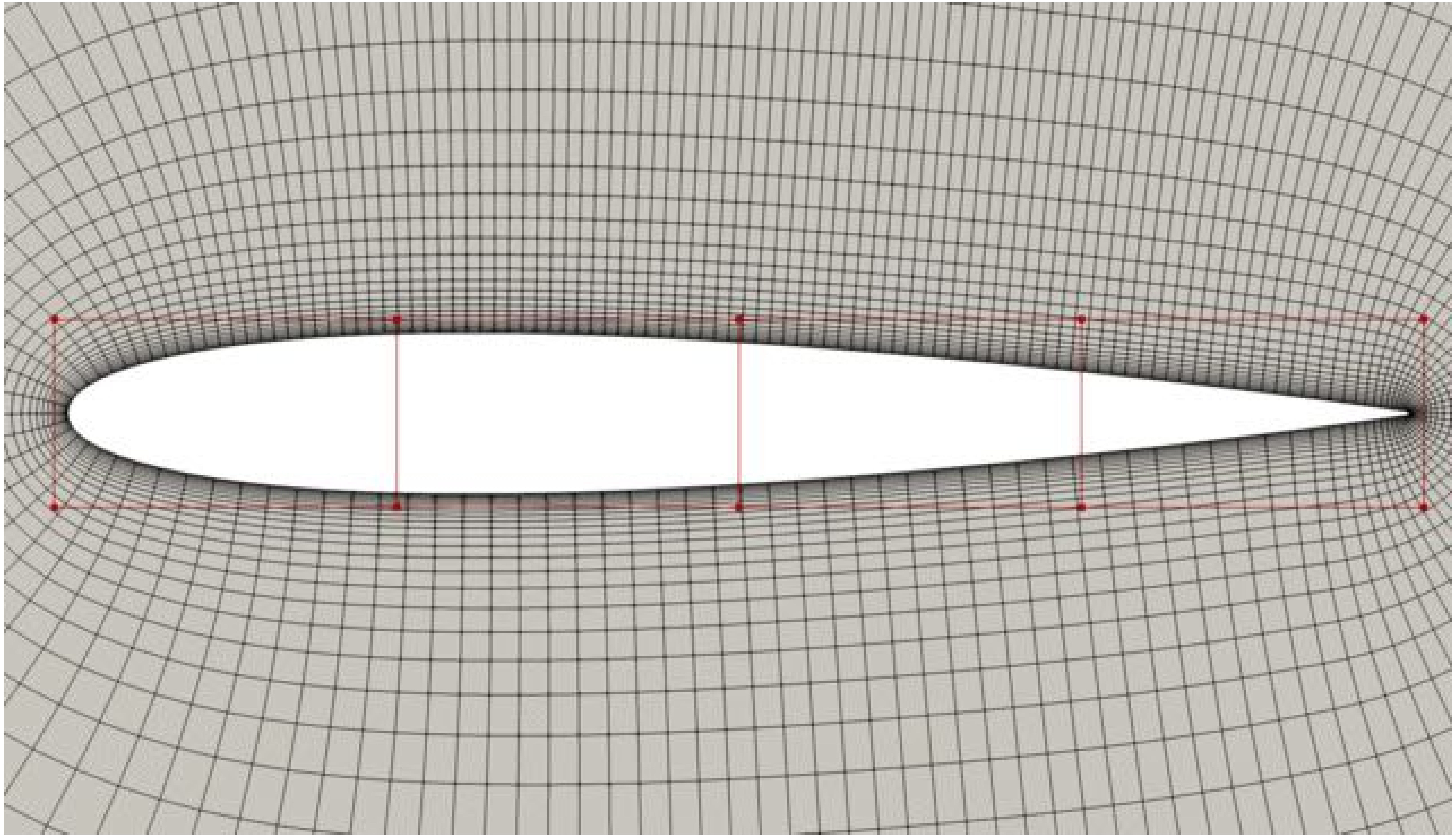
Mesh Deformation: IDWarp



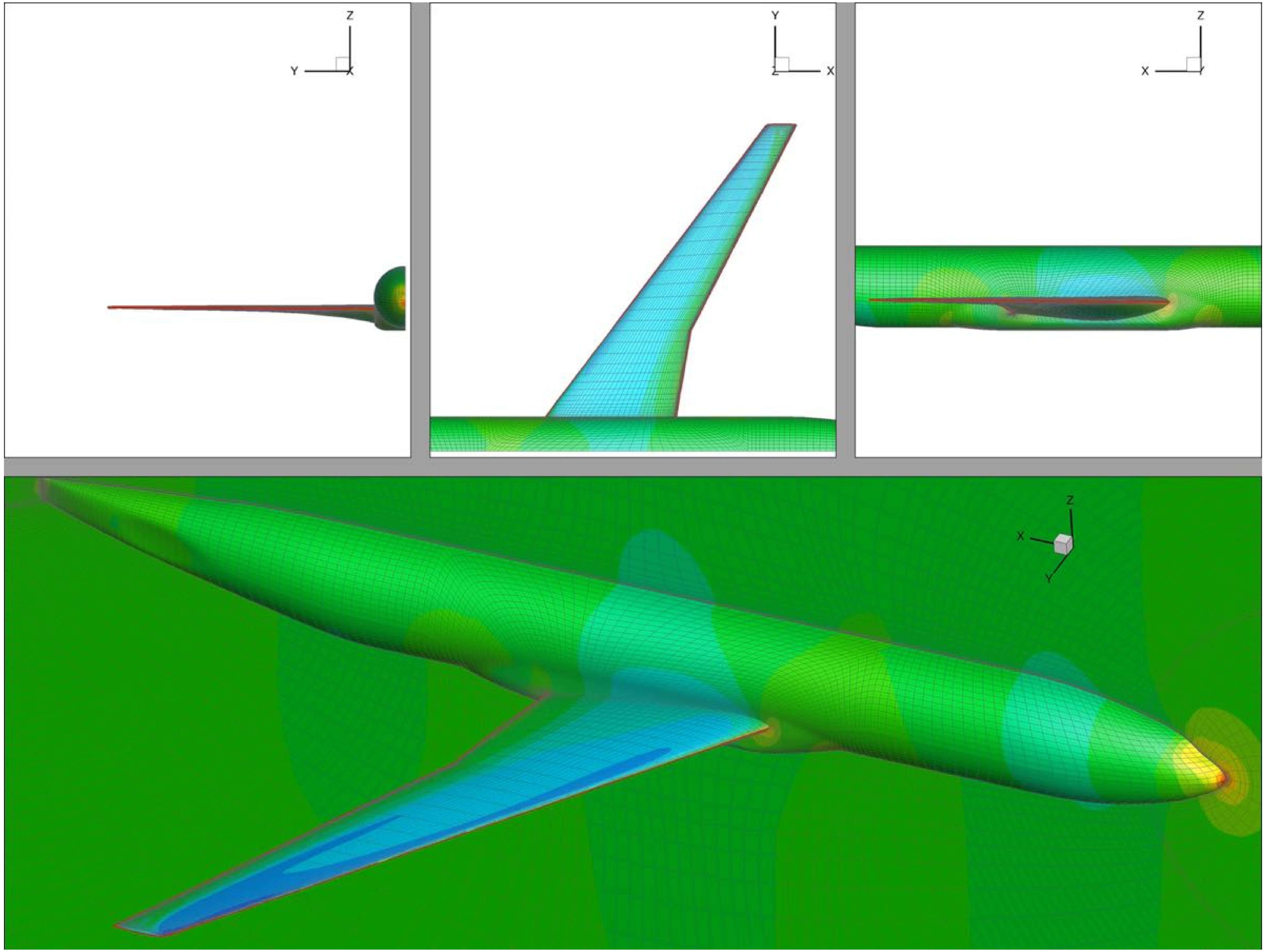
https://github.com/mdolab/idwarp

IDWarp works together with pyGeo for a seamless propagation of design shape variables to the mesh

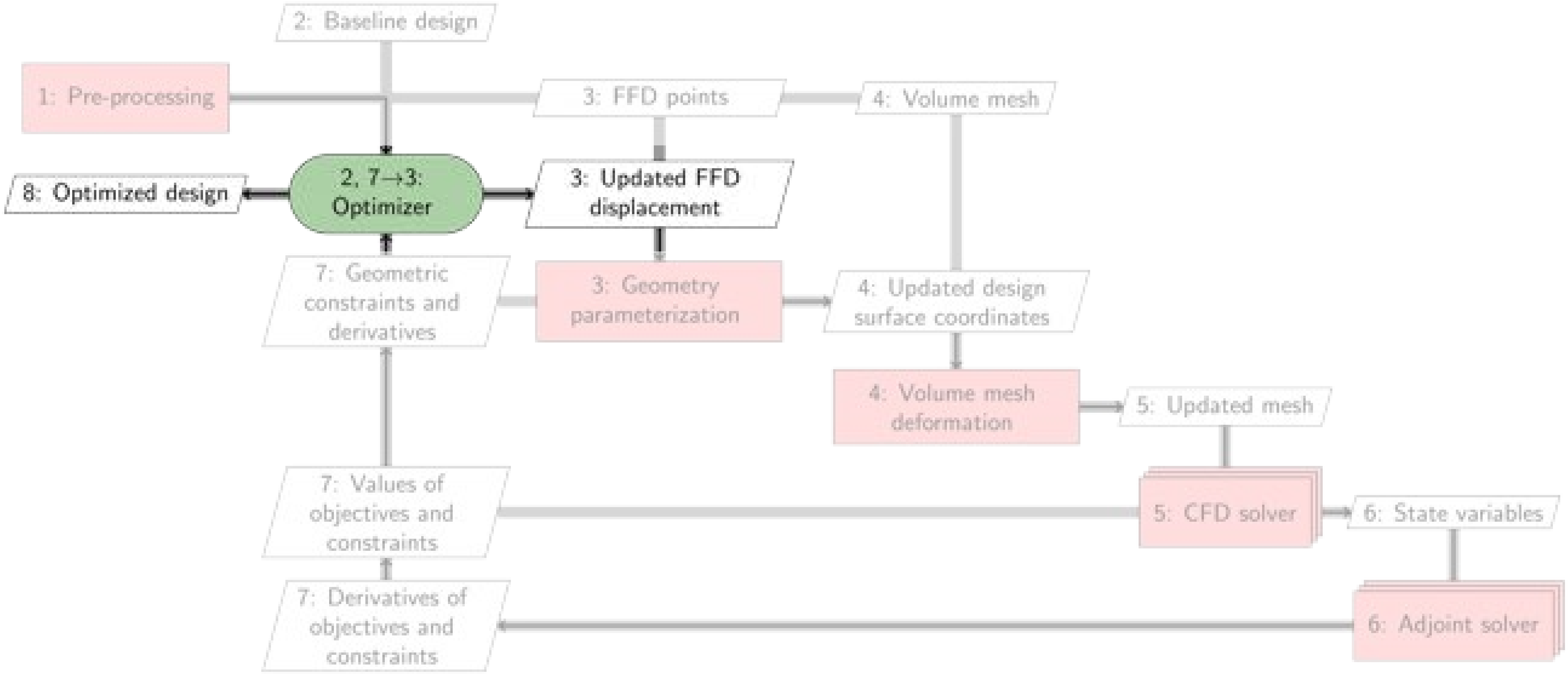
- Can be used for both structured and unstructured meshes
- Fast and robust



IDWarp deforms the volume mesh based on new surface mesh



Optimization Algorithms



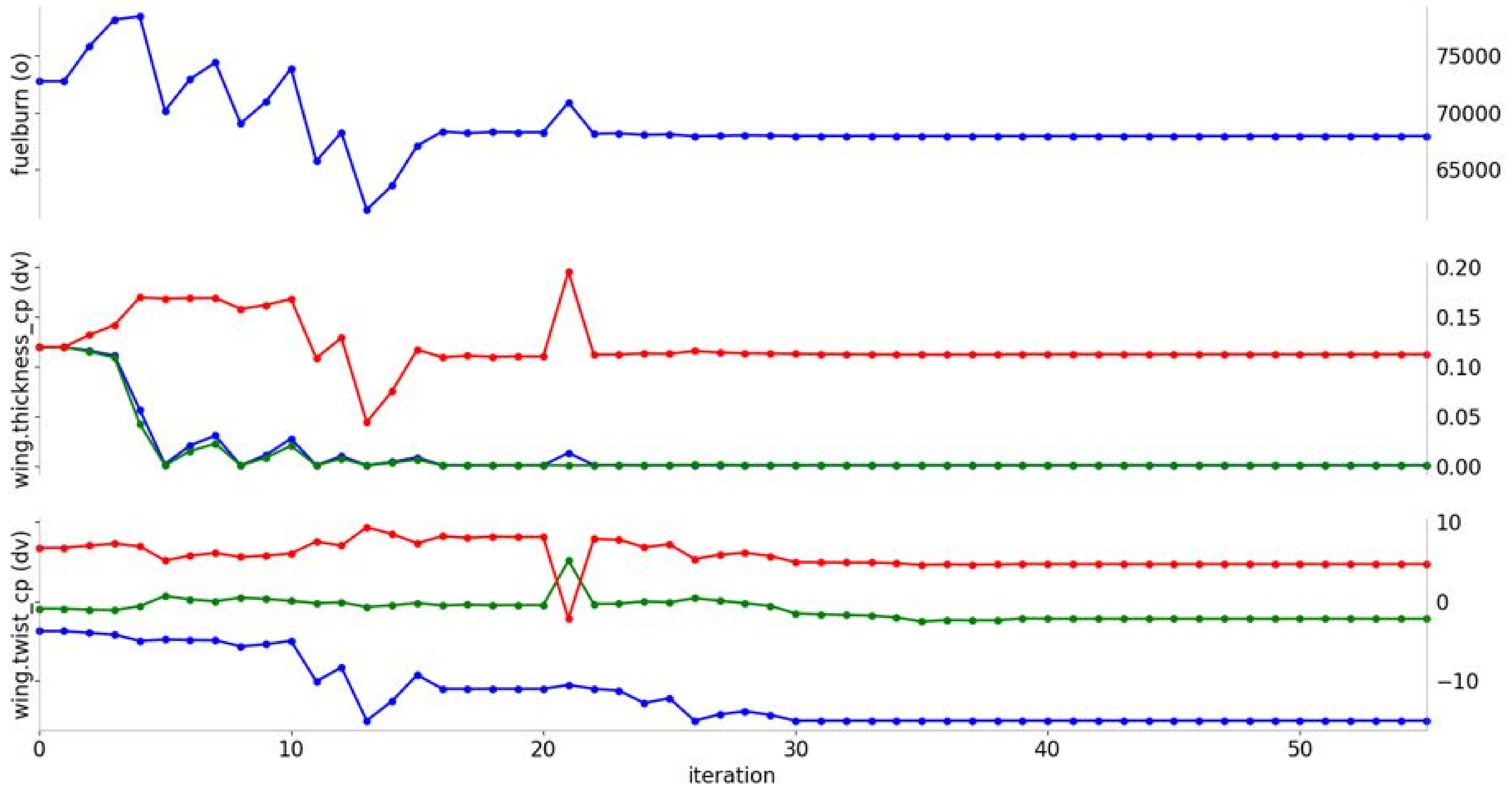
https://github.com/mdolab/pyoptsparse

pyOptSparse and OptView facilitate the use of optimization algorithms

pyOptSparse

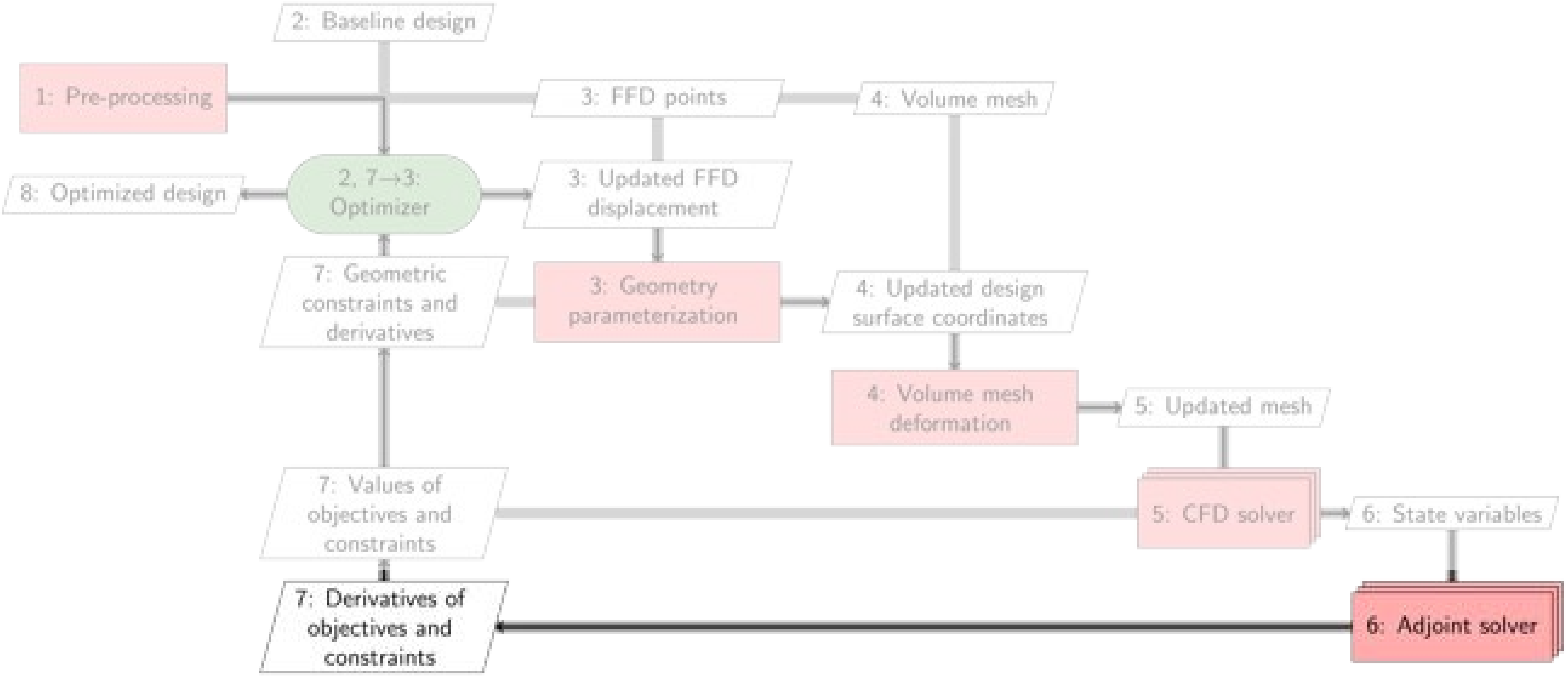
[https://github.com/mdolab/pyoptsparse]

- Python wrapper for for various optimizers
- Supports both gradient-based and gradient-free optimizers
- Facilitates comparisons
- Includes OptView for history visualization
- Open source



Kenway, Mader, Jasa, Martins. **pyOptSparse: a Python framework for large-scale constrained nonlinear optimization of sparse systems**. *Journal of Open Source Software*, 2020.

Derivative Computation



Methods for computing derivatives

Black box

Inputs and outputs

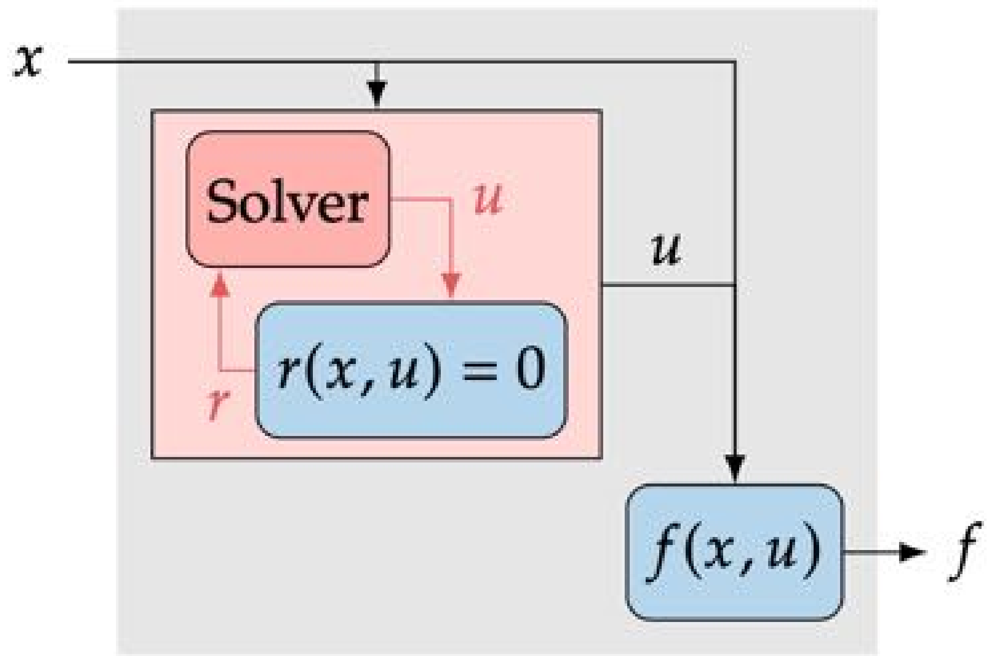
Finite differences

$$\frac{\partial f}{\partial x_j} = \frac{f(x) - f(x - h\hat{e}_j)}{h} + \mathcal{O}(h)$$

Complex-step method

$$\frac{\partial f}{\partial x_j} = \frac{\operatorname{Im}\left[f(x + ih\hat{e}_j)\right]}{h} + \mathcal{O}(h^2)$$

Analytic



Governing equation residuals and states

Direct method

$\phi \quad (n_u \times n_x)$

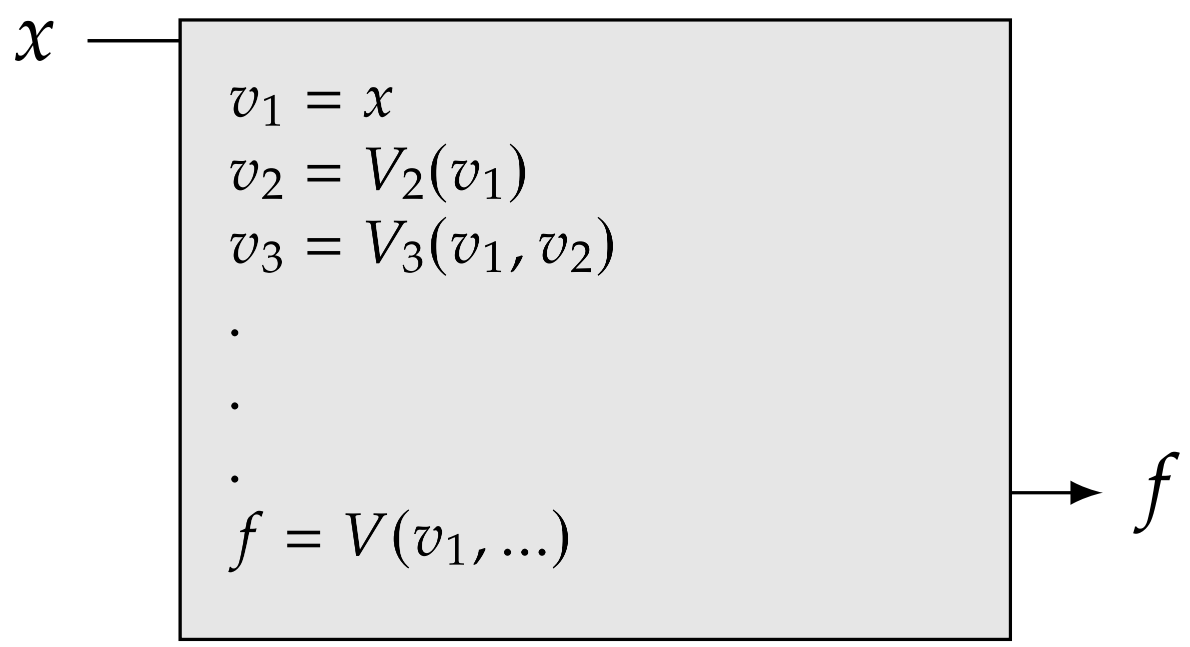
$$\frac{df}{dx} = \frac{\partial f}{\partial x} - \frac{\partial f}{\partial u}\,\frac{\partial r}{\partial u}^{-1}\,\frac{\partial r}{\partial x}$$

$(n_f \times n_x) \quad (n_f \times n_x) \quad (n_f \times n_u) \quad (n_u \times n_u) \quad (n_u \times n_x)$

$\psi^{\top} \quad (n_f \times n_u)$

Adjoint method

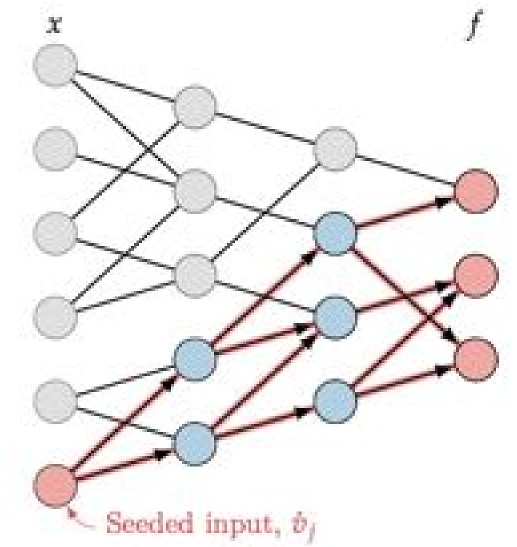
Algorithmic



Lines of code

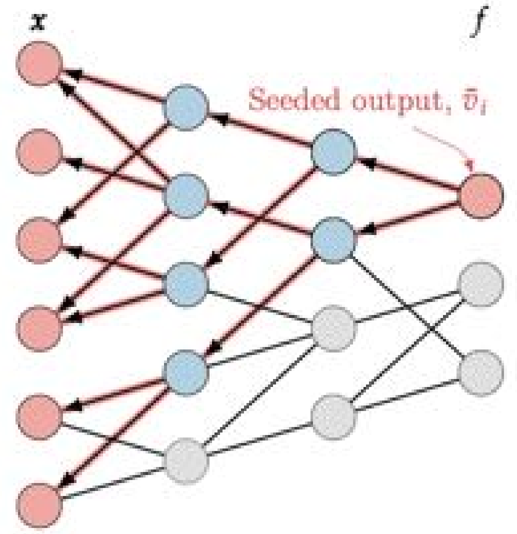
Forward mode

$$\frac{dv_i}{dv_j} = \sum_{k=j}^{i-1} \frac{\partial v_i}{\partial v_k}\frac{dv_k}{dv_j}$$



Reverse mode

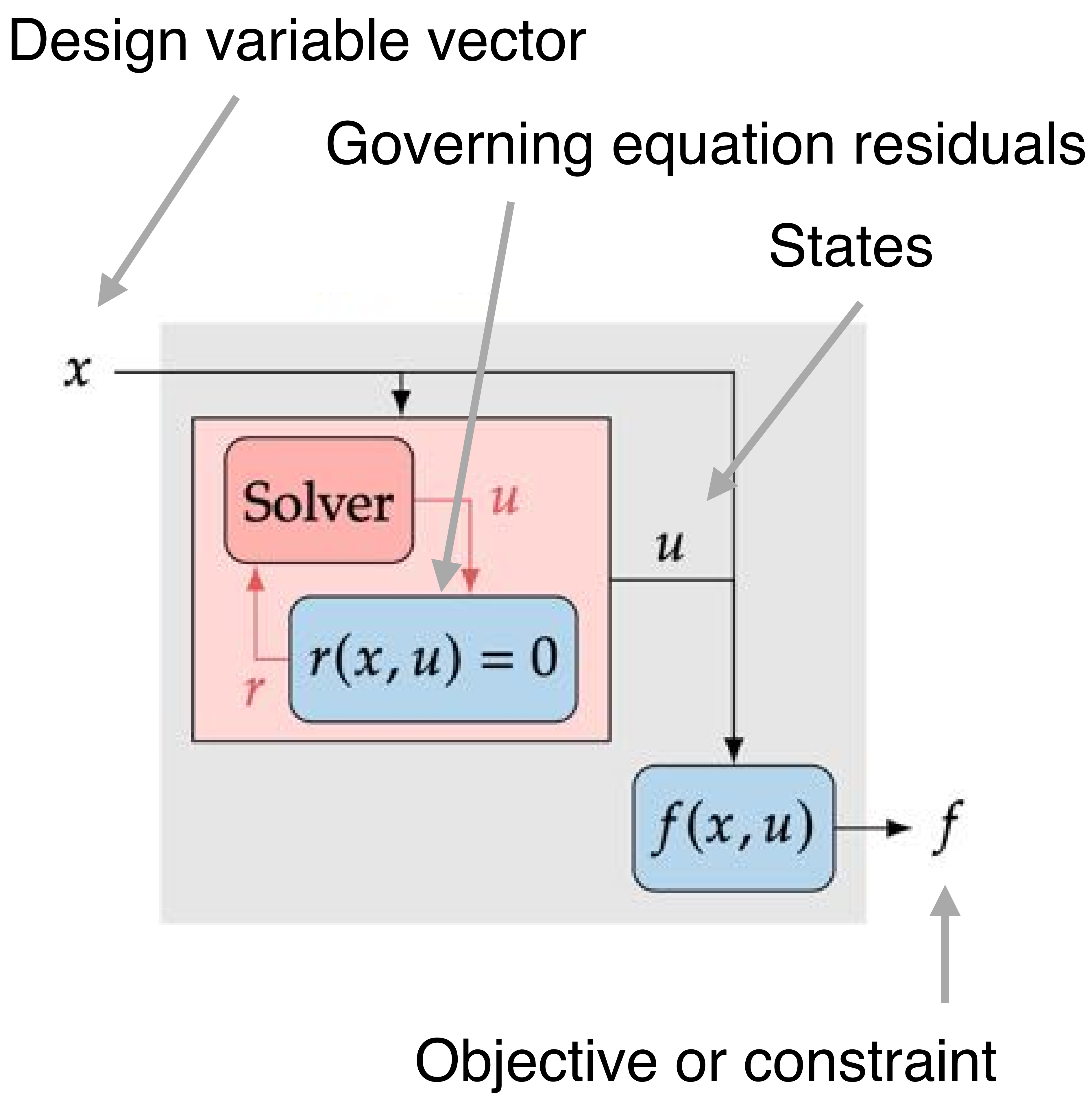
$$\frac{dv_i}{dv_j} = \sum_{k=j+1}^{i} \frac{\partial v_k}{\partial v_j}\frac{dv_i}{dv_k}$$



Martins, Sturdza, and Alonso. **The complex-step derivative approximation**. *ACM Transactions on Mathematical Software*, 2003.

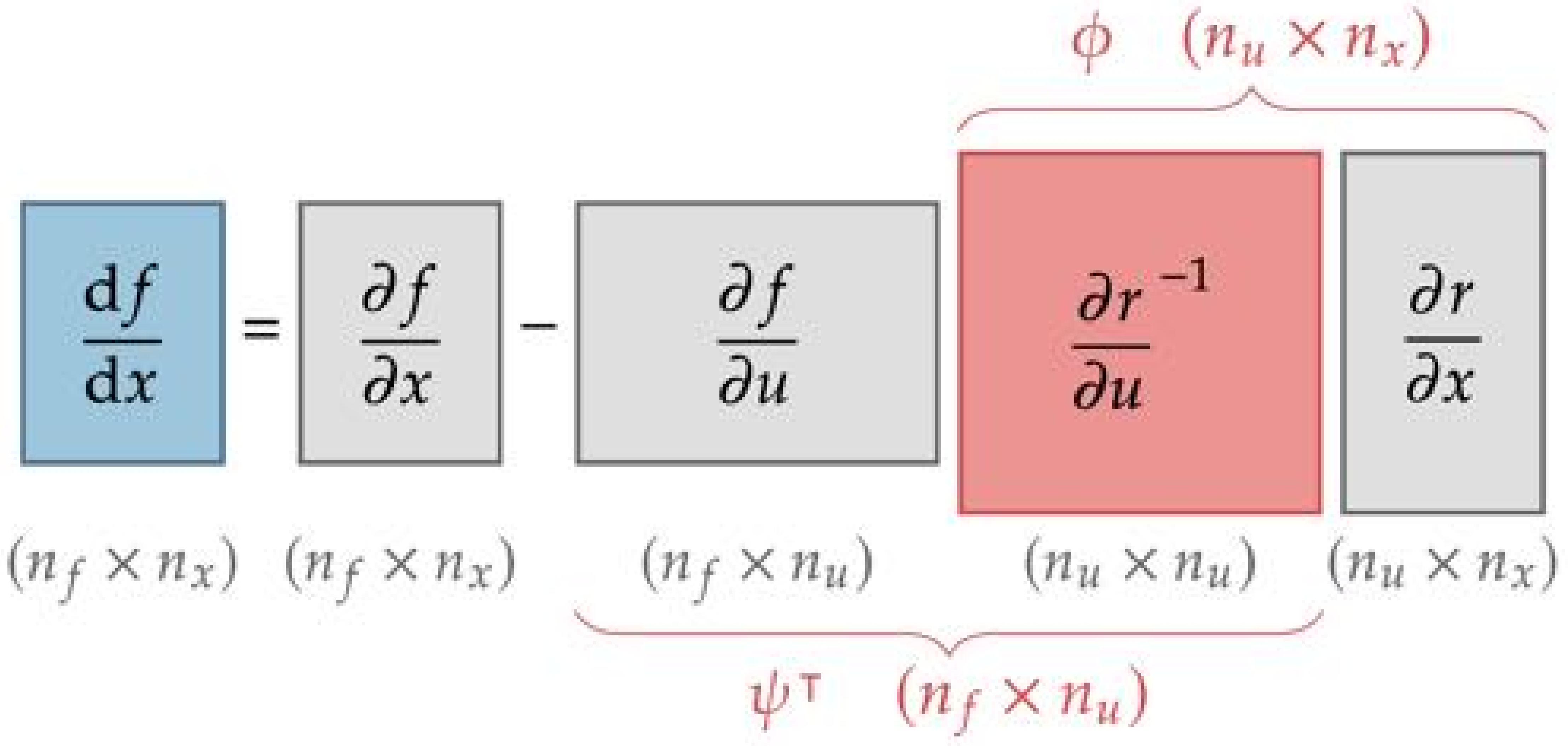
Martins and Hwang. **Review and unification of methods for computing derivatives of multidisciplinary computational models**. *AIAA Journal*, 2013.

The adjoint method is efficient when computing derivatives for large numbers of design variables



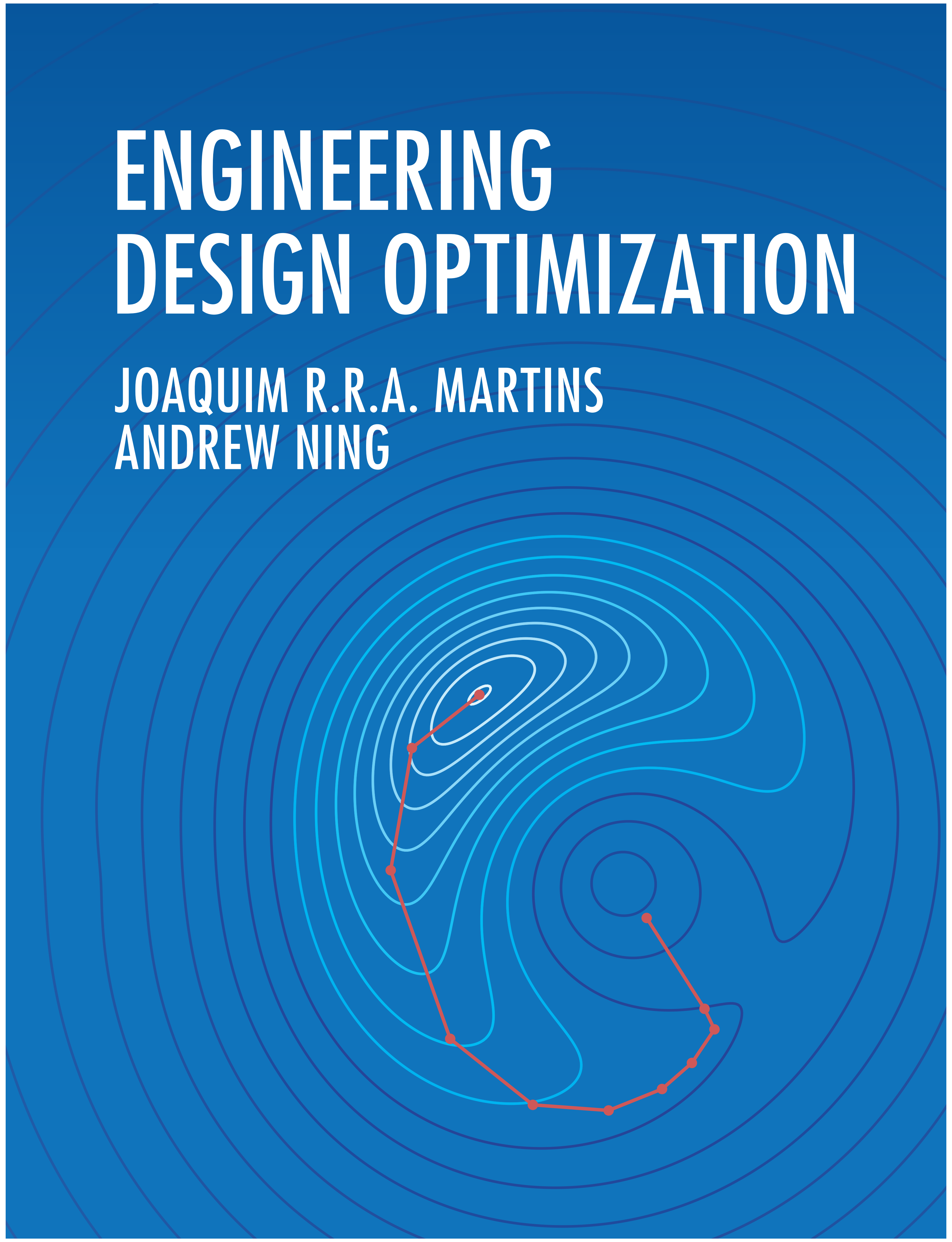
$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}x}$$

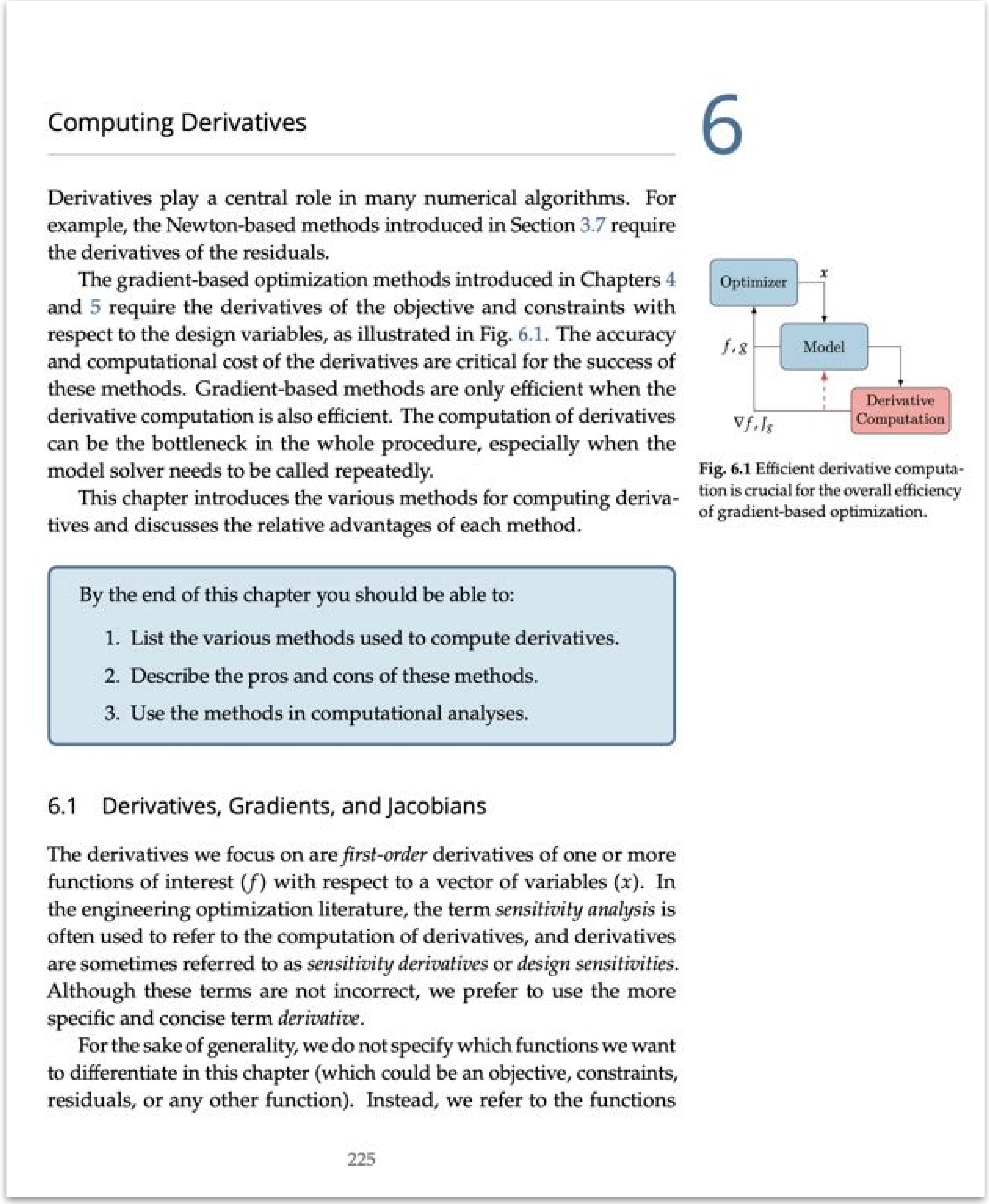
$$\mathrm{d}r = \frac{\partial r}{\partial x}\mathrm{d}x + \frac{\partial r}{\partial u}\mathrm{d}u = 0 \quad \rightarrow \quad \frac{\partial r}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}x} = -\frac{\partial r}{\partial x}$$



Kenway, Mader, He, and Martins. **Effective adjoint approaches for computational fluid dynamics**. *Progress in Aerospace Sciences*, 2019

For more details, see Chapter 6 of my new book (the PDF if free at https://mdobook.github.io)





Computing Derivatives

6

Derivatives play a central role in many numerical algorithms. For example, the Newton-based methods introduced in Section 3.7 require the derivatives of the residuals.

The gradient-based optimization methods introduced in Chapters 4 and 5 require the derivatives of the objective and constraints with respect to the design variables, as illustrated in Fig. 6.1. The accuracy and computational cost of the derivatives are critical for the success of these methods. Gradient-based methods are only efficient when the derivative computation is also efficient. The computation of derivatives can be the bottleneck in the whole procedure, especially when the model solver needs to be called repeatedly.

This chapter introduces the various methods for computing derivatives and discusses the relative advantages of each method.

Optimizer — x — Model — Derivative Computation — f, g — $\nabla f, J_g$

Fig. 6.1 Efficient derivative computation is crucial for the overall efficiency of gradient-based optimization.

By the end of this chapter you should be able to:

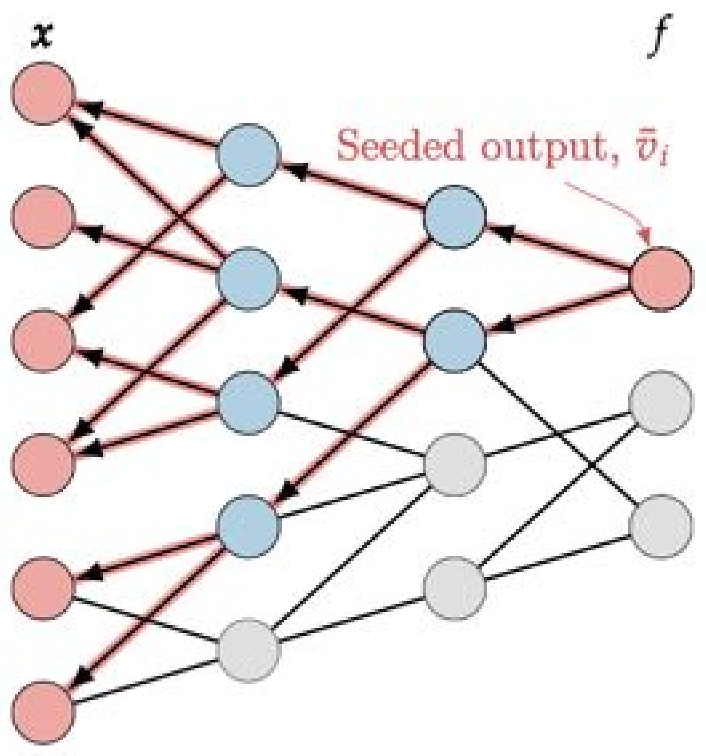
1. List the various methods used to compute derivatives.
2. Describe the pros and cons of these methods.
3. Use the methods in computational analyses.

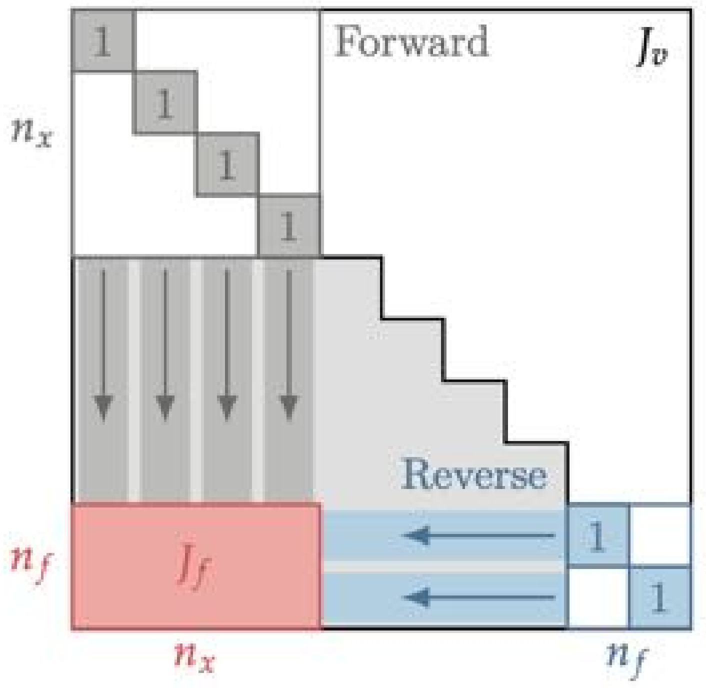
6.1 Derivatives, Gradients, and Jacobians

The derivatives we focus on are *first-order* derivatives of one or more functions of interest (f) with respect to a vector of variables (x). In the engineering optimization literature, the term *sensitivity analysis* is often used to refer to the computation of derivatives, and derivatives are sometimes referred to as *sensitivity derivatives* or *design sensitivities*. Although these terms are not incorrect, we prefer to use the more specific and concise term *derivative*.

For the sake of generality, we do not specify which functions we want to differentiate in this chapter (which could be an objective, constraints, residuals, or any other function). Instead, we refer to the functions

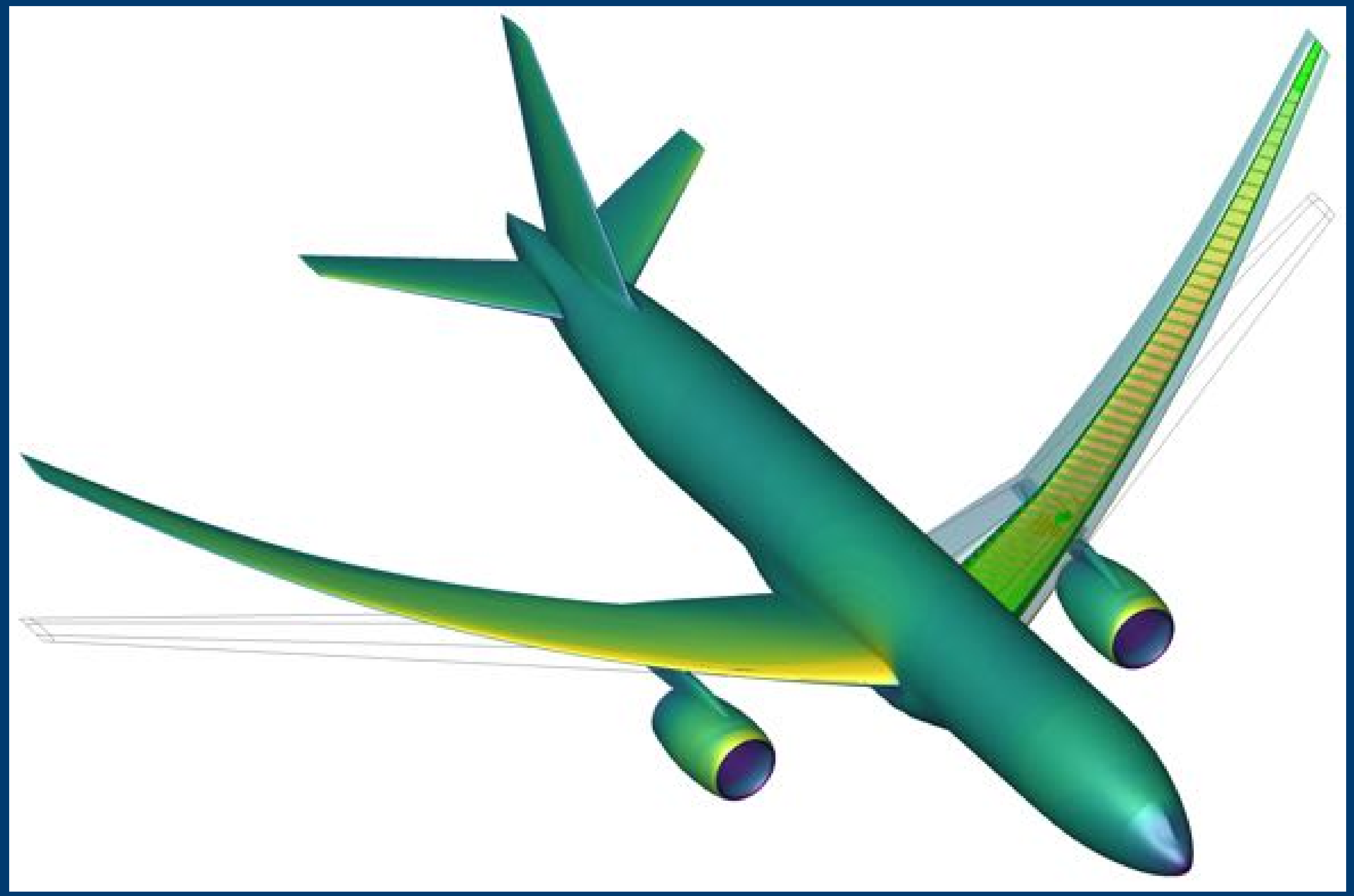
225

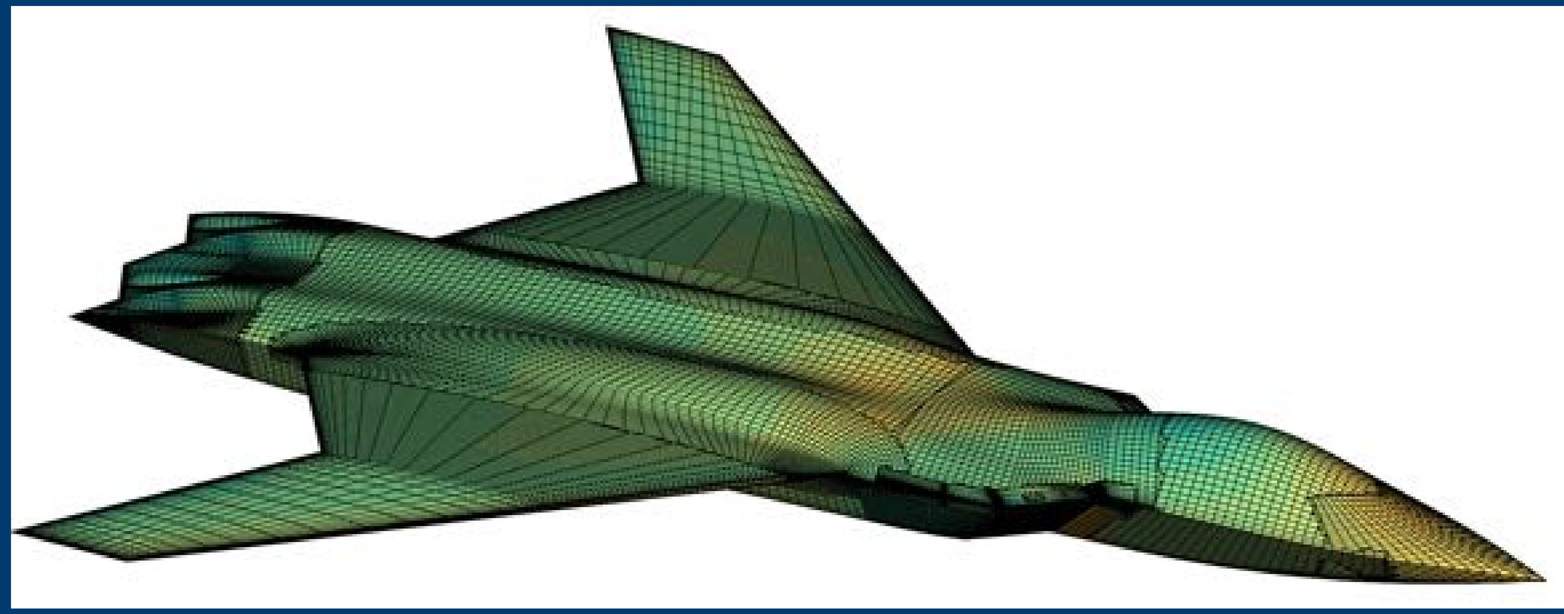


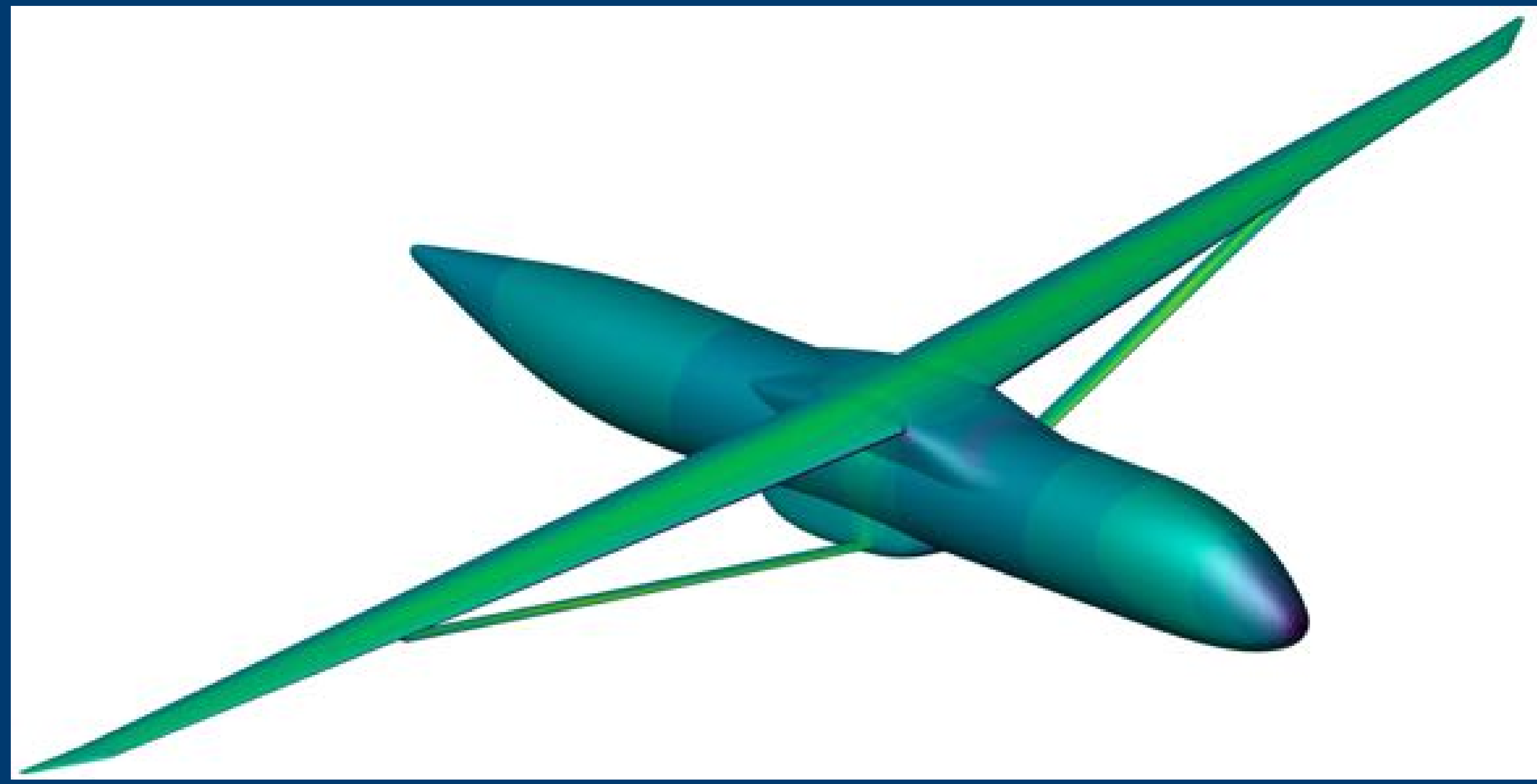


Martins and Ning. **Engineering Design Optimization.** Cambridge University Press, 2021.

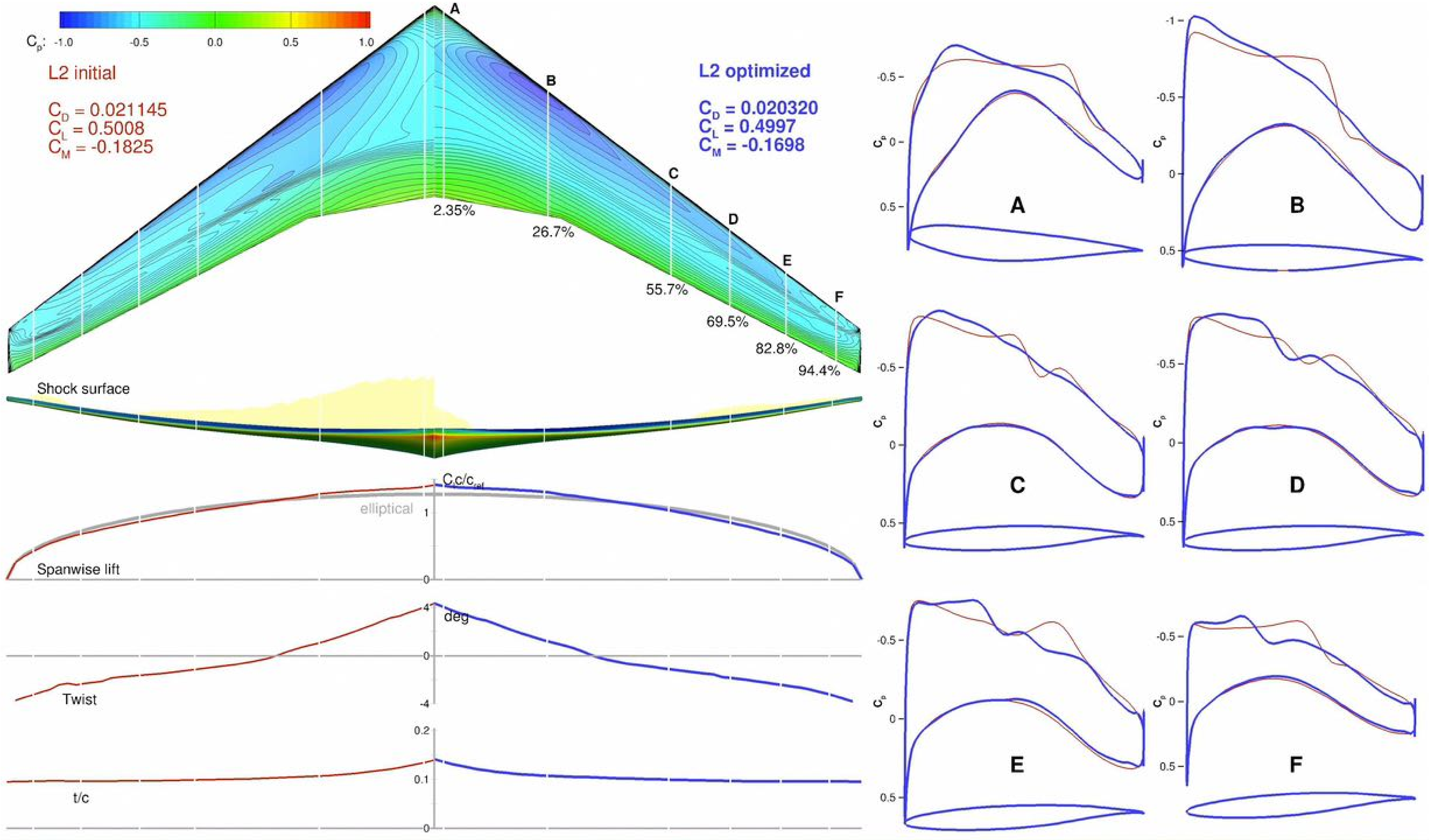
Applications





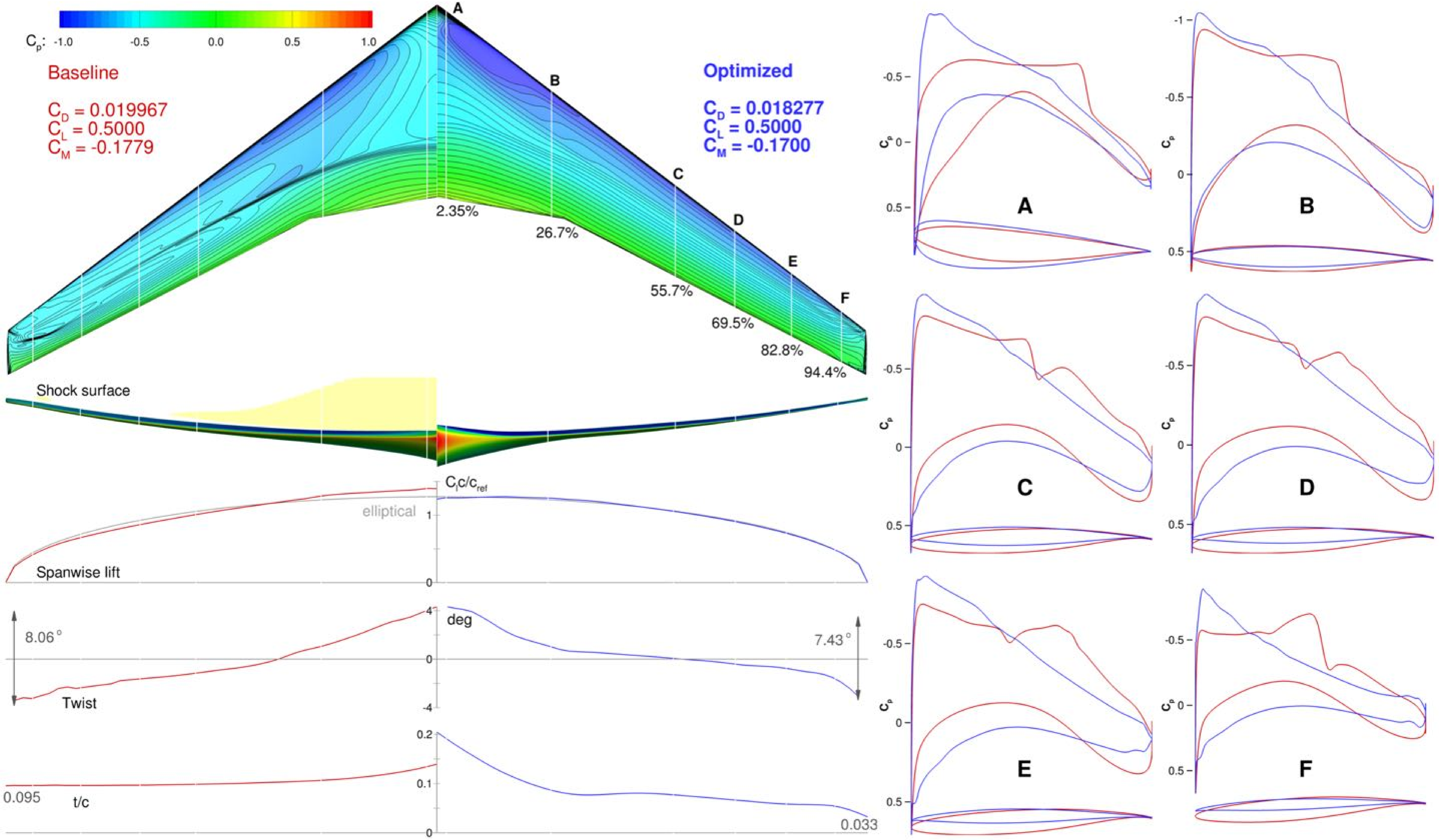


Wave drag is eliminated, and total drag is reduced by 8.5%

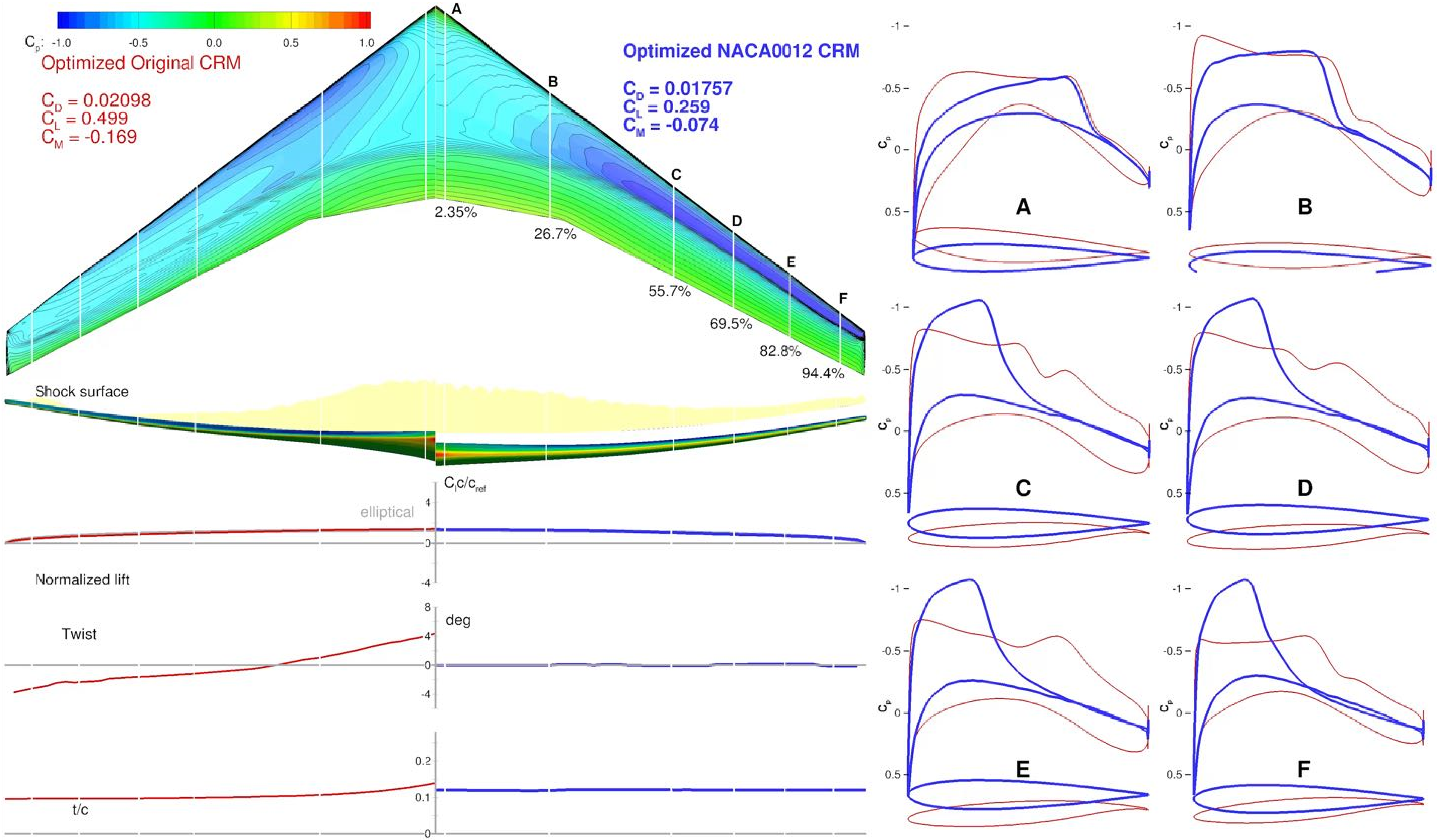


Lyu, Kenway, and Martins. **Aerodynamic shape optimization investigations of the Common Research Model wing benchmark**. *AIAA Journal*, 2015.

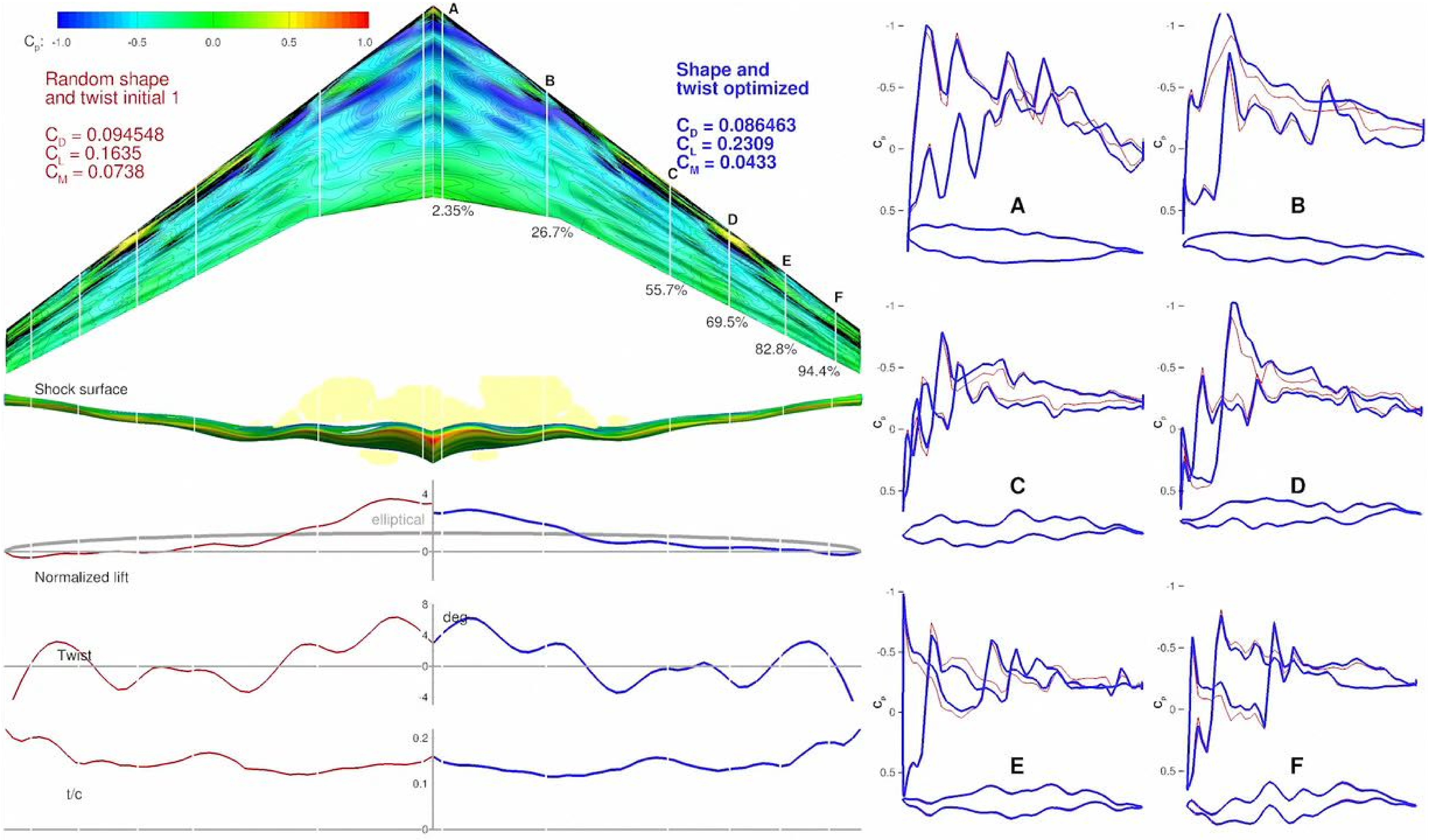
Optimization takes 6 hours using 128 cores



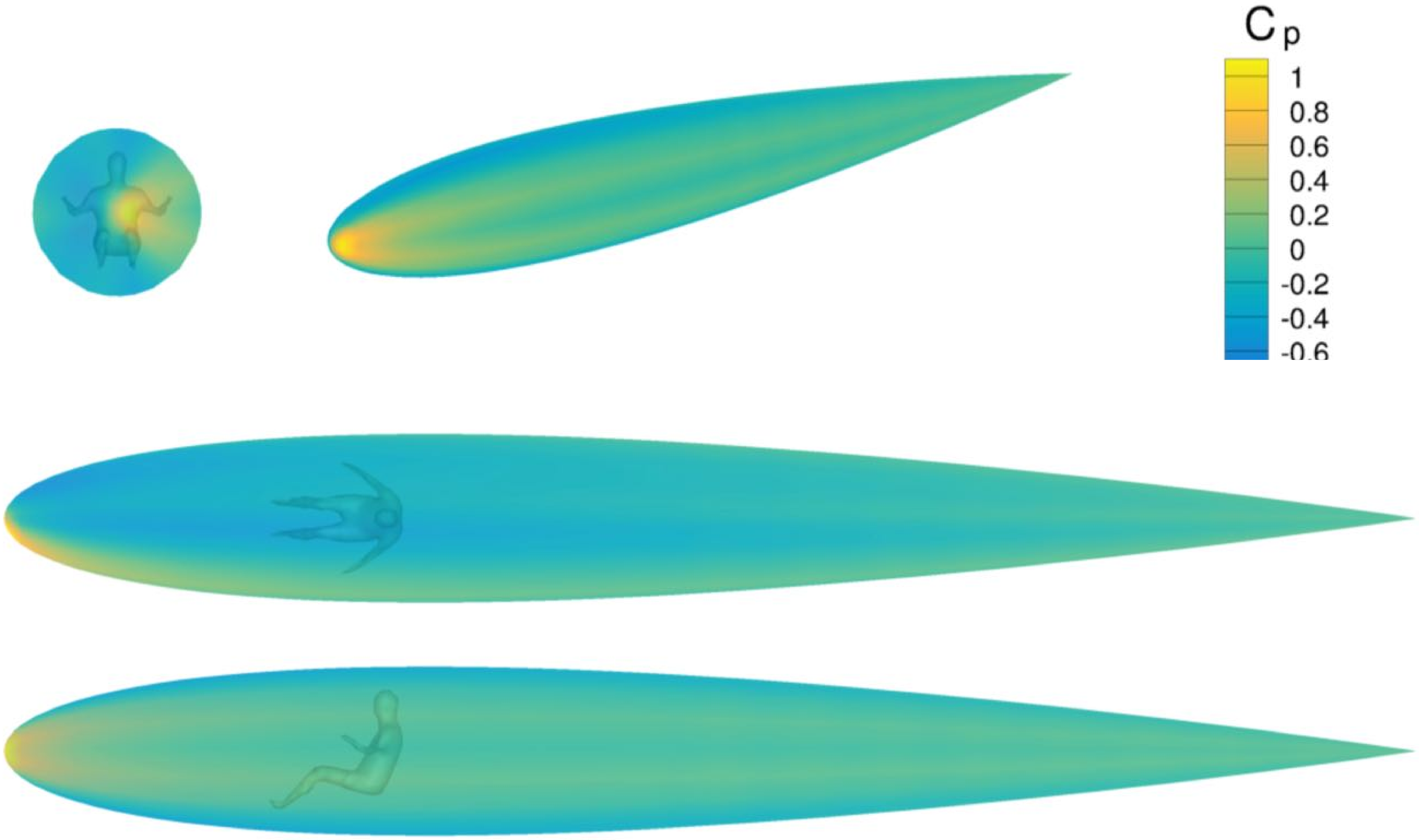
Two very different starting points:
CRM baseline vs. NACA0012 airfoil with no twist



Now, let's start with an even worse design!

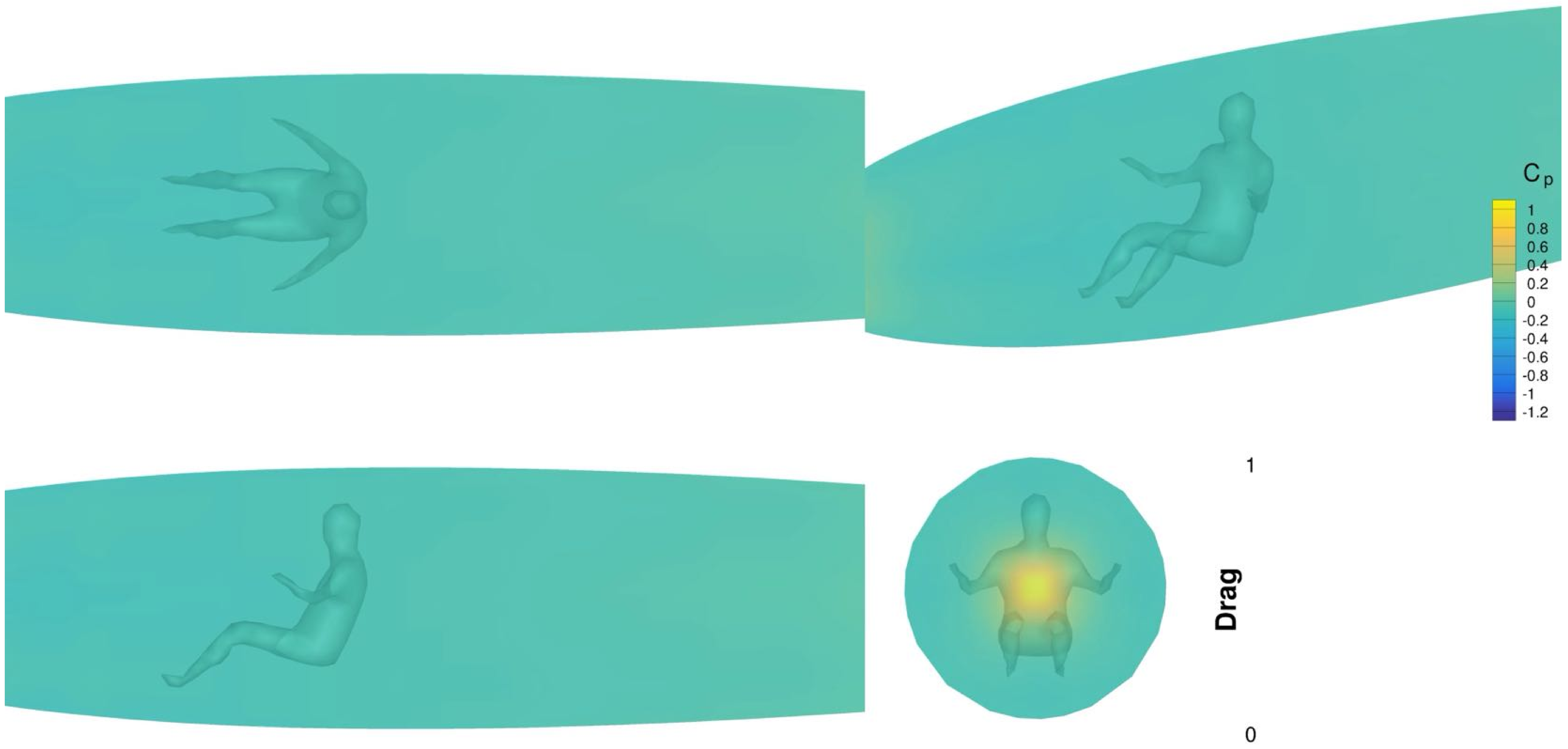


Can we get an airplane starting from a sphere?



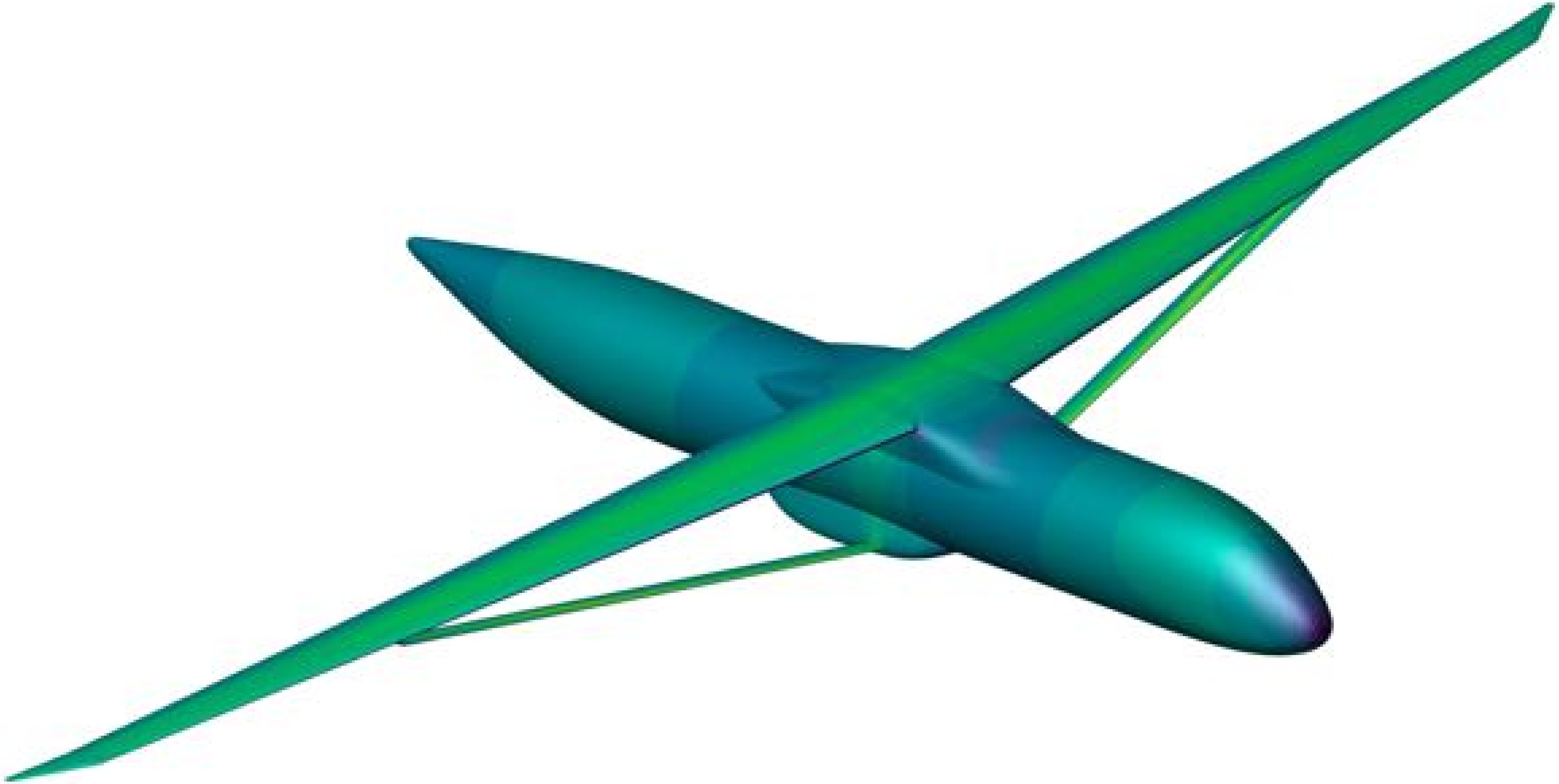
Brelje, Anibal, Yildirim, Mader, Martins. **Flexible formulation of spatial integration constraints in aerodynamic shape optimization**. *AIAA Journal, 2020.*

Can we get an airplane starting from a sphere?

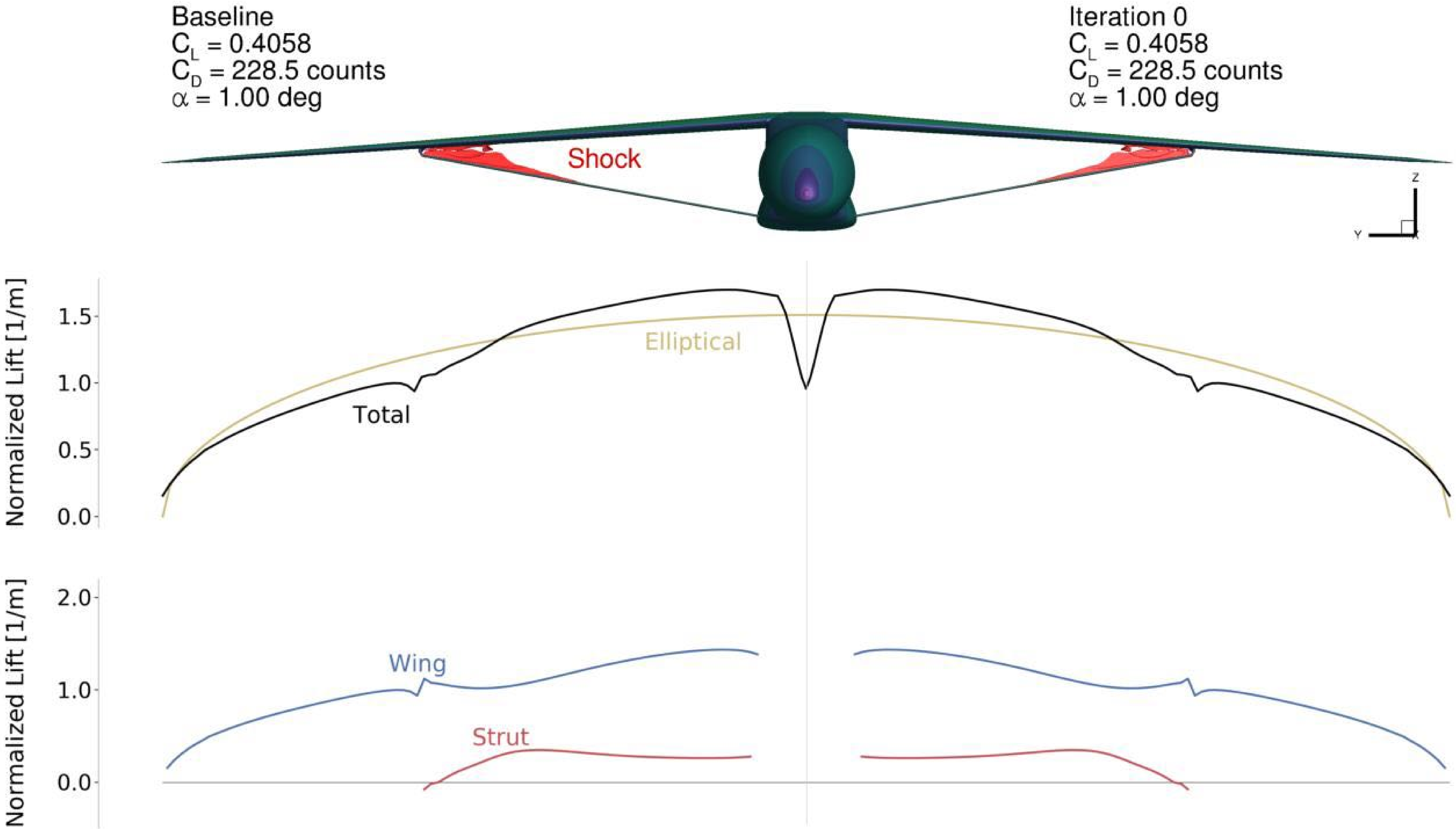


Brelje, Anibal, Yildirim, Mader, Martins. **Flexible formulation of spatial integration constraints in aerodynamic shape optimization**. *AIAA Journal, 2020*. (in press)

We were able to tackle the aerodynamic design optimization of the strut-braced wing thanks to the overset capability

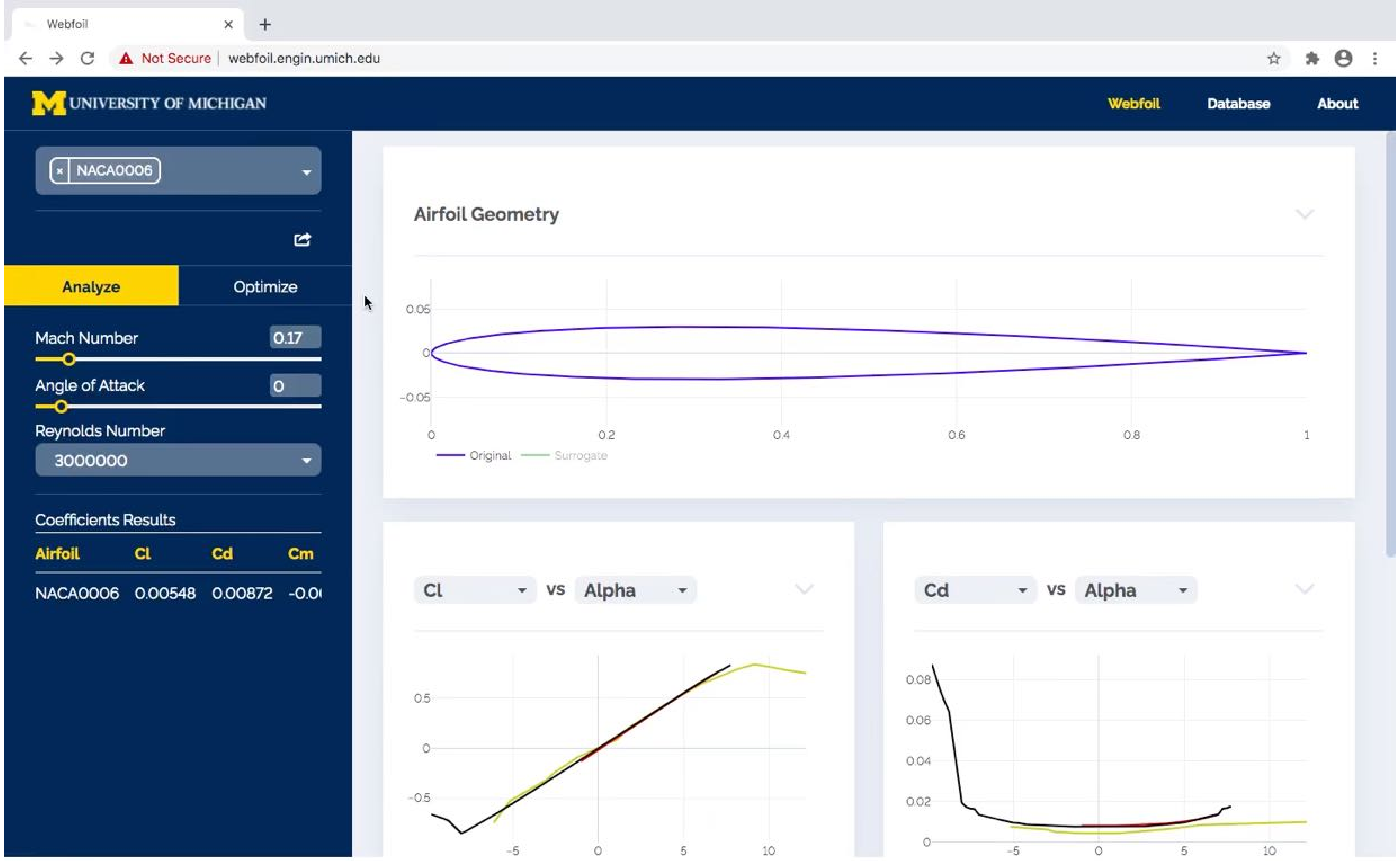


Final design reduced interference drag and resulted in a strut with negative lift



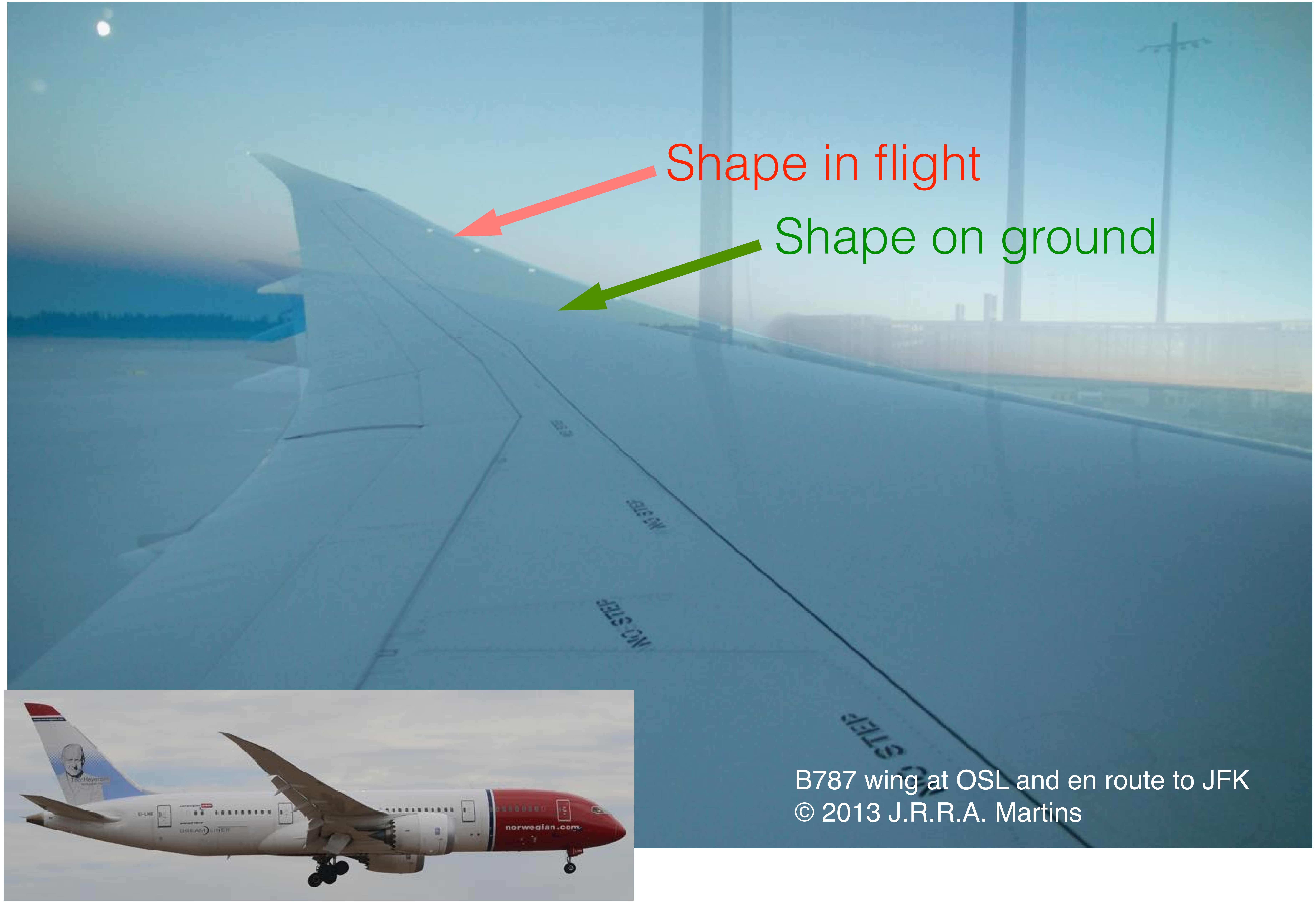
Secco and Martins. **RANS-based aerodynamic shape optimization of a strut-braced wing with overset meshes**. *Journal of Aircraft*, 2019

Webfoil is an airfoil database that optimizes airfoils in a few seconds based on machine learning



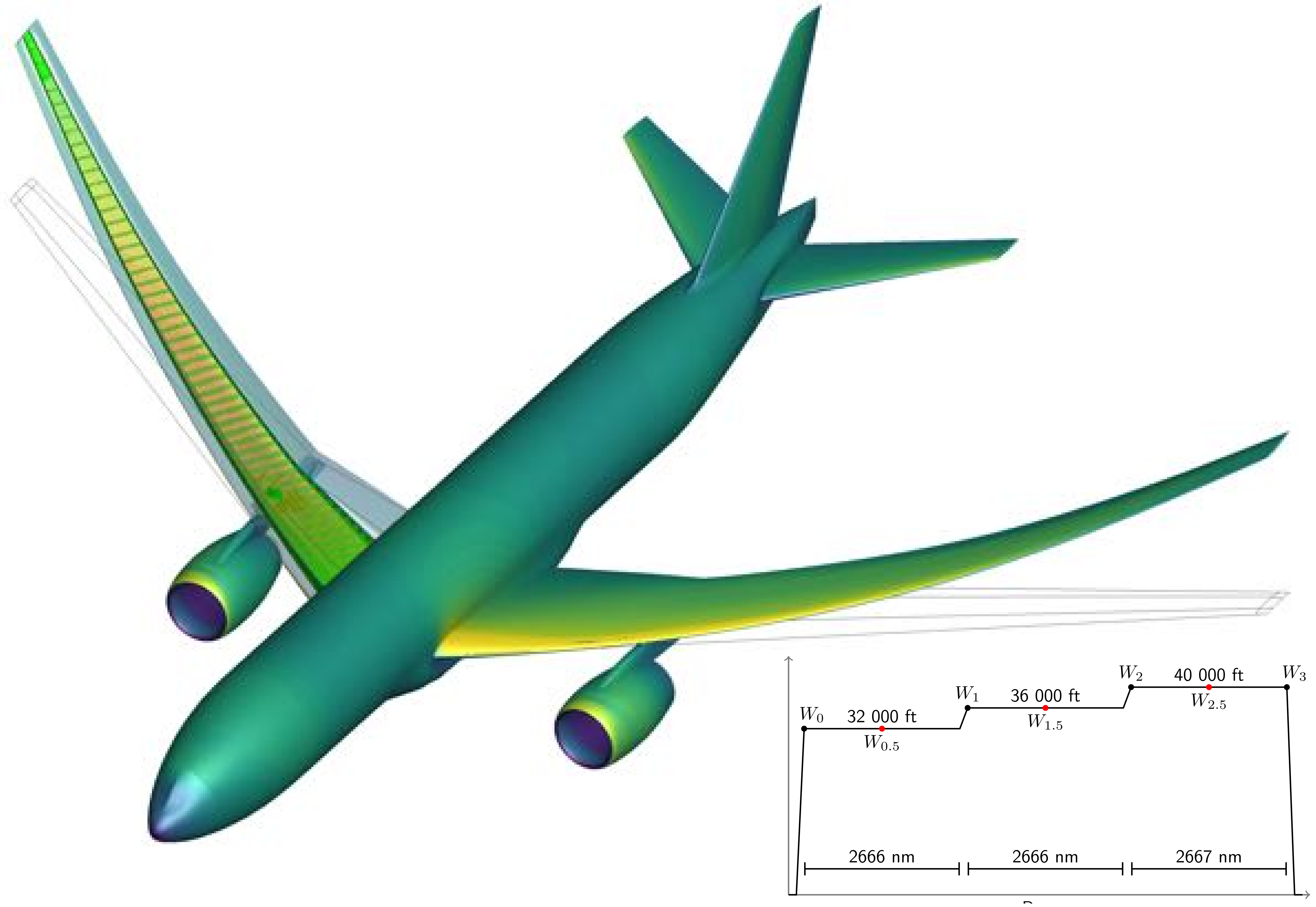
http://webfoil.engin.umich.edu

Wing design demands more than just aerodynamics

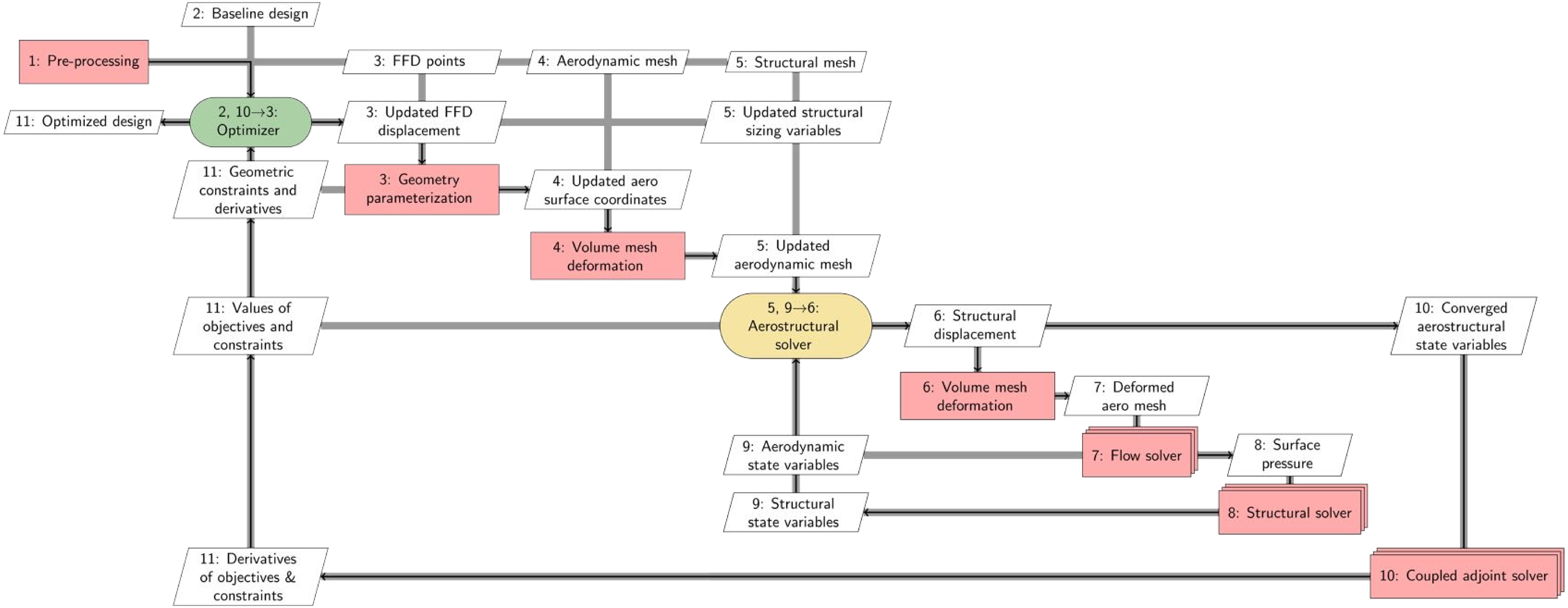


B787 wing at OSL and en route to JFK

Want to optimize both aerodynamic shape and structural sizing, with high-fidelity



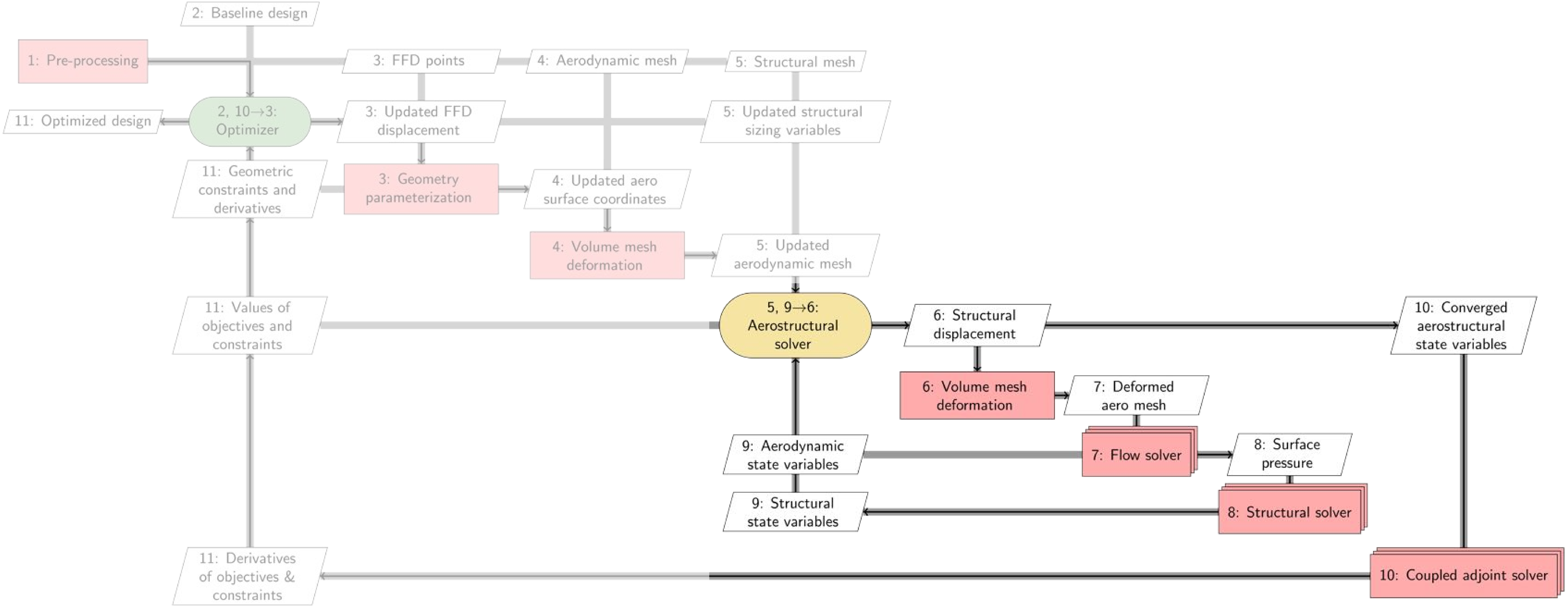
Coupled solution of aerodynamics and structures, and the corresponding coupled adjoint



Kenway, Kennedy, and Martins. **Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and derivative computations**. *AIAA Journal*, 2014

Kennedy and Martins. **A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures**. *Finite Elements in Analysis and Design*, 2014

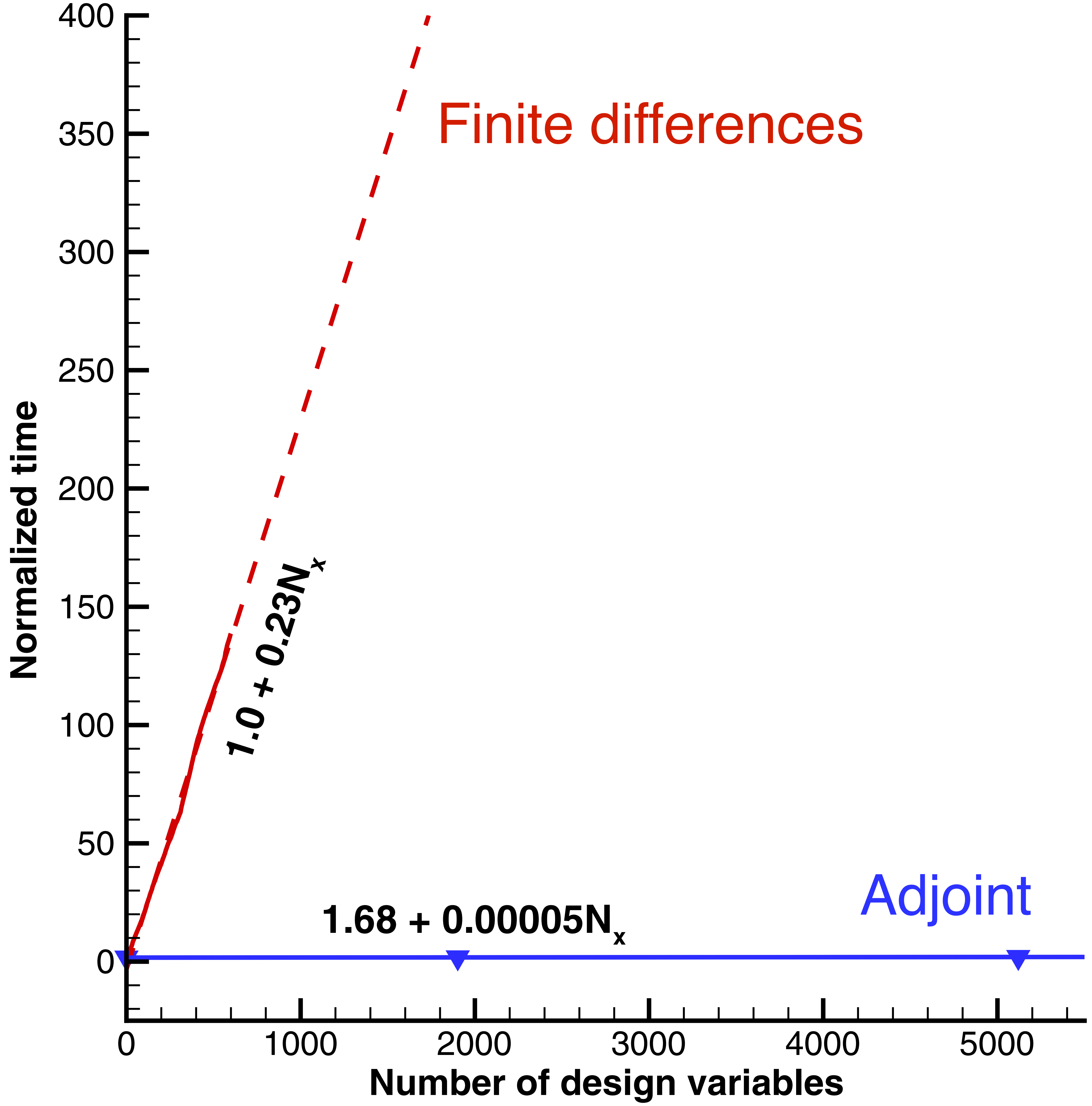
Coupled solution of aerodynamics and structures, and the corresponding coupled adjoint



Kenway, Kennedy, and Martins. **Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and derivative computations**. *AIAA Journal*, 2014

Kennedy and Martins. **A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures**. *Finite Elements in Analysis and Design*, 2014

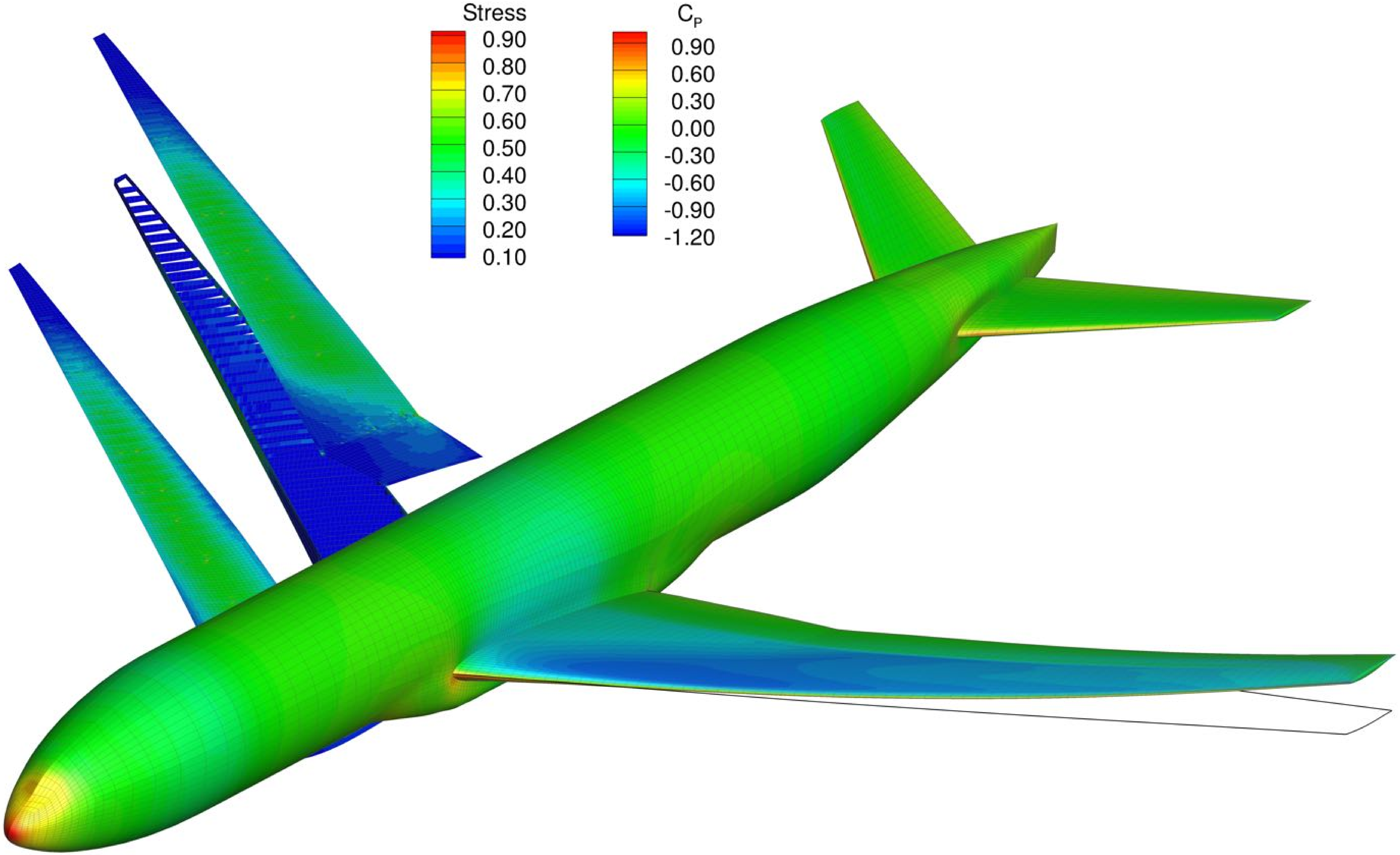
Coupled adjoint method efficiently computes gradients with respect to thousands of variables



Kenway, Kennedy, and Martins. **Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and derivative computations**. *AIAA Journal*, 2014

Let's do aerostructural optimization!

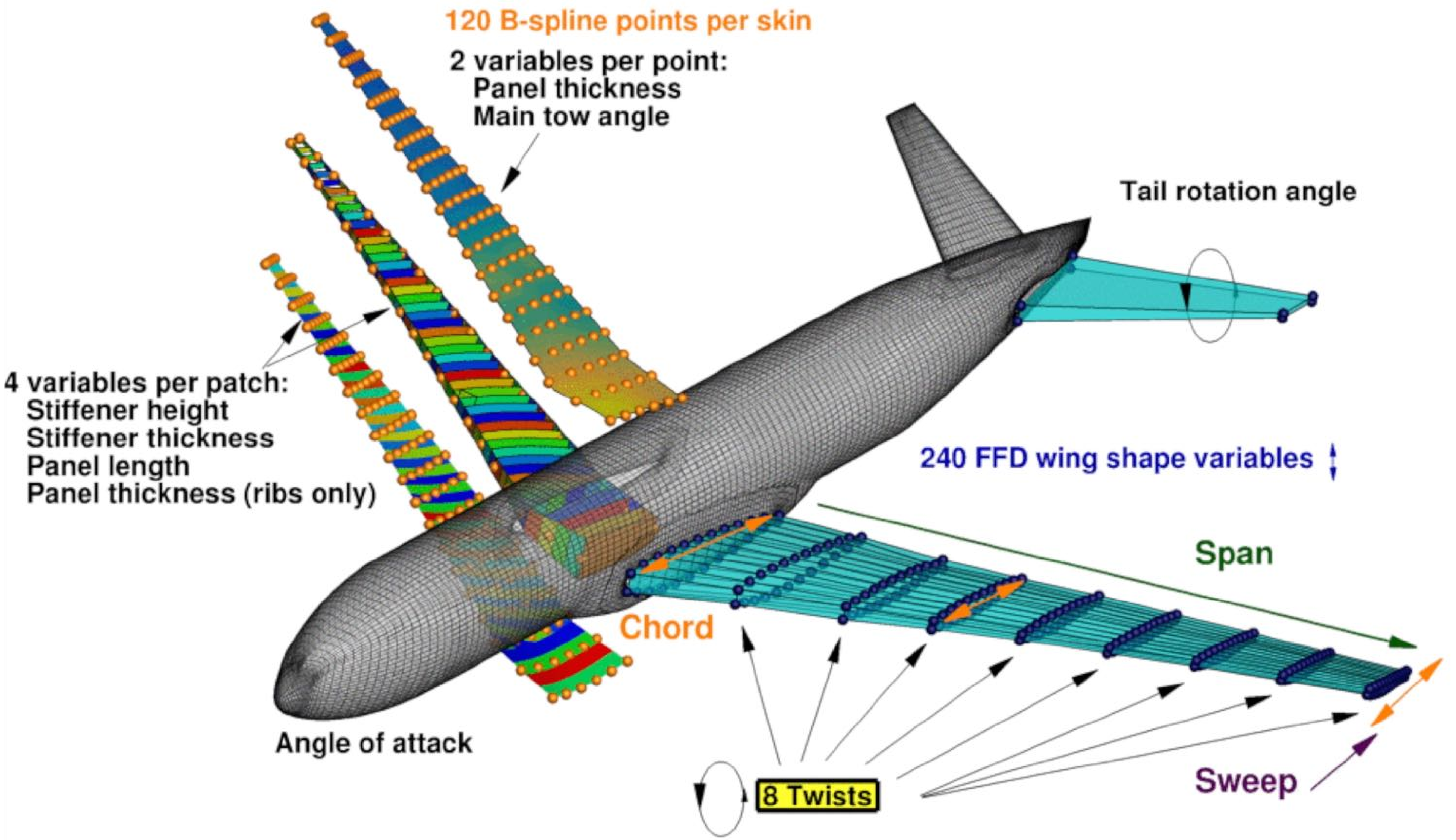
NASA-Michigan undeformed Common Research Model (uCRM)

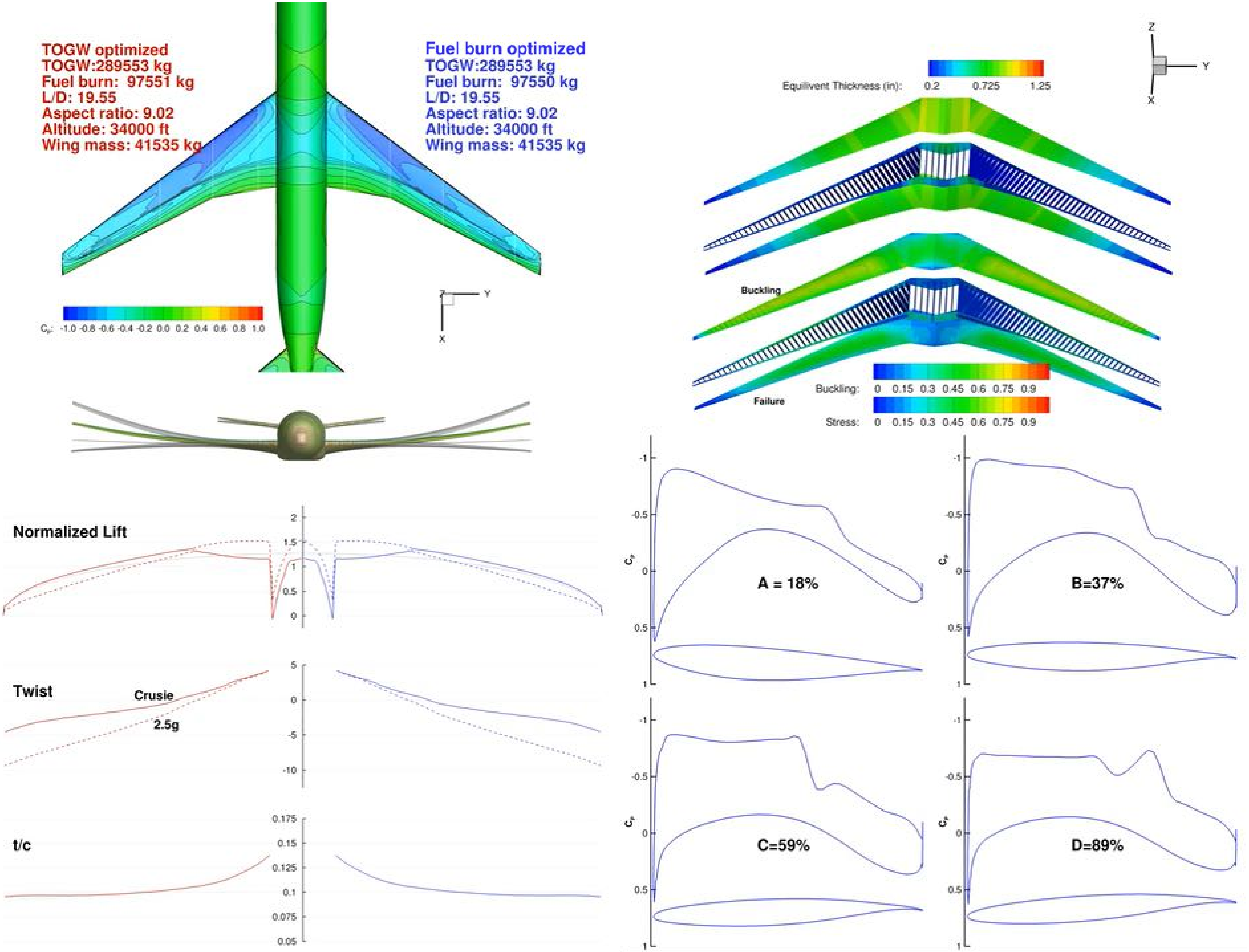


Kenway,Martins. **High-fidelity aerostructural optimization considering buffet onset**, *AIAA 2015-2790.*

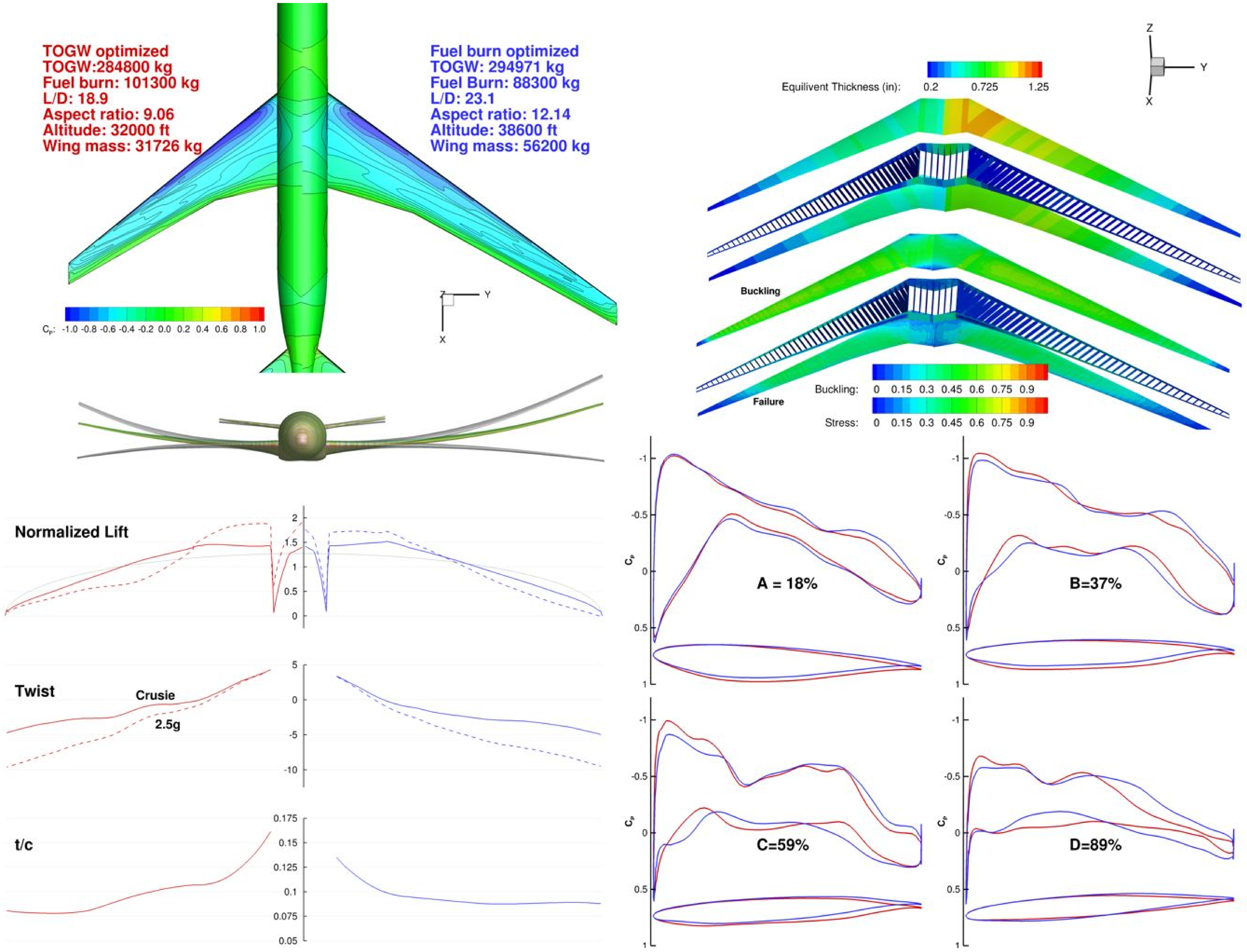
Brooks, Kenway,Martins. **Benchmark aerostructural models for the study of transonic aircraft wings**, *AIAA Journal,* 2018.

Optimize 973 “aerodynamic” and structural sizing design variables



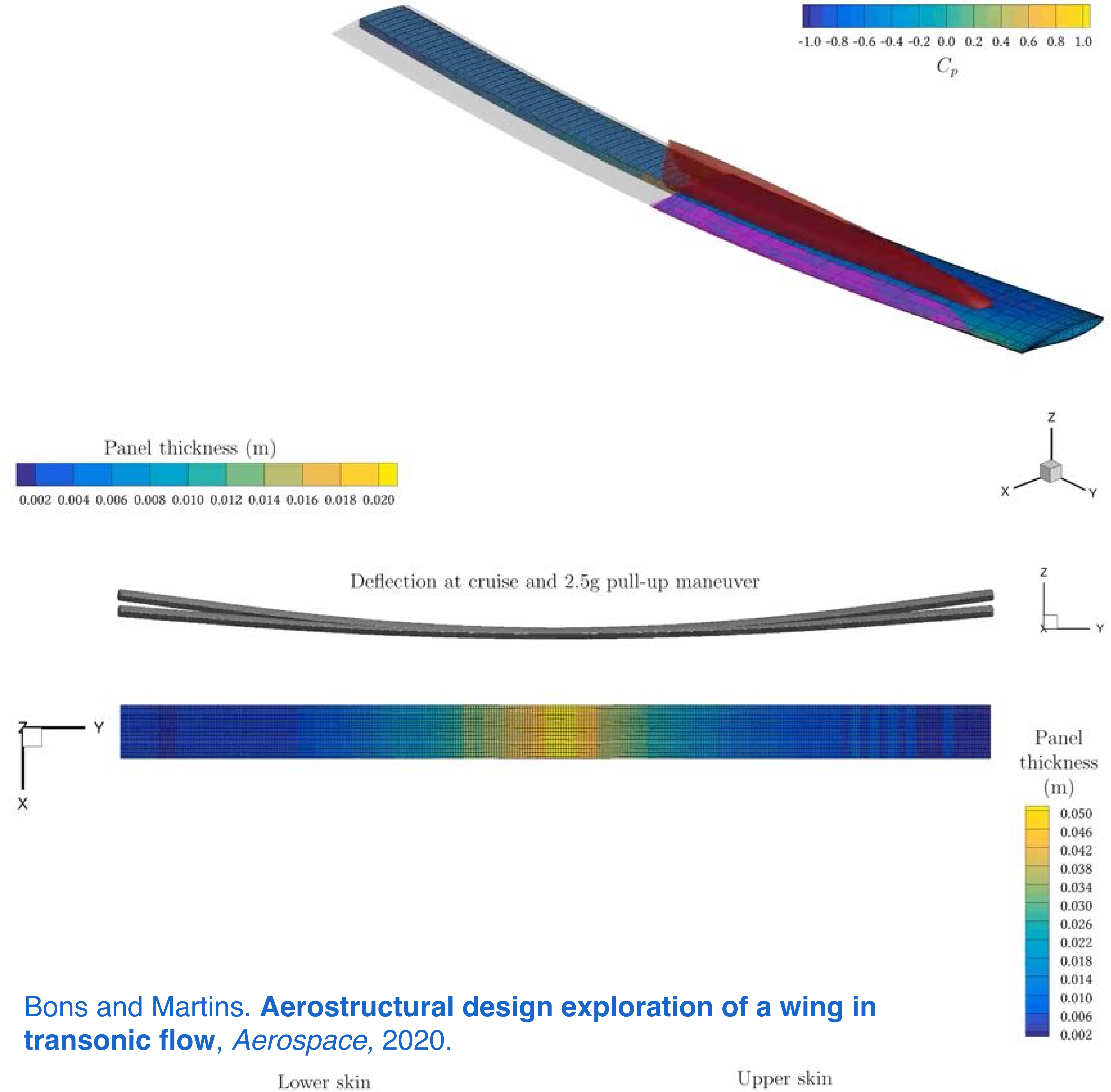


TOGW optimized
TOGW:289553 kg
Fuel burn: 97551 kg
L/D: 19.55
Aspect ratio: 9.02
Altitude: 34000 ft
Wing mass: 41535 kg
Fuel burn optimized
TOGW:289553 kg
Fuel burn: 97550 kg
L/D: 19.55
Aspect ratio: 9.02
Altitude: 34000 ft
Wing mass: 41535 kg
C_p: -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
Equilivent Thickness (in): 0.2 0.725 1.25
Buckling
Failure
Buckling: 0 0.15 0.3 0.45 0.6 0.75 0.9
Stress: 0 0.15 0.3 0.45 0.6 0.75 0.9
Normalized Lift
Twist
Crusie
2.5g
t/c
A = 18%
B=37%
C=59%
D=89%



Kenway and Martins. **High-fidelity aerostructural optimization considering buffet onset**, *AIAA 2015-2790*

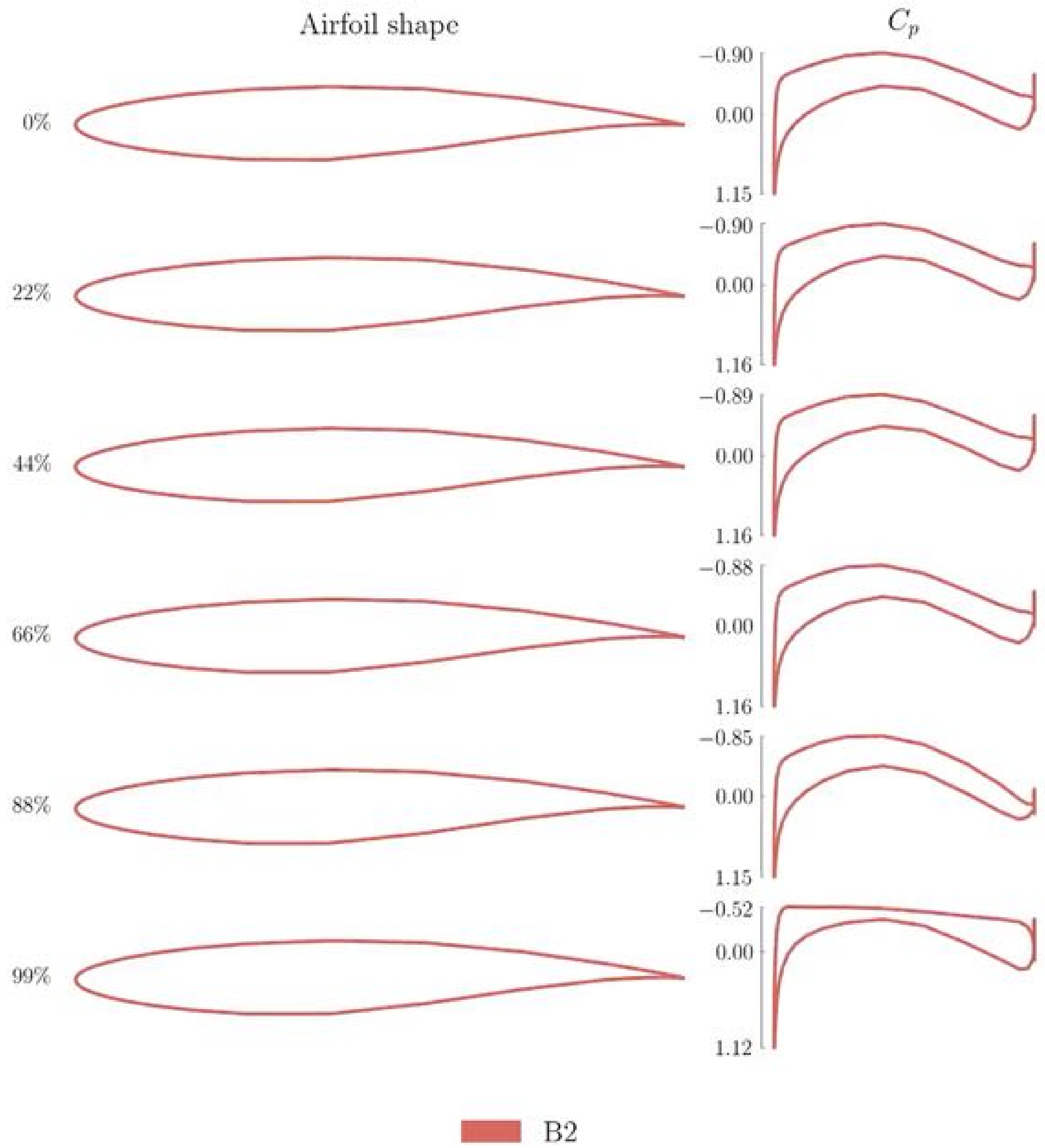
From a plank to a transonic wing

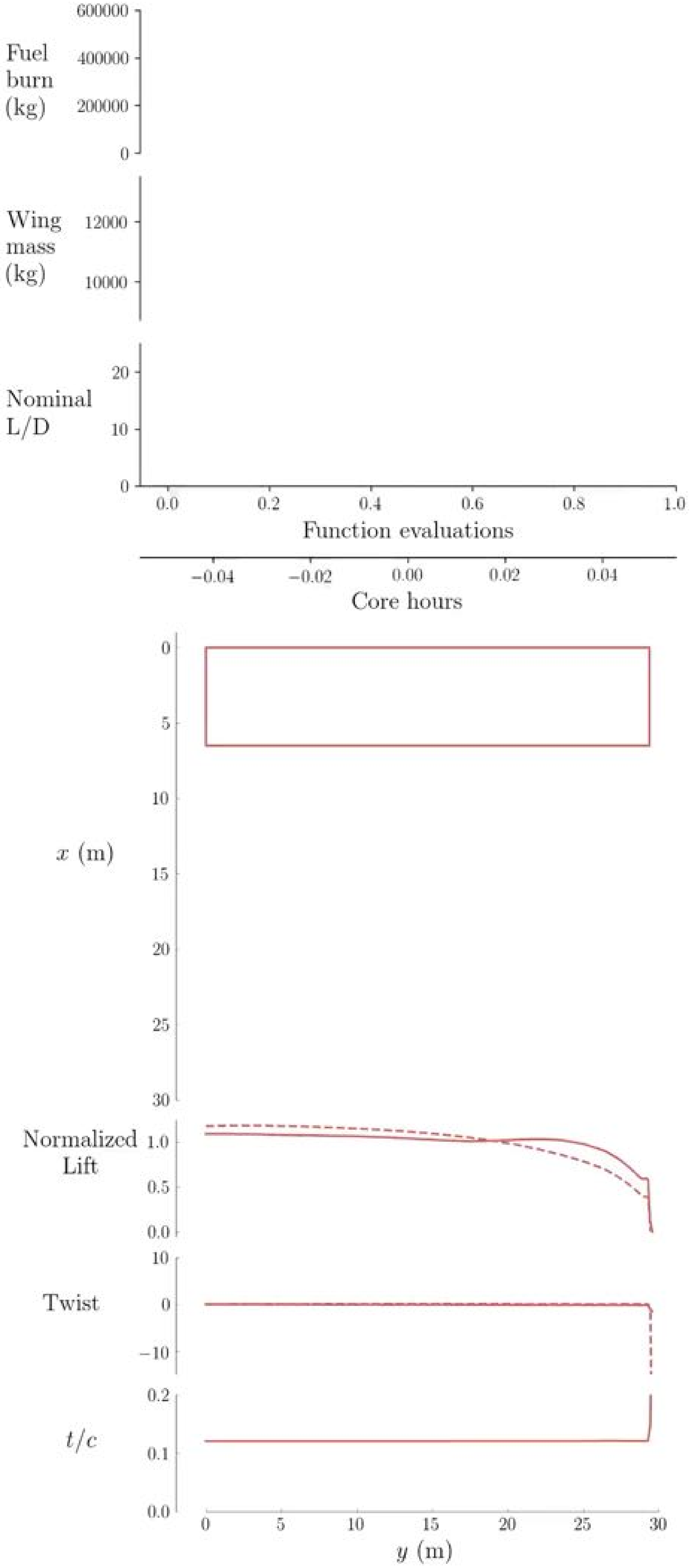


Fuel burn minimization
Mach 0.85
37,000 ft

Off-design
2.5g maneuver
1.3g buffet

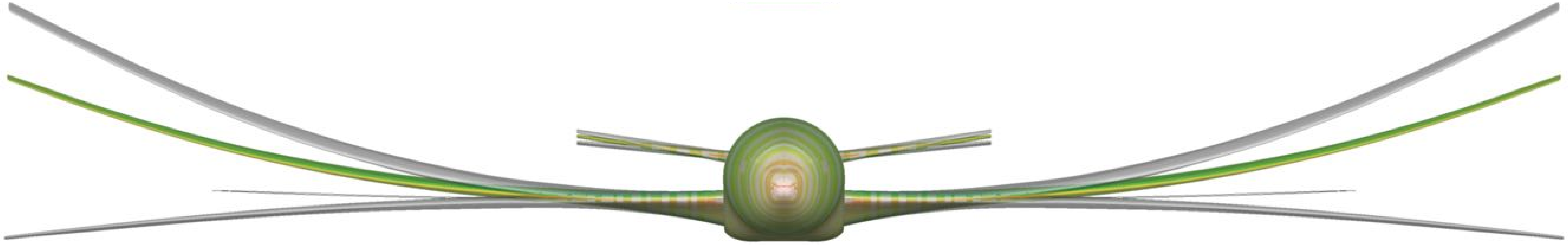
Grid sizes
Coarse: 52k cells
Medium: 152k cells
Fine: 417k cells
Very fine: 1.2M cells

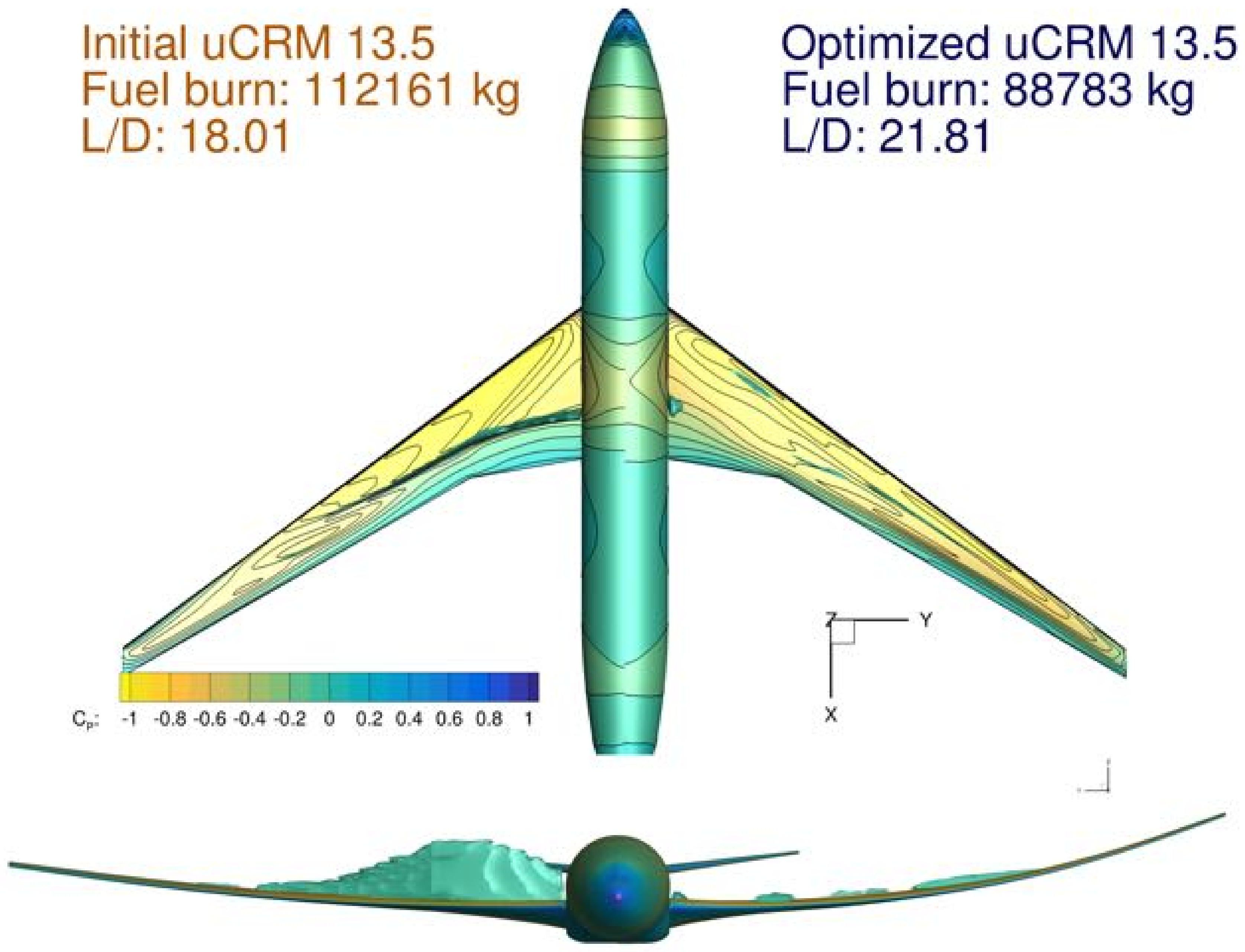


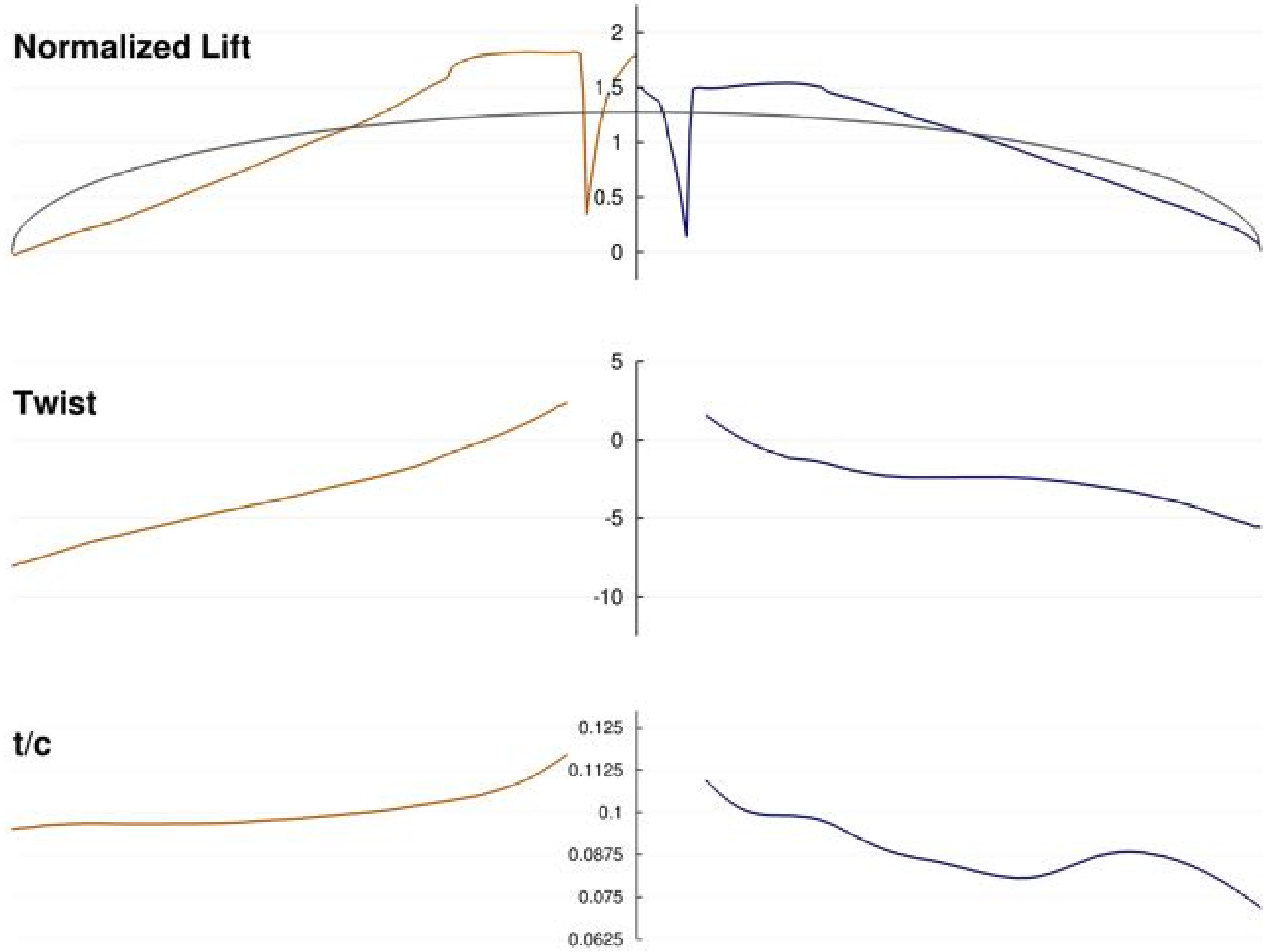


Bons and Martins. **Aerostructural design exploration of a wing in transonic flow**, *Aerospace,* 2020.

Developed uCRM-13.5, a high aspect ratio flexible version of the CRM

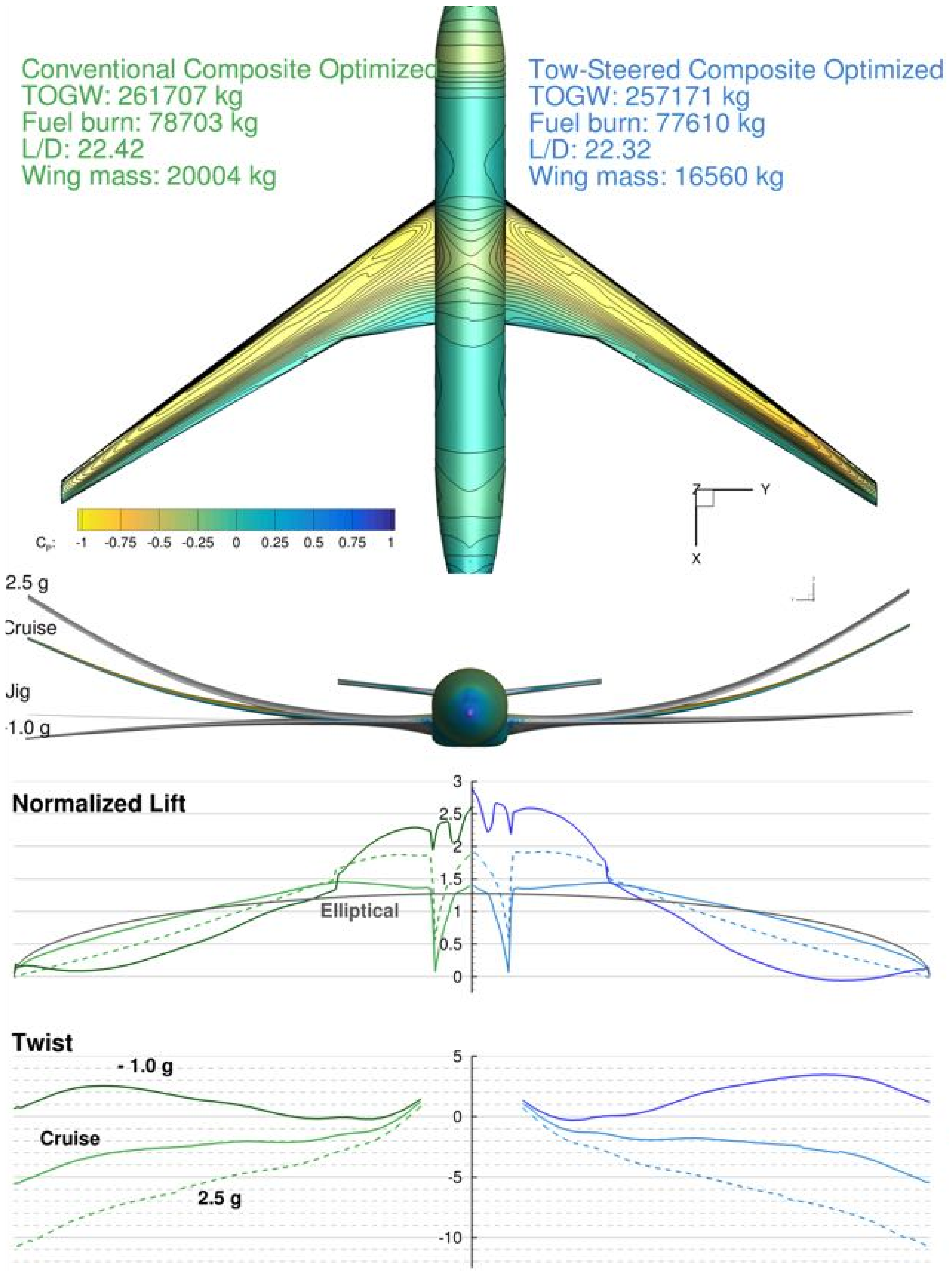


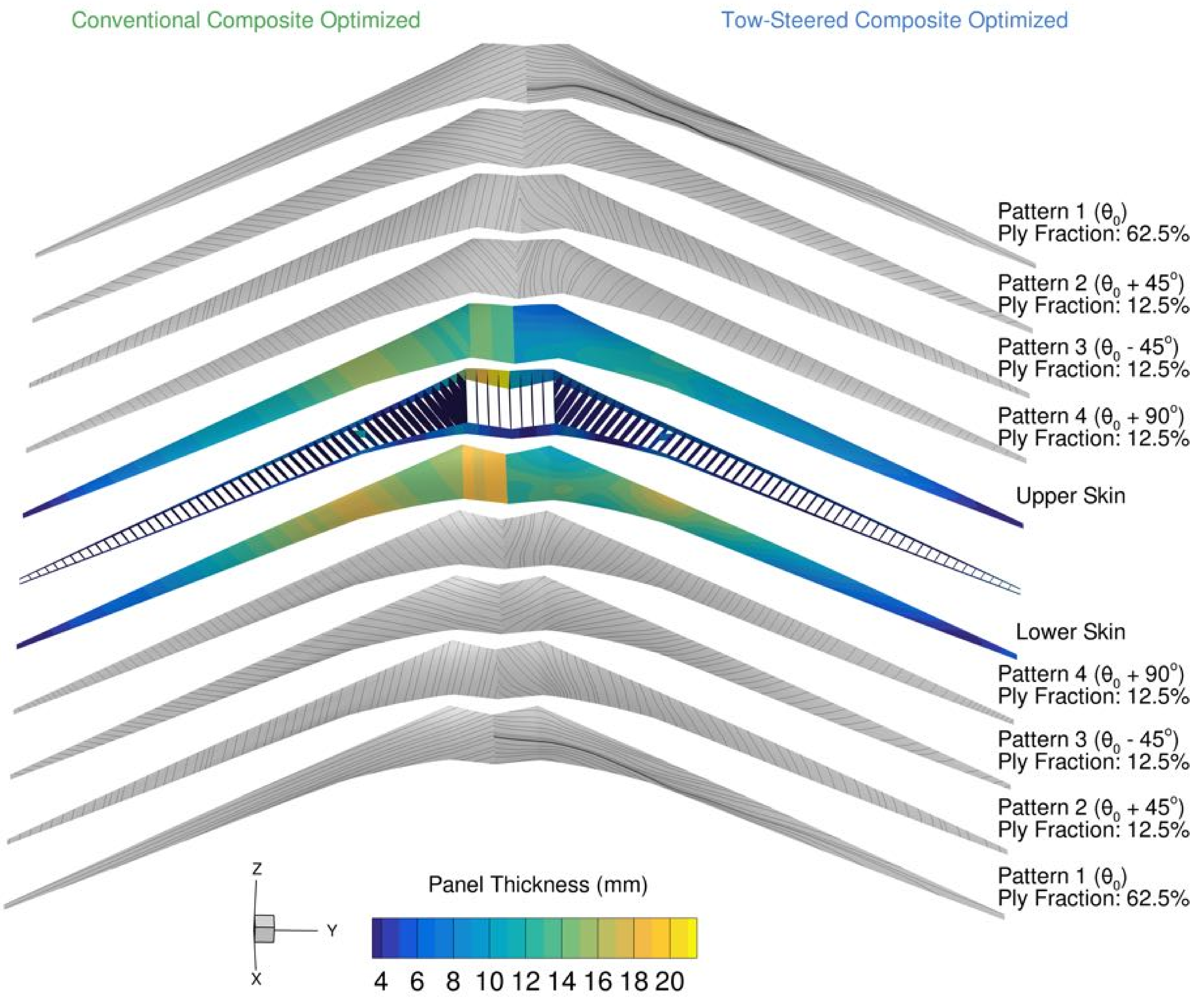




Brooks, Kenway, and Martins. **Benchmark aerostructural models for the study of transonic aircraft wings**. *AIAA Journal*, 2018

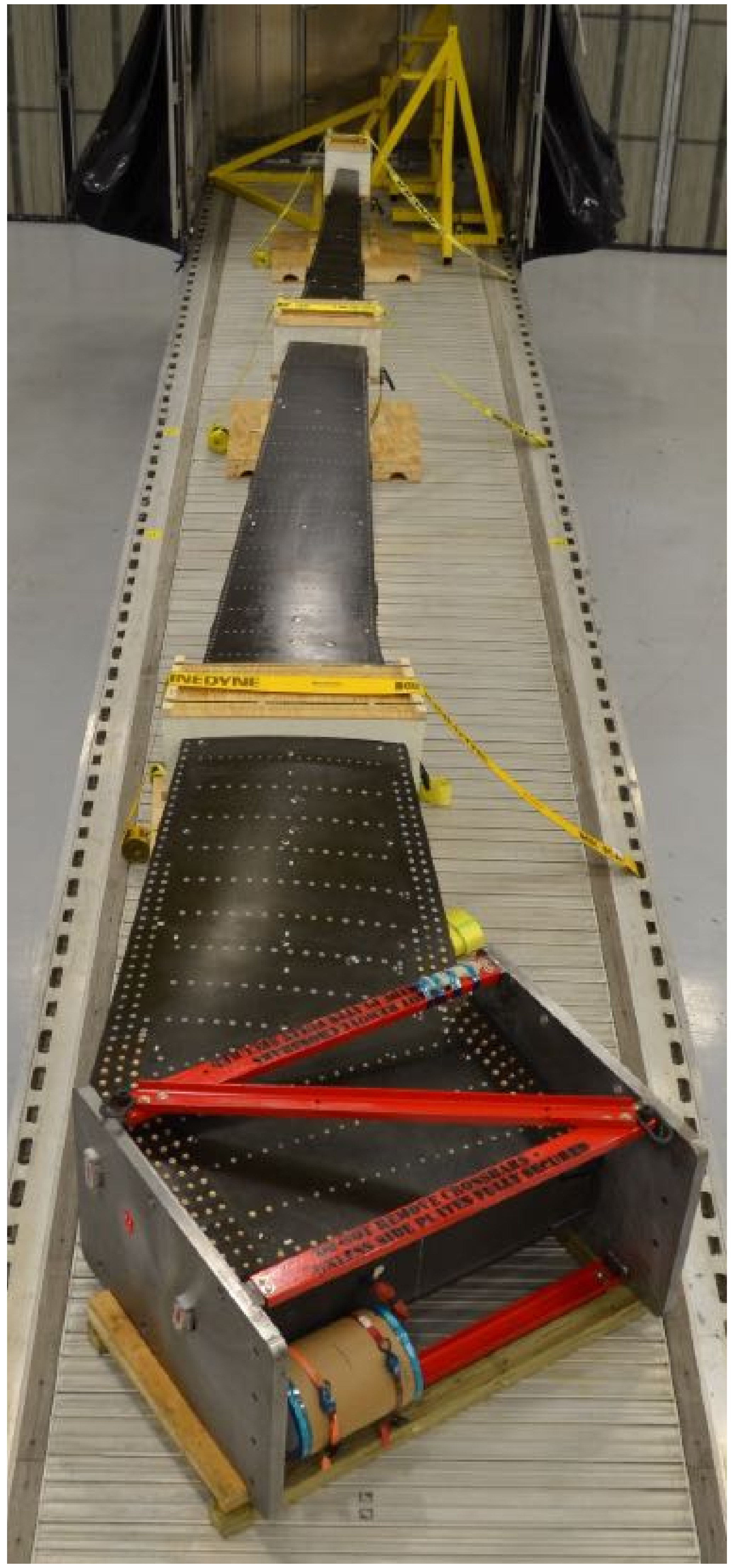
Tow-steered composite high AR wing





Brooks, Martins, and Kennedy. **High-fidelity aerostructural optimization of tow-steered composite wings**. *Journal of Fluids and Structures*, 2019

Our design was built using an AFP machine



Aurora Builds Tow-Steered Carbon Wing for NASA, *Aviation Week & Space Technology*, 2017

The wingbox was tested by NASA

The coupled adjoint method was generalized in as modular analysis and unified derivatives (MAUD)

Unified derivatives equation (UDE)

$$\frac{\partial r}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}r} = I = \frac{\partial r}{\partial u}^{\intercal}\frac{\mathrm{d}u}{\mathrm{d}r}^{\intercal}$$

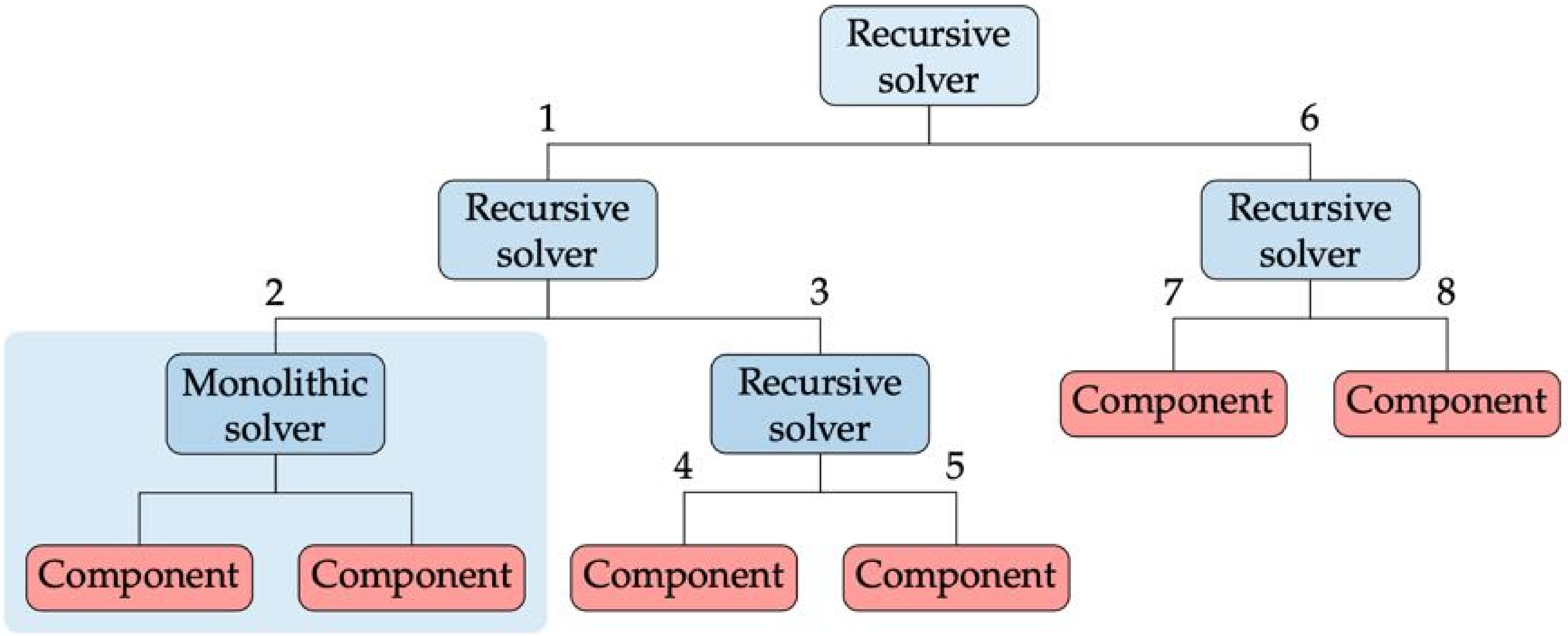
- Chain rule
- Implicit analytic: direct and adjoint
- AD: forward and reverse
- Coupled systems with mixed explicit and implicit components: direct and adjoint

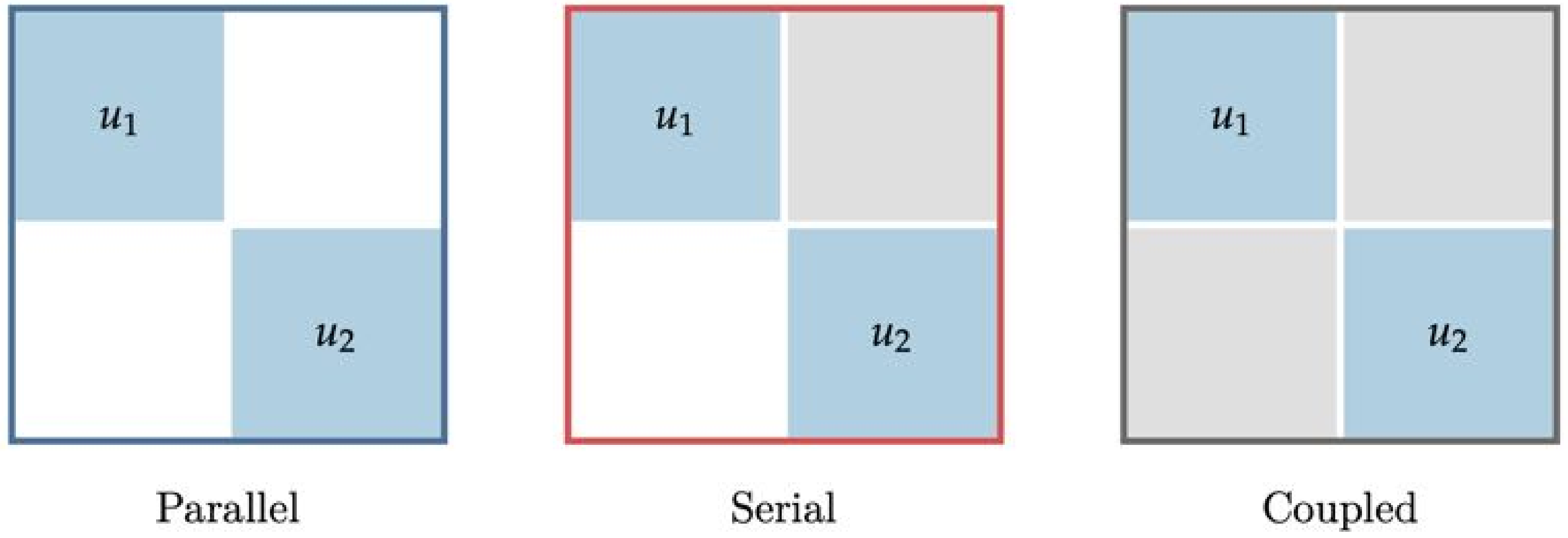
Martins and Hwang. **Review and unification of methods for computing derivatives of multidisciplinary computational models**. *AIAA Journal*, 2013.

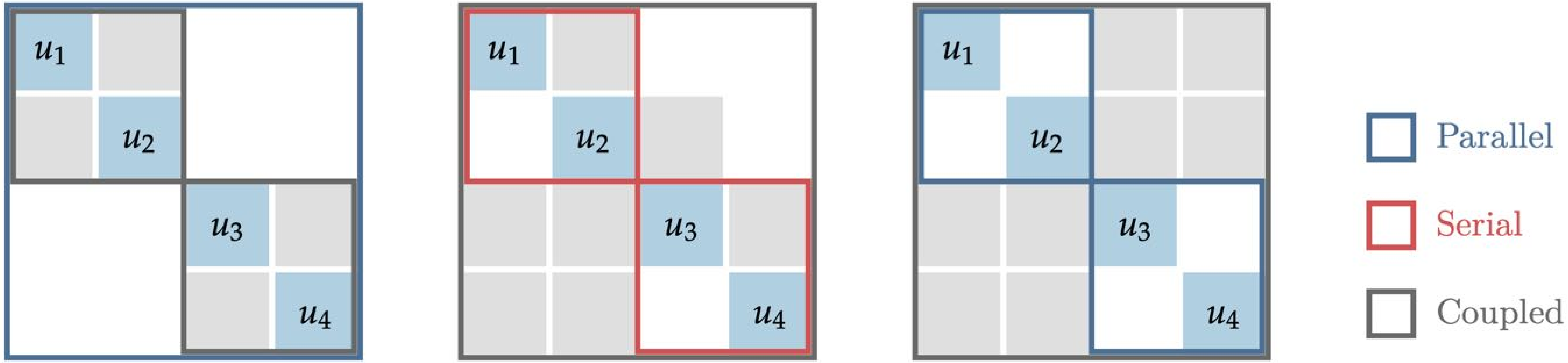
Hwang and Martins. **A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives**. *ACM Transactions on Mathematical Software*, 2018

Martins and Ning. **Engineering Design Optimization.** Cambridge University Press, 2021.

MAUD includes hierarchical solvers and coupled derivatives for complex systems



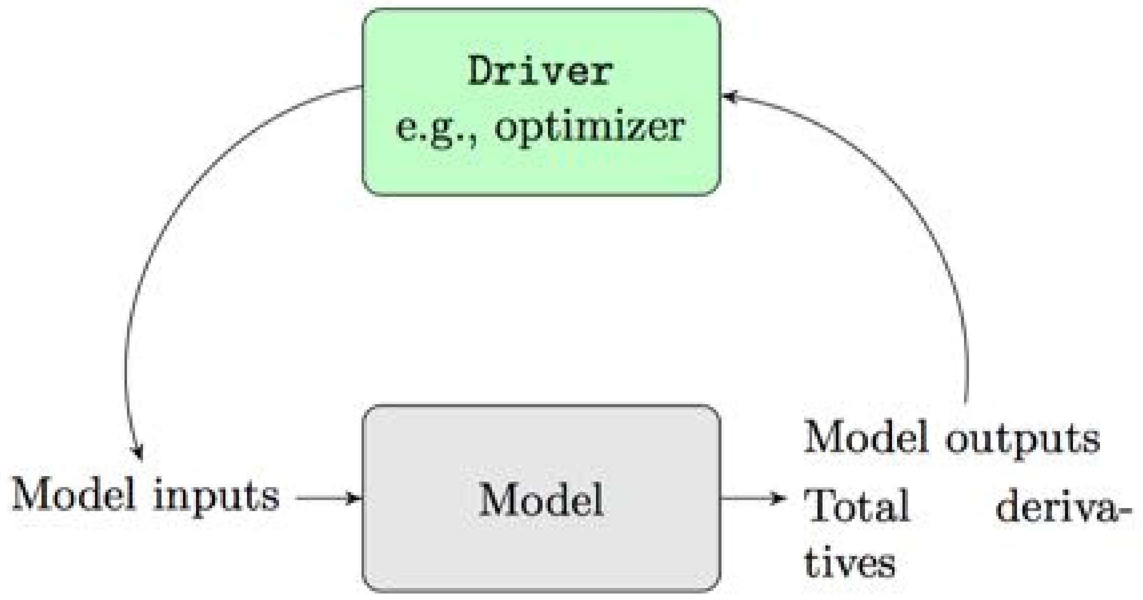


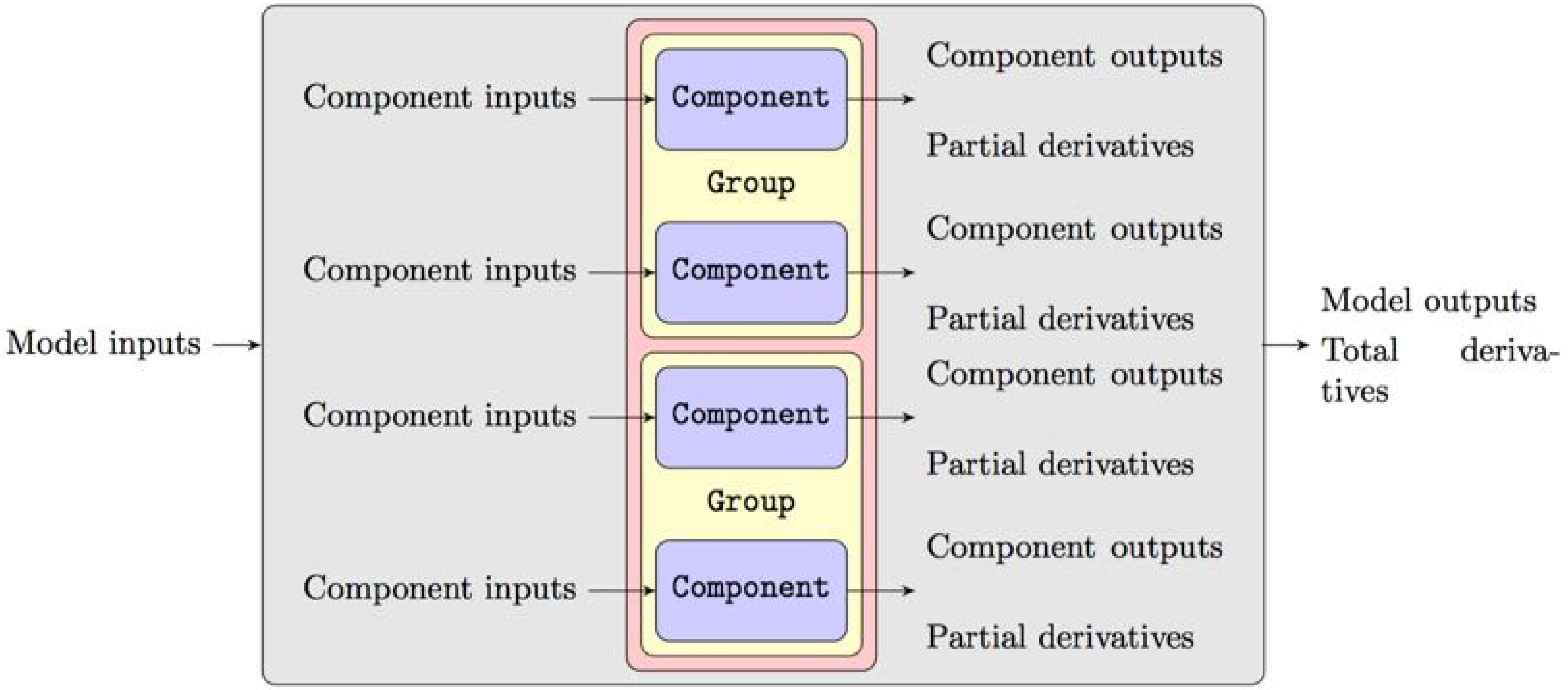


Hwang and Martins. **A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives**. *ACM Transactions on Mathematical Software*, 2018

Martins and Ning. **Engineering Design Optimization.** Cambridge University Press, 2021.

MAUD was implemented in openMDAO

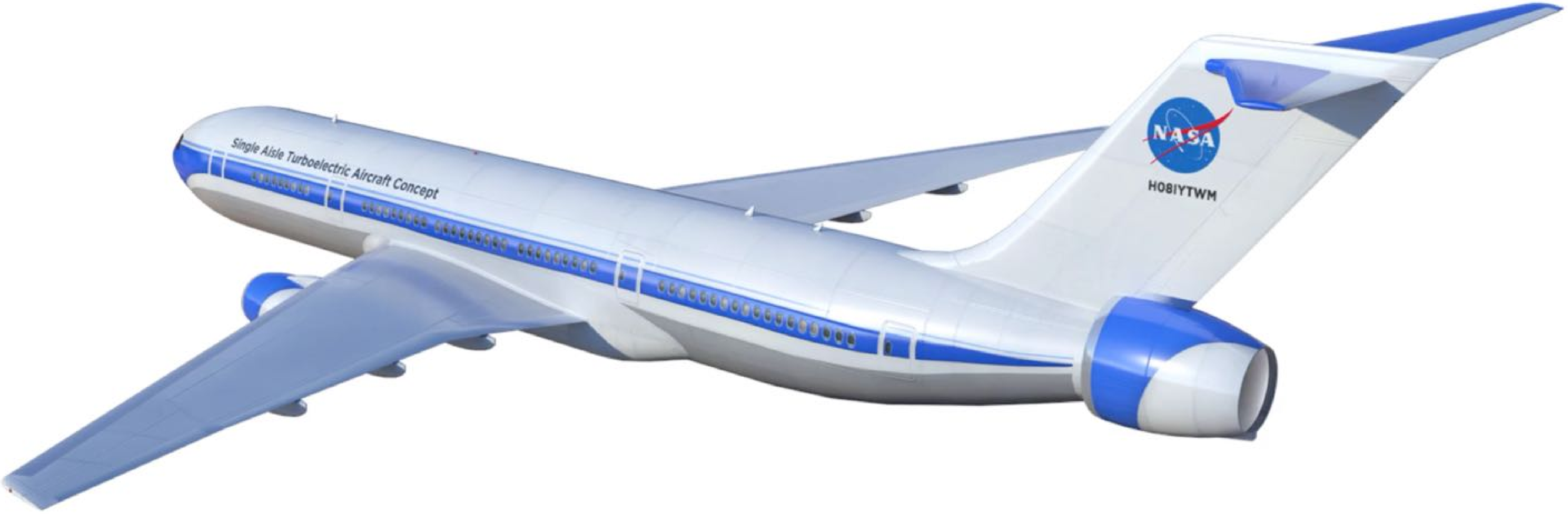




- Developed at NASA Glenn
- Python-based
- Open-source framework
- Facilitates the coupling multiple models and optimization
- Efficient coupled solution via Newton-type methods
- Efficient coupled adjoint derivative computation

Gray, Hwang, Martins, Moore, and Naylor. **OpenMDAO: An open- source framework for multidisciplinary design, analysis, and optimization**. *Structural and Multidisciplinary Optimization*, 2019

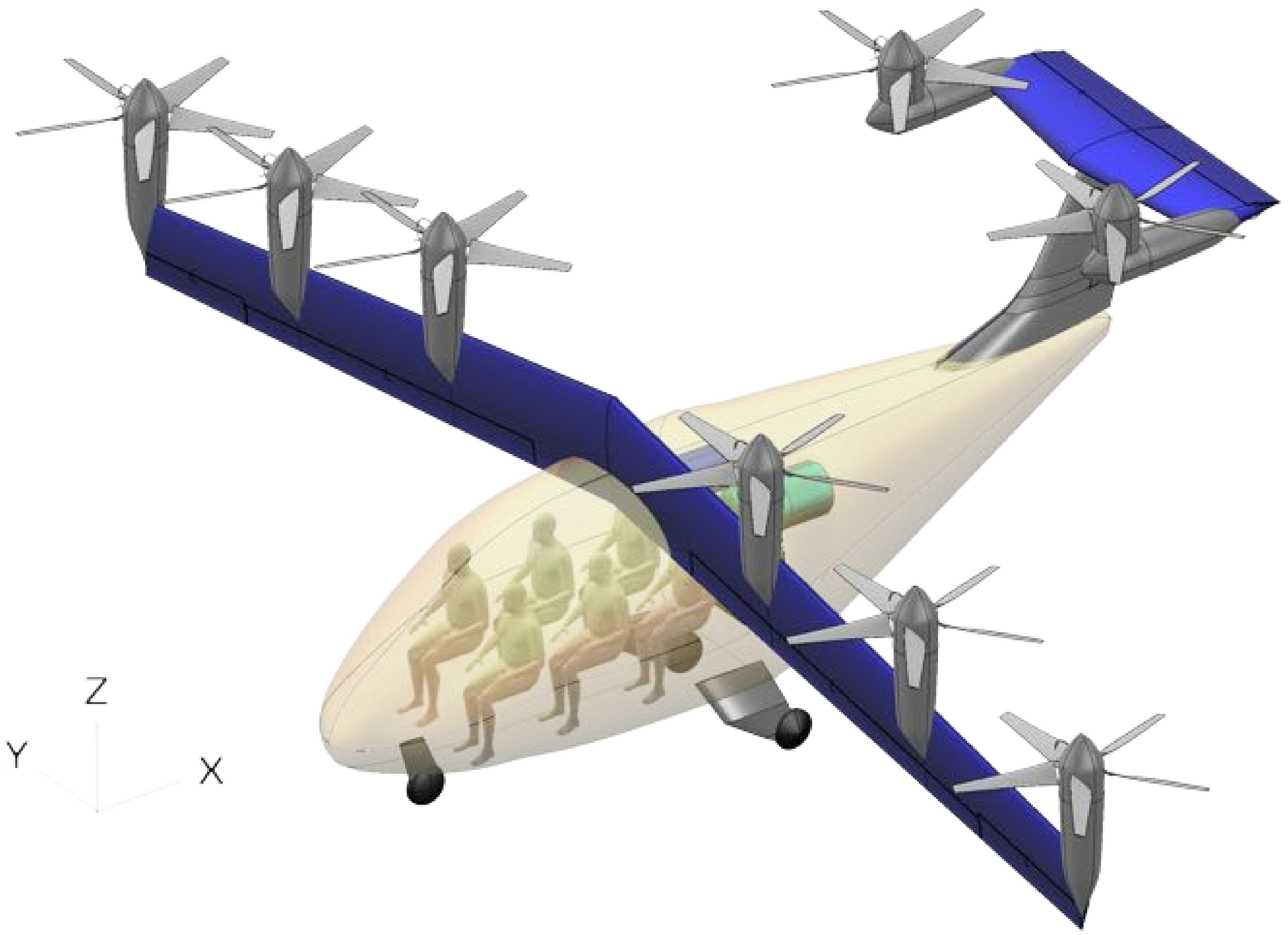
Airframe-propulsion integration demands CFD-based MDO

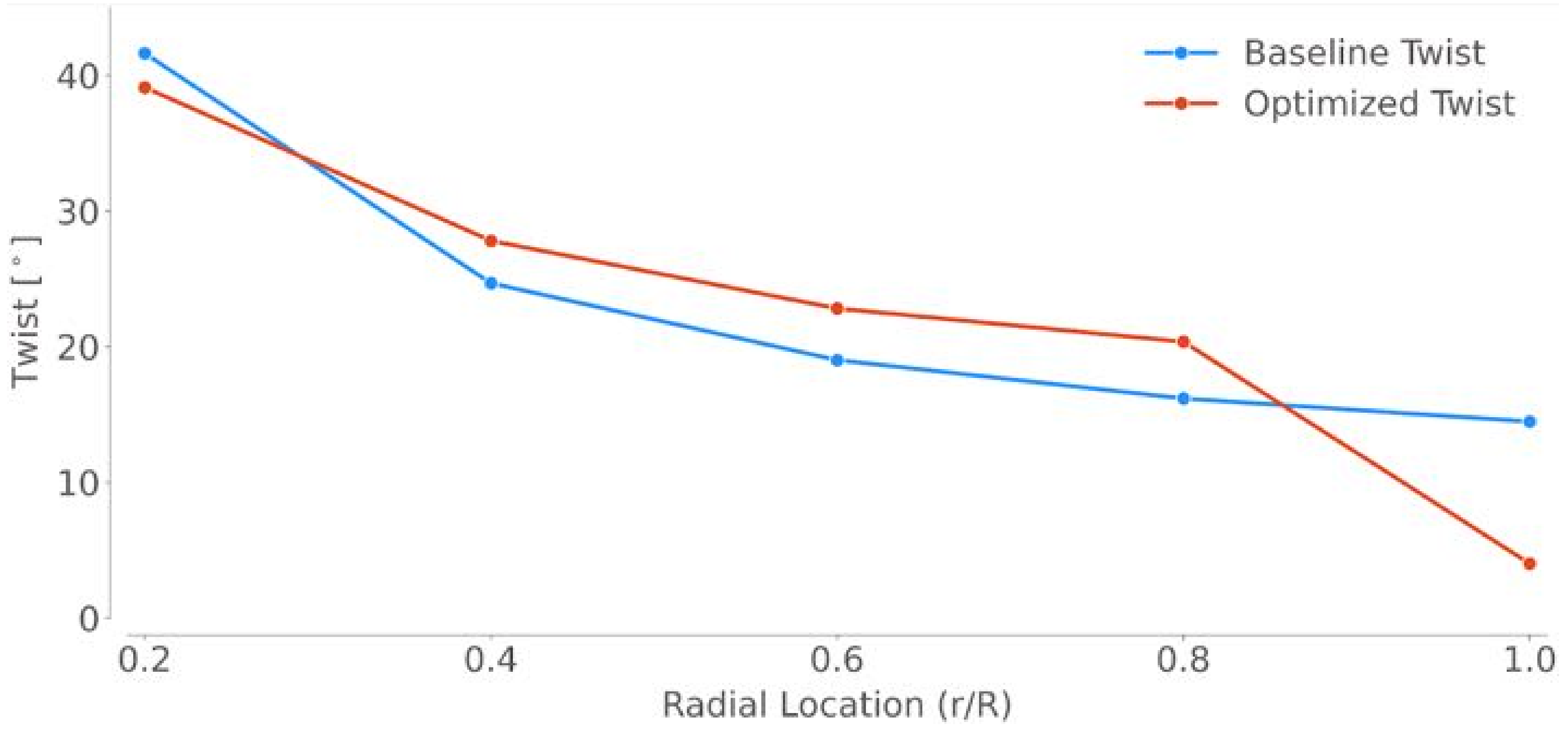


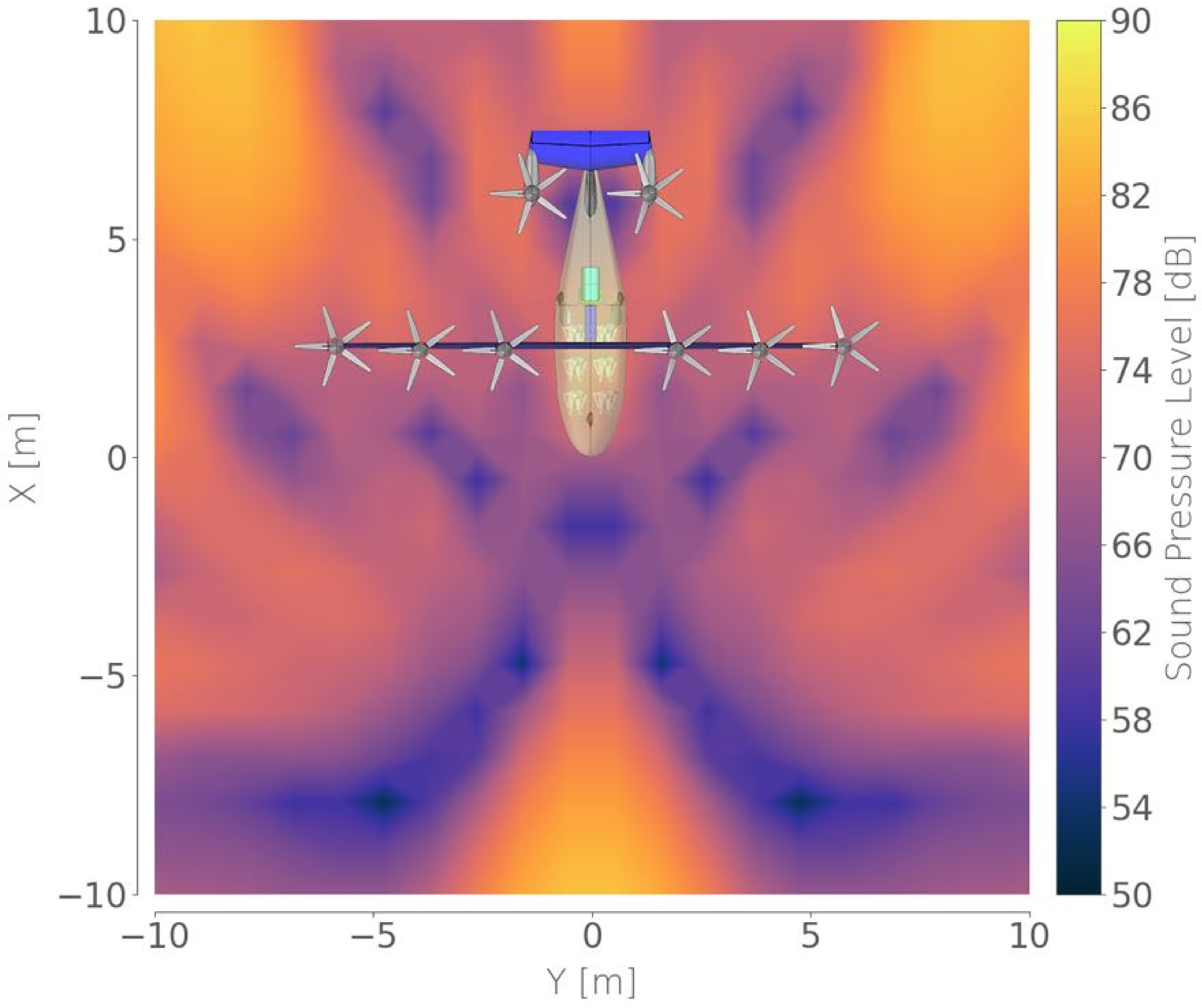
This is the STARC–ABL concept

Yildirim, Gray, Mader, and Martins. **Aeropropulsive design optimization of a boundary layer ingestion system**. *AIAA 2019- 3455*.

Rotor optimization of NASA Tiltwing vehicle subject to noise constraints

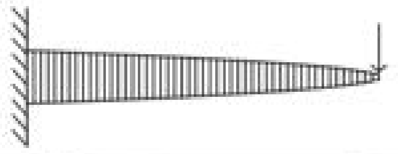
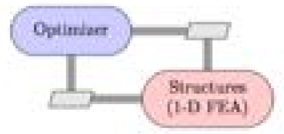
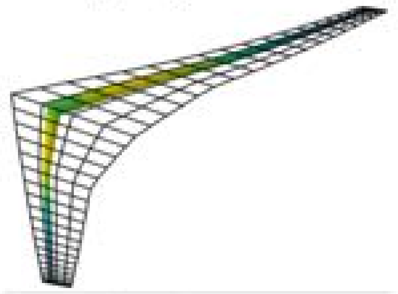
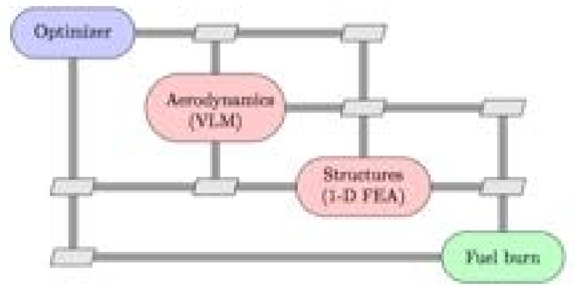
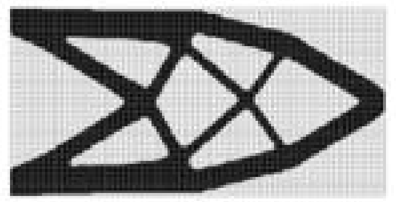
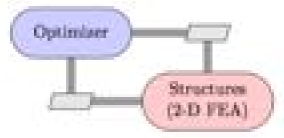
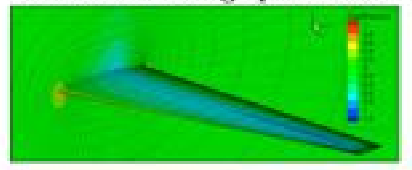
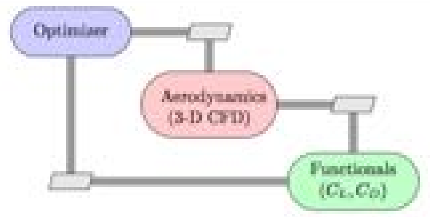
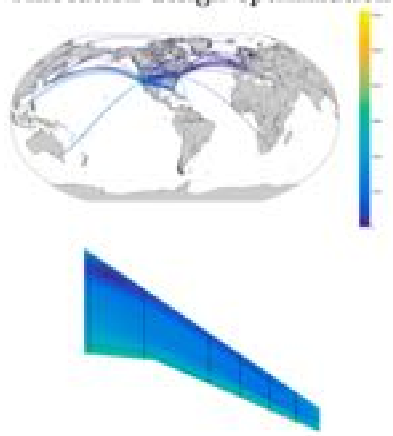
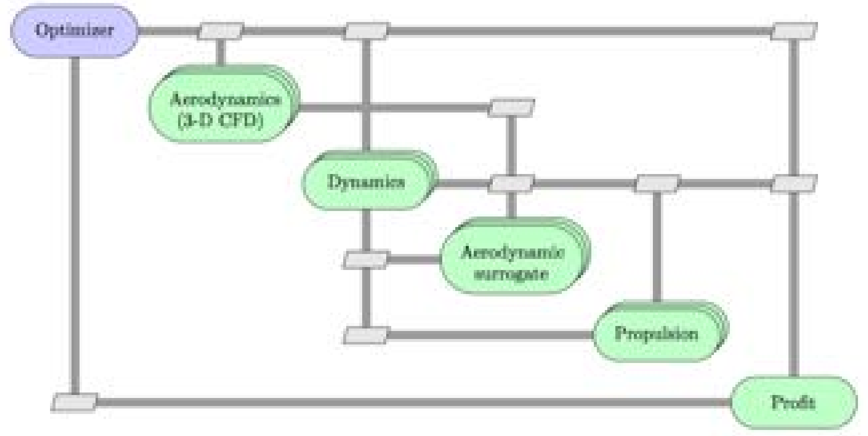
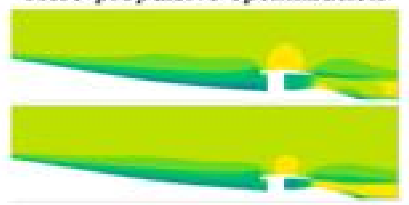
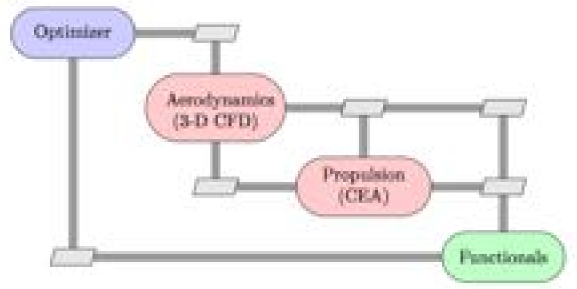
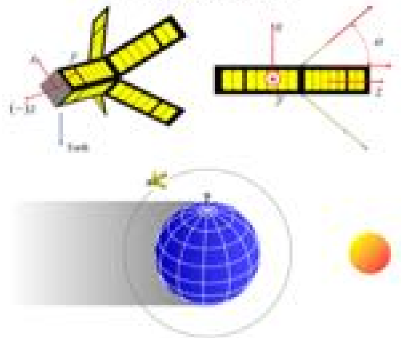
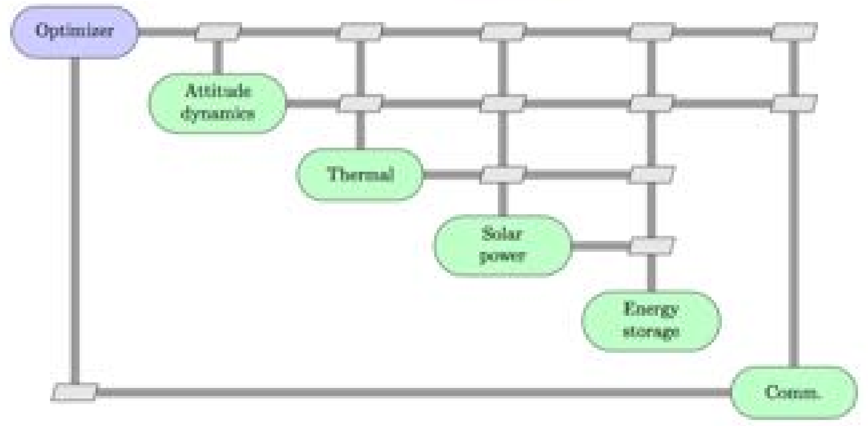
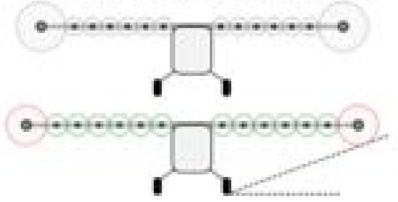
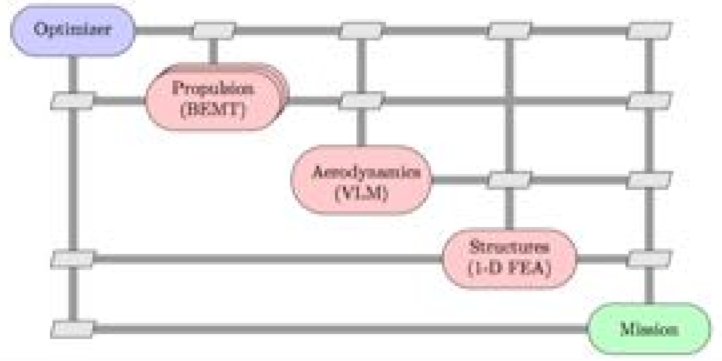






Pacini et al. **Towards Efficient Aerodynamic and Aeroacoustic Optimization for Urban Air Mobility Vehicle Design.** AIAA SciTech 2022.

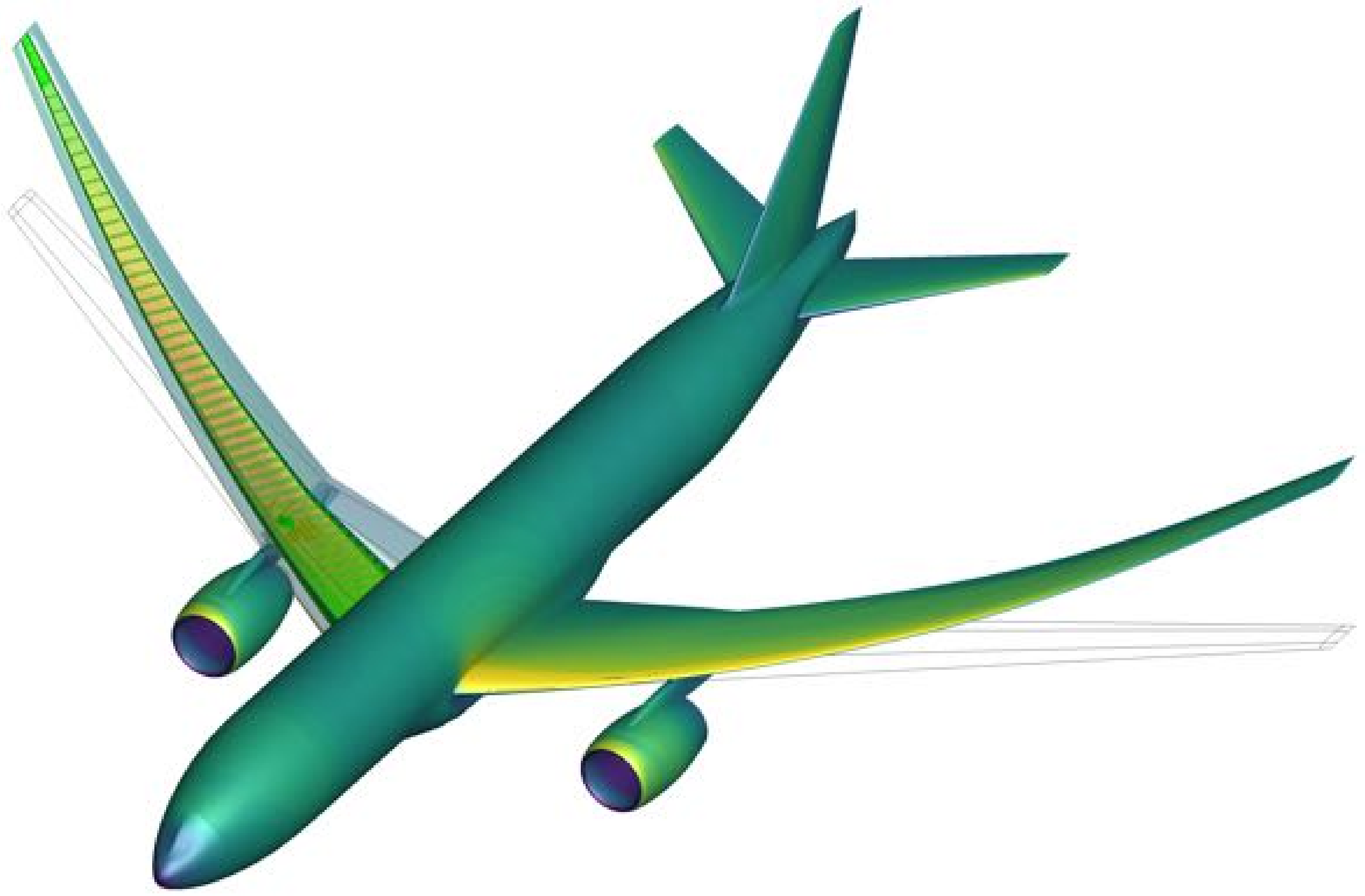
Other OpenMDAO applications

Problem	Model structure	Design variables	Objective	Constraints
Cantilever beam thickness opt. 		thickness distribution	compliance	weight
Low-fidelity wing aerostructural opt. 		thickness & twist dist.	fuel burn	trim stress
Structural topology optimization 		element densities	compliance	mass fraction
RANS-based wing optimization 		shape variables	drag coefficient	lift coefficient
Allocation-design optimization 		wing variables; altitude profiles; cruise Machs; allocation vars.	profit	wing geometry; thrust limits; demand & fleet limits
Aero-propulsive optimization 		inlet shape variables	fuel burn	trim
CubeSat MDO 		solar panel angle; antenna angle; num. radiators; power distribution; attitude profile; solar panel controls	data downloaded	batt. charge rate; batt. charge level
CTOL electric aircraft MDO 		altitude prof.; velocity prof.; prop RPM profs.; prop chord; prop twist; prop diam.; wing twist; beam thickness	range	average speed; eqs. of motion; max. power; min. torque; ground clear.; tip speed; wing failure

Gray, Hwang, Martins, Moore, and Naylor. **OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization**. *Structural and Multidisciplinary Optimization*, 2019

Summary

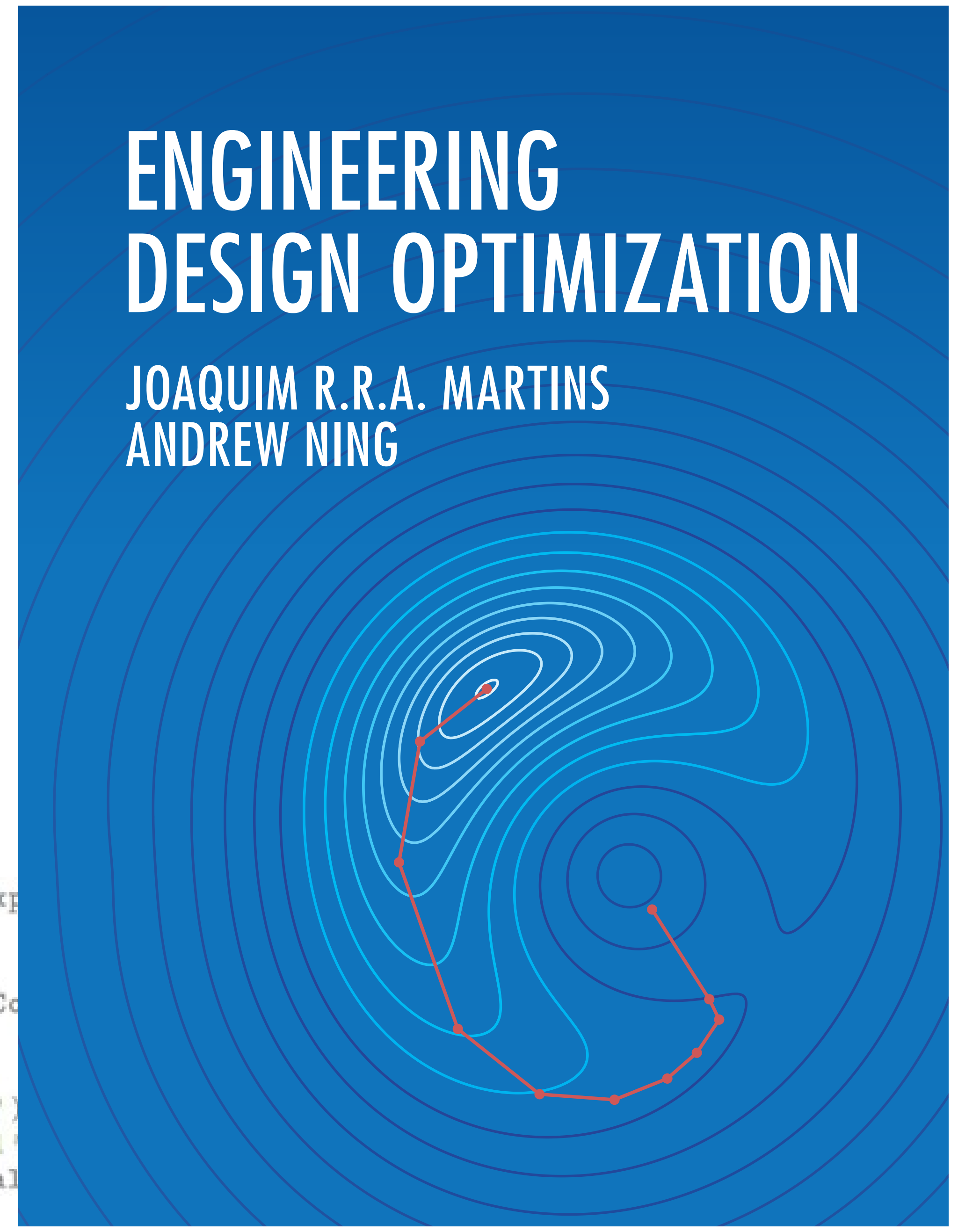
- Gradient-based optimization and efficient gradient computation are a powerful combination.
- Implementing adjoint methods is hard work, but it is worth it.
- Demonstrated large-scale high-fidelity aircraft design applications.
- OpenMDAO facilitates the implementation of these methods for multidisciplinary problems.
- A lot more work to do!

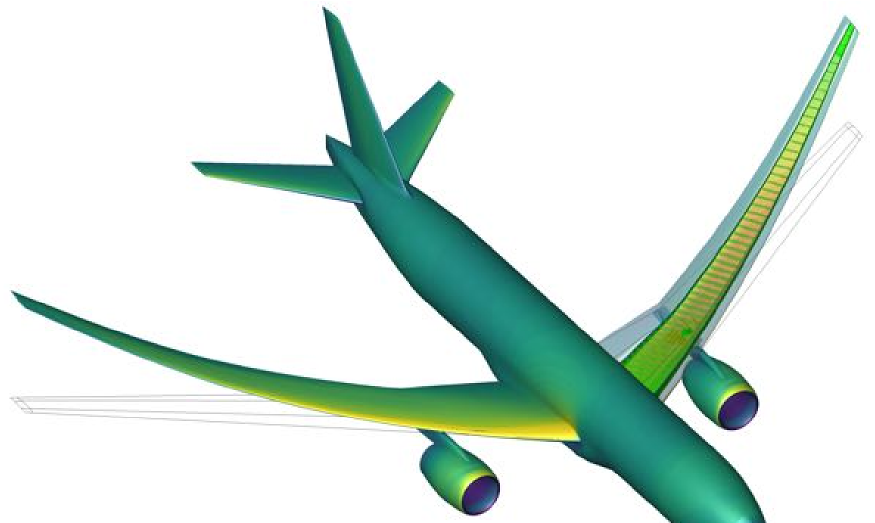


Many of the implemented theoretical developments are now available as open-source software

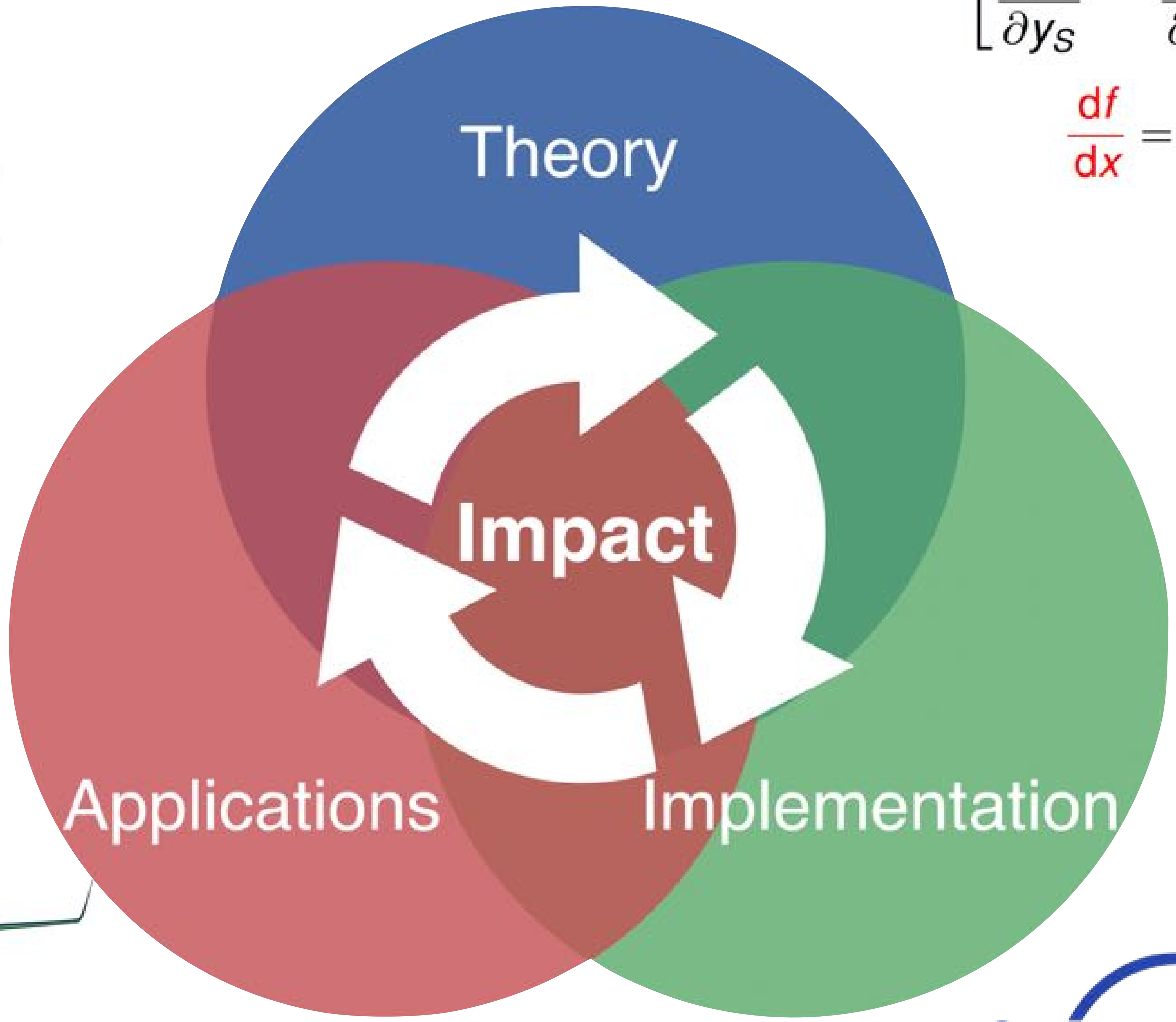
$$\begin{bmatrix} \frac{\partial R_A}{\partial y_A}^T & \frac{\partial R_S}{\partial y_A}^T \\ \frac{\partial R_A}{\partial y_S}^T & \frac{\partial R_S}{\partial y_S}^T \end{bmatrix} \psi = \frac{\partial f}{\partial y}^T$$

$$\frac{df}{dx} = \frac{\partial f}{\partial x} - \psi^T \frac{\partial R}{\partial x}$$





AIRBUS




```
import numpy as np

from openmdao.api import Exp

class Discipline1(ExplicitCo

    def setup(self):
        self.add_input('y2'
        self.add_output('y1
        self.declare_partial

    def compute(self, inputs, outputs):
        outputs['y1'] = inputs['y2'] ** 2

    def compute_partials(self, inputs, partials):
        partials['y1', 'y2'] = 2 * inputs['y2']


class Discipline2(ImplicitComponent):

    def setup(self):
        self.add_input('x')
        self.add_input('y1')
        self.add_output('y2')
        self.declare_partials('y2', 'x')
        self.declare_partials('y2', 'y1')
        self.declare_partials('y2', 'y2')

    def linearize(self, inputs, outputs, partials):
        partials['y2', 'x'] = -outputs['y2']
        partials['y2', 'y1'] = (-outputs['y2'] * np.exp(-inputs['y1'] *
                                outputs['y2']))
        partials['y2', 'y2'] = (-inputs['y1'] *
                                np.exp(-inputs['y1'] * outputs['y2']) - inputs['x'])
```

EMBRAER

openMDAO

AERION SUPERSONIC

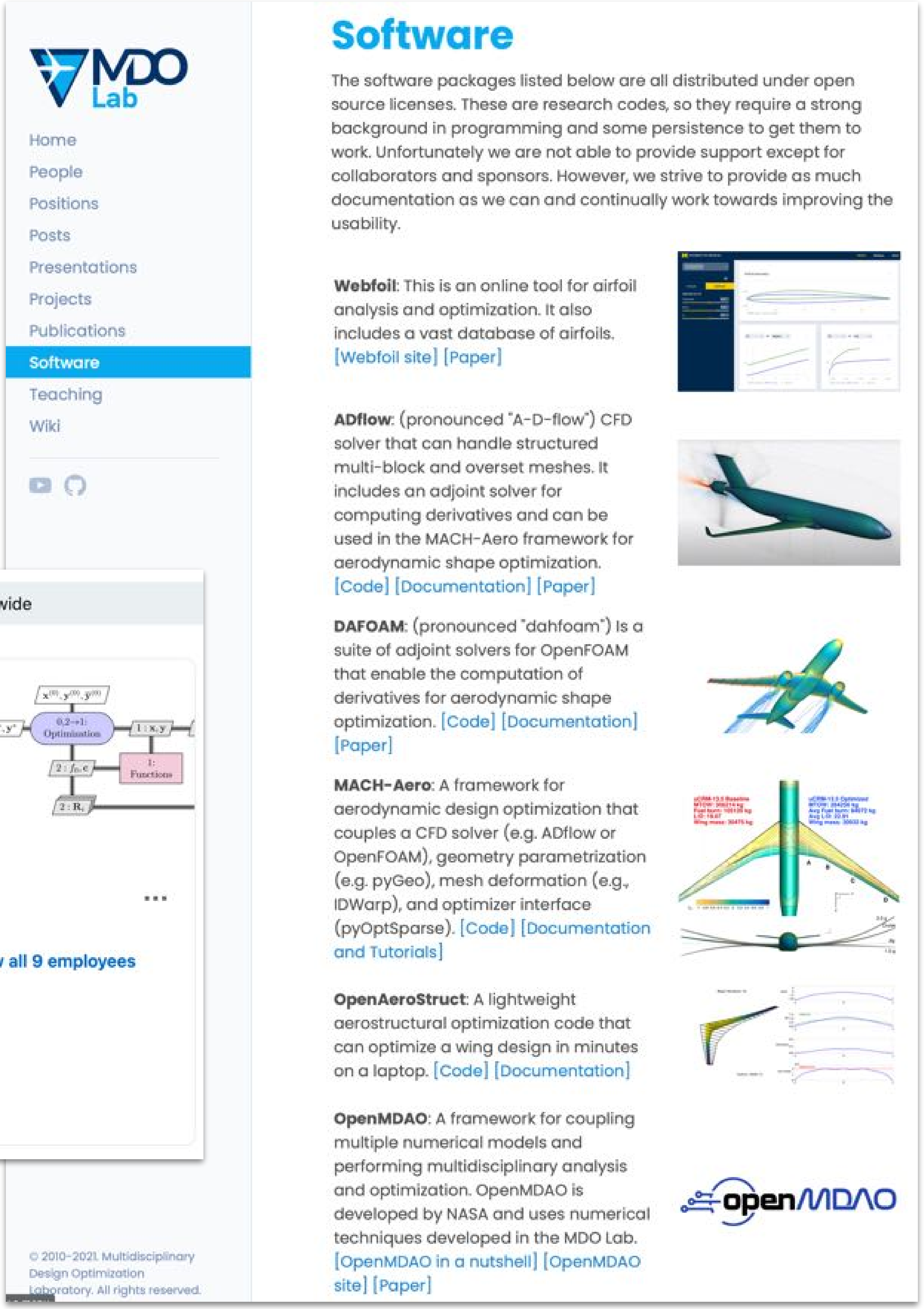
Aurora FLIGHT SCIENCES

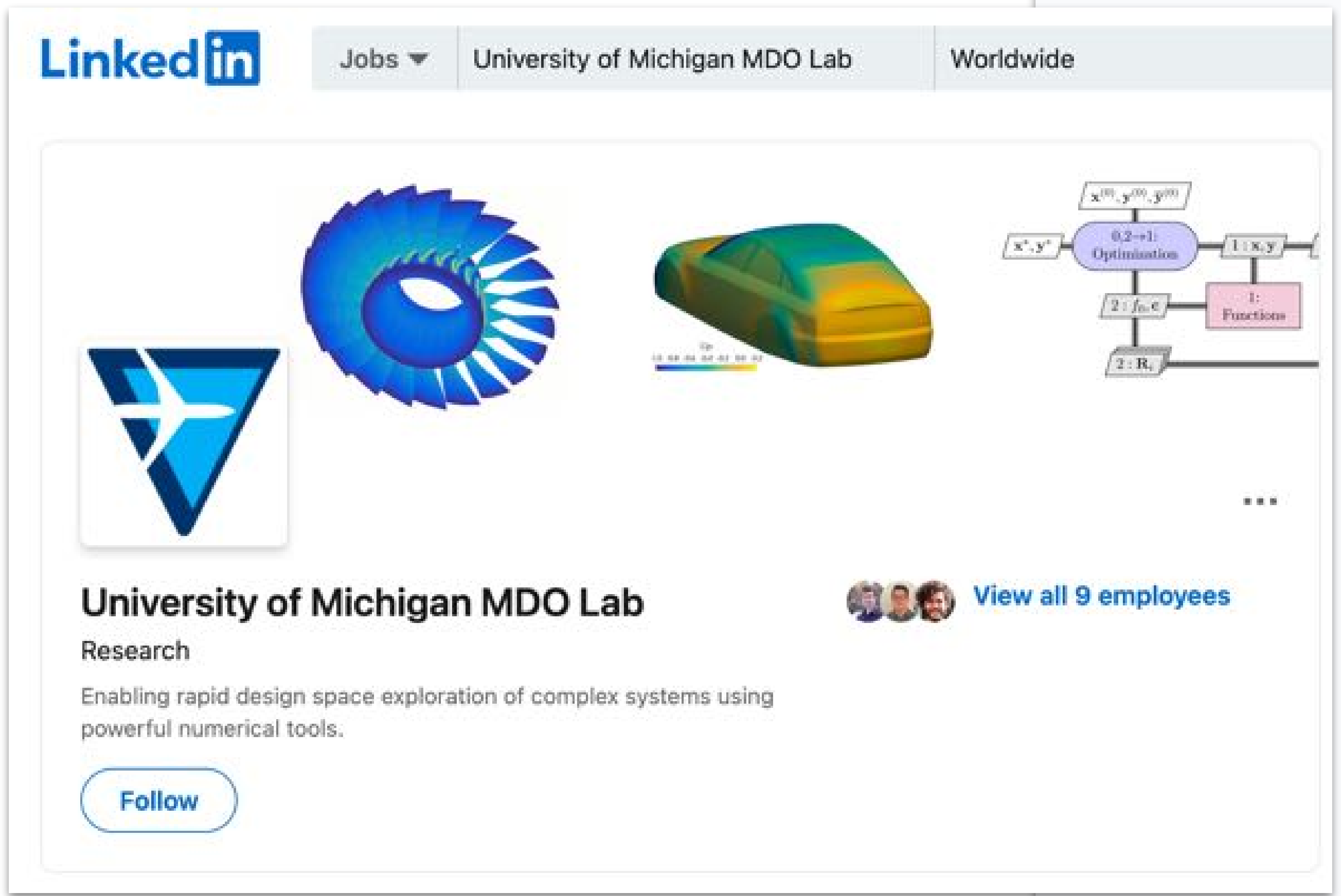
Go forth and optimize!

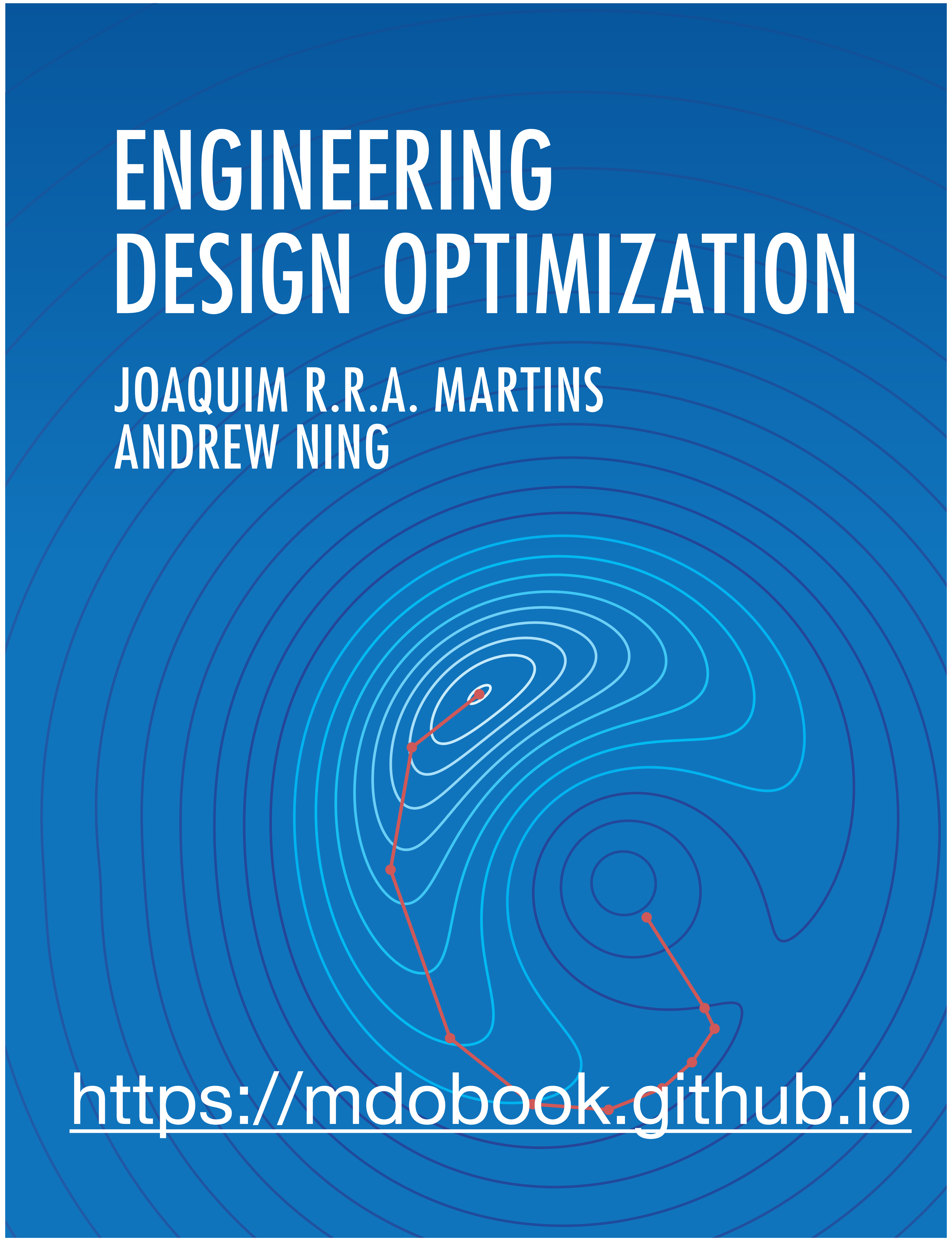
AEROSPACE ENGINEERING
UNIVERSITY of MICHIGAN

http://mdolab.engin.umich.edu/publications

More information: http://mdolab.engin.umich.edu







https://mdobook.github.io