Towards a Framework for Parametric Models and Geometric Sensitivity Computations with SU2 and pyCAPS

Yiren Shen*, Harsh C. Patel†, and Juan J. Alonso‡
Stanford University, Stanford, CA 94305, USA

Marshall C. Galbraith §
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Shape design optimization relies on obtaining objective and constraint functional sensitivities to geometric parameters in order to explore the design space effectively. To facilitate computer-aided design (CAD) applications in industry, this study presents a workflow for effectively handling geometric constraints and computing analytic sensitivities. The workflow is capable of computing parametric sensitivities by combines geometric sensitivities obtained using Engineering Sketch Pad with adjoint surface sensitivities from the SU2 adjoint solver through a dot product within the pyCAPS framework. The effectiveness and limitations of this workflow are evaluated through three test cases, including a preliminary demonstration of a shape optimization problem in wing design. The results indicate that the conceptual effectiveness of the proposed workflow, while noting the need for further improvement in the mesh generation routine.

I. Nomenclature

$\mathcal{A}$	=	aerodynamic residuals
E	=	total energy per unit mass
$f_{i,j}$	=	face parametric coordinates
I	=	output quantity of interest
$n_{i,j}$	=	surface normal at face coordinate (i, j)
p	=	geometric parameters in CAD modeling
q	=	vector of aerodynamic state variables
s	=	a surface on CAD geometry
$\vec{v}$	=	flow velocity vector
x_s	=	aerodynamic surface mesh coordinates
x_v	=	aerodynamic volume mesh coordinates

II. Introduction

Extensive research has been conducted on multidisciplinary design and optimization (MDO) applications for aircraft design in recent years [1–7]. Within the context of aerodynamic shape optimization, a sub-discipline of MDO, gradient-based optimization algorithms are routinely employed to recursively improve objective functionals and conducts design space exploration using heuristics [8, 9]. Recent efforts within the SU2 open-source community have led to the development of a multi-physics analysis and design software suite [10, 11]. The discrete adjoint solver in SU2 is implemented using algorithmic differentiation, which enables efficient and accurate computation of sensitivities of functions of interest with respect to an arbitrary number of design variables [12–14]. Since the introduction of the discrete adjoint solver in SU2, various works have used SU2's sensitivity outputs for gradient-based design parameter optimization, such as design benchmark optimization cases from the Aerodynamic Design Optimization Discussion Group (ADODG) [15], wind turbines [16], and noise reduction [17].

To perform shape optimization, a geometry must be defined with design variables [8], which can be parameterized using constructive or deformative methods [18]. The constructive methods use geometric parameters to define the geometry. Traditionally, polynomials and B-Splines are used to construct geometries [19–21]. More recently, methods have been developed to use class function transformations (CST) (or shape function transformations) to define the surface

*PhD Candidate, Department of Aeronautics and Astronautics, AIAA Student Member.
†PhD Candidate, Department of Aeronautics and Astronautics, AIAA Student Member.
‡Vance D. and Arlene C. Coffman Professor, Department of Aeronautics and Astronautics, AIAA Fellow.
§Research Engineer, Department of Aeronautics and Astronautics, AIAA Senior Member.

geometry of an airfoil [22, 23]. Geometry surrogate modeling, which relates discrete geometry surface displacement with computer-aided design (CAD) parameter changes using proper orthogonal decomposition, has also been examined to produce geometric sensitivities [24]. Alternatively, geometry parameters can be defined directly through CAD, but a CAD engine is required within the optimization loop to generate the geometry surface.

Deformative methods utilize geometric parameters to deform a discretized geometry. These methods vary from discrete deformation, where the discrete geometry is perturbed [25], to the superposition of analytical functions such as cosine and Hicks-Henne bumps [15, 26], and the use of free-form deformation (FFD) boxes. The FFD method involves embedding the original discrete geometry surface in a parametrically defined, flexible control frame that is defined by control points [27]. By deforming the control points, the perturbation is smoothly propagated into the control frame, which in turn deforms the original surface [27]. The construction, size, and number of design parameters of the FFD box are based on user expertise, usually through a trial-and-error approach. Despite this limitation, optimization using geometric sensitivities through FFD have been widely used [28–32]. Moreover, methods have also been developed to automatically refine and optimize the placement of FFD control points [27, 33, 34]. `SU2` utilizes deformative methods including superposition of analytical functions and FFD for geometric handling.

Despite the maturity of deformative methods, several limitations persist in practical industry applications that extensively use CAD for geometry definition [35, 36]. Firstly, the deformation and design parameters are defined on a discrete surface, which poses major limitations for real-world applications. Optimized geometry resulting from this approach often need to be manually remodeled in a CAD software, leading to prolonged design-and-test cycle and potential manual errors [37]. Secondly, in order to obtain the original sensitivity of CAD geometry design parameters, it is necessary to perform additional chain-rule multiplication using the deformation design variable sensitivities. Furthermore, realistic design intents and constraints built into the CAD model of the original geometry are lost during the analysis [38]. Although imposing constraints on FFD control points can alleviate certain issues, such as bounding the deformation or linking a group of control points, these constraints may not be effective in enforcing the complex geometry relationships found in CAD models. Moreover, the implementation of such constraints may be cumbersome due to the high degree of freedom involved in defining multiple design parameters.

These limitations have drawn the attention of researchers to CAD-based parameterization using constructive methods. In the past decade, numerous approaches for obtaining sensitivities of output functionals to CAD design parameters have been proposed [37, 39–43]. Among these methods, the Computational Aircraft Prototype Syntheses (`CAPS`), shipped with Engineering Sketch Pad (`ESP`), caught the interest of the authors due to its demonstrated capability to construct solid geometry from CAD-like parameters, calculate parametric sensitivities, and handle geometric constraints efficiently [43, 44]. In addition, this framework includes Application Programming Interfaces (APIs) to `SU2` and meshing packages[42], allowing efficient field data communication among the geometry builder, meshing software, and `SU2` solver to calculate parametric sensitivities. In this paper, we present a workflow for interfacing `SU2` with `CAPS` to handle geometric constraints and obtain functional sensitivities with respect to geometric parameters for MDO.

This paper is organized as follows. Section III provides an overview of the computational tools and their application to analysis and sensitivity analysis. Section IV introduces the test cases used for framework verification and validation and describes the successful outcomes of the framework application to aerodynamic applications of interest. Finally, Section V summarizes the key contributions of this research effort and outlines potential future work.

III. Methods

A. Geometric Sensitivities

A geometric sensitivity $\mathrm{d}x_s/\mathrm{d}p$ represents the derivative of spatial displacement at any point on the geometry with respect to one or more CAD design parameters [45]. Due to the proprietary nature of mainstream CAD software, many researchers rely on finite-difference methods, which use surface velocities computed by perturbing design parameters and examining the spatial shift of discretized surfaces [39–41]. In this approach, the topology of the deformed and initial mesh must be consistent such that point associations can be established and vertex velocities can be defined [37, 45]. These deficiencies have limited the usage of finite-difference methods in the calculation of geometric sensitivities.

Alternatively, if the solid geometry build process can be accessed, including the generation of primitives and construction operations, surface geometric sensitivities $\partial w/\partial p$ can be calculated analytically [45]. The use of `ESP` enables such computations. `ESP` utilizes the Engineering Geometry Aerospace Design System (`EGADS`) [46], an Application Programming Interface (API) built on top of the solid modeling geometry kernel OpenCASCADE [47], to support both a bottom-up construction approach as well as the ability to perform Constructive Solid Geometry (CSG)

operations [44, 48]. In the ESP environment, a geometry is defined by an OpenCSM script that uses constructive solid modeling to "grow" geometry from primitives by taking a series of operations while bookkeeping a feature tree [48]. As the analytical surface geometric sensitivities for primitives are well-defined, and the transformations of sensitivities can be traced from the feature tree using the chain rule, the surface geometric sensitivity of the solid geometry can be calculated analytically. This surface geometric sensitivity can then be projected onto an arbitrary discretization of the solid body to compute geometric sensitivity as

$$\frac{\mathrm{d}x_s}{\mathrm{d}p} = \frac{\partial s}{\partial p} \cdot \hat{n} + \frac{\partial x_s}{\partial f} \frac{\mathrm{d}f}{\mathrm{d}p} \,. \tag{1}$$

where f is the face parametric coordinate, s is a face on the geometry, and $\hat{n}$ is the face normal. Such capabilities allow for both analytical and enhanced finite-difference parametric sensitivity calculations, and further details on the implementation were articulated by Dannenhoffer and Haimes [45].

This study utilizes ESP to construct solid geometry and compute geometric sensitivities. The OpenCSM scripting language is used to define the design parameters and the feature tree for building solid geometry, which is then built via EGADS and passed on to meshing software through the Pointwise® [49] API using pyCAPS, a CAPS Python extension [50]. This workflow for generating discrete surface and volume meshes will be described in Section III.D.

B. Flow Solver

The SU2 software suite is an open-source collection of computational fluid dynamics (CFD) tools written in C++ and Python for performing multi-physics simulation and design [51–53]. SU2 is built specifically for the analysis of partial differential equations (PDEs) and PDE-constrained optimization problems on unstructured meshes with state-of-the-art numerical methods for spatial integration, time integration, and convergence acceleration with multi-grid [53].

The finite-volume flow solver in SU2 solves the compressible Navier-Stokes equations, which expressed in differential form are

$$\mathcal{A}\left(q\right) = \frac{\partial q}{\partial t} + \nabla \cdot F^{c}\left(q\right) - \nabla \cdot F^{v}\left(q\right) - Q_{s}\left(q\right) = 0 \,, \tag{2}$$

where t is a time variable, $q = [\rho, \rho\vec{v}, \rho E]^{T}$ is the vector of conservative variables, ρ is the fluid density, $\vec{v}$ is the fluid velocity vector in a Cartesian coordinate system, and E is the total energy per unit mass, $F^{c}\left(q\right)$ are the convective fluxes, $F^{v}\left(q\right)$ are the viscous fluxes, and $Q_{s}\left(q\right)$ a generic source term.

The inviscid Euler equations can be recovered by omitting the viscous fluxes $F^{v}(q)$ in (2). In this study, the flows around aerodynamic bodies are modeled by the compressible Euler and incompressible Navier–Stokes equations. The governing equations are discretized using the finite-volume method with an edge-based data structure. The Jameson–Schmidt–Turkel [54] scheme is used for spatial discretization and the implicit Euler scheme for the time marching. SU2 also features continuous and discrete adjoint solvers for the computation of analytic sensitivities.

C. Adjoint Flow Sensitivities

Aerodynamic design optimization using high-fidelity analysis models requires solvers for the governing equations of the fluid flow. The discretized equations governing the flow, as implemented in the solver, are typically expressed in the residual form as a vector of non-linear equations for each control volume in the fluid domain,

$$\mathcal{A}\left(q,\, x_{v}\right) = 0 \,, \tag{3}$$

where q is the vector of conservative flow state variables at each vertex in the computational fluid mesh and x_v are the mesh coordinates of the fluid domain. For aerodynamic shape optimization, some quantity of interest I, which represents a measure of performance or feasibility, is also a function of the flow and mesh geometry that can be expressed in general form as

$$I = I\left(q,\, x_{v}\right) \,. \tag{4}$$

Using the chain rule, the total derivative of this quantity with respect to the volume mesh coordinates becomes

$$\frac{\mathrm{d}I}{\mathrm{d}x_{v}} = \frac{\partial I}{\partial x_{v}} + \frac{\partial I}{\partial q} \frac{\mathrm{d}q}{\mathrm{d}x_{v}} \,. \tag{5}$$

The total derivative $\mathrm{d}q/\mathrm{d}x_v$ describes the sensitivity of the flow states to the mesh nodes, which requires a computationally expensive solution of the governing equations. However, observing that for all mesh coordinates, the solution q must

satisfy (3) so we can apply the chain rule once more to obtain

$$\frac{\mathrm{d}\mathcal{A}}{\mathrm{d}x_v} = \frac{\partial \mathcal{A}}{\partial x_v} + \frac{\partial \mathcal{A}}{\partial q}\frac{\mathrm{d}q}{\mathrm{d}x_v} = 0\,. \tag{6}$$

Substituting this equality back into the expression for the total derivative (5), we obtain

$$\frac{\mathrm{d}I}{\mathrm{d}x_v} = \frac{\partial I}{\partial x_v} - \frac{\partial I}{\partial q}\frac{\partial \mathcal{A}}{\partial q}^{-1}\frac{\partial \mathcal{A}}{\partial x_v}\,. \tag{7}$$

Two main approaches can be distinguished for the solution of the sensitivity equations: the so-called *direct* and *adjoint* methods. By changing the order of operations used to solve (7), the overall computational cost of the solution is affected significantly. The direct method involves solving the linear system (6), which requires a linear solve for each mesh node. On the other hand, the adjoint method introduces *adjoint variables* ψ^T that must satisfy

$$\frac{\partial \mathcal{A}}{\partial q}^T = \psi \frac{\partial I}{\partial q}^T\,. \tag{8}$$

This system must be solved once for each quantity of interest I, after which the total derivatives in equation (7) can be obtained as

$$\frac{\mathrm{d}I}{\mathrm{d}x_v} = \frac{\partial I}{\partial x_v} + \psi^T \frac{\partial \mathcal{A}}{\partial x_v}\,. \tag{9}$$

The final step for obtaining total surface sensitivities requires a projection

$$\frac{\mathrm{d}I}{\mathrm{d}x_s} = \frac{\mathrm{d}I}{\mathrm{d}x_v}\frac{\mathrm{d}x_v}{\mathrm{d}x_s}\,, \tag{10}$$

where $\mathrm{d}x_v/\mathrm{d}x_s$ is the sensitivity obtained by differentiating the mesh deformation algorithm. If the number of surface coordinates x_s significantly exceeds the number of outputs I, the adjoint method incurs a reduced computational cost compared to the direct method.

D. Parameteric Sensitivity

To calculate parametric sensitivity $\mathrm{d}I/\mathrm{d}p$, which is defined as the sensitivity of objective functionals to CAD design parameters, the following chain rule dot product can be applied

$$\frac{\mathrm{d}I}{\mathrm{d}p} = \frac{\mathrm{d}I}{\mathrm{d}x_s}\frac{\mathrm{d}x_s}{\mathrm{d}p}\,. \tag{11}$$

This dot product is implemented in `CAPS`, which is an MDO framework that incorporates `ESP` and facilitates multi-disciplinary and multi-fidelity analysis by combining geometry, meshing, and analysis model generation into a unified framework [43, 50]. `CAPS` provides interfaces, referred to as Analysis Interface Modules (AIMs), for exchanging data between the `CAPS` framework and analysis tools. AIMs are provided for low-fidelity analysis, high-fidelity CFD, structural analysis, and meshing packages [42, 50]. In this work, the `EGADS`, `Pointwise`®, and `SU2` AIMs of `CAPS` are used to construct a workflow for defining parametric solid geometry, constructing surface and volume meshes, computing primal and adjoint CFD solutions, and exchanging data among the computational tools. For each test case, a Python script employing `pyCAPS` is created to define and execute the analysis workflow illustrated in Fig. 1. A detailed explanation of this workflow is provided in the subsequent sections.

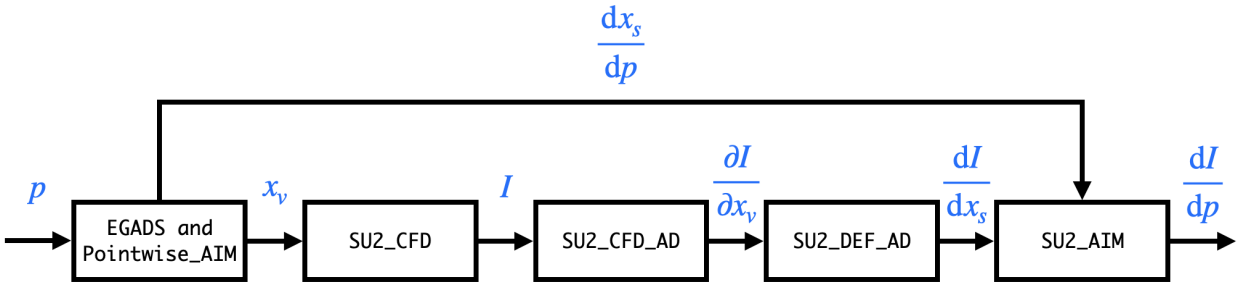


Fig. 1 Block diagram of geometric parameter sensitivity calculation workflow using pyCAPS and SU2.

1. Defining and Modifying Geometric Parameters

The analysis process begins by initializing a `caps.Problem` class, which describes and contains all the elements related to a problem or mission, including geometric models that was built using `ESP` using approaches described in Section III.A, analyses, and data connectivity. Once the `caps.Problem` class is initiated, all CAD geometric parameters can be modified and accessed. The geometric parameters are indicated by p in Fig. 1.

2. Generating Surface and Volume Meshes

The generation of surface and volume meshes for third-party software is accomplished with AIMs by instantiating a `caps.Analysis` class. This class corresponds to a specific analysis tool and is stored within the `analysis` attribute of the `caps.Problem` class as instances within a Python dictionary. Various meshing software, including Tetgen, AFLR, and `Pointwise`® [49, 55, 56], are currently supported by CAPS. `Pointwise`® is used for this study due to the authors' familiarity with the software. Mesh properties, such as average nodal spacing, maximum domain element edge length, and growth factors, are controlled by setting analysis inputs. Domain-specific attributes, such as setting the average nodal spacing on a specific face or an edge, can be designated in `OpenCSM` script with "ATTRIBUTE" statements. The `preAnalysis` function in `caps.Analysis` is called after mesh properties attribution to generate a Glyph script that defines mesh building procedures for `Pointwise`® to read. The generation of the Glyph script and execution of `Pointwise`® were all through `Pointwise` AIM.

After discretizing the surface of a solid body, the geometric sensitivity are computed using `OpenCSM`, as described in Section III.A. This geometric sensitivity is called during the computation of parametric sensitivities. The `EGADS and Pointwise_AIM` block in Fig. 1 illustrates this step.

3. Primal and Adjoint Flow Solver

A `SU2 caps.Analysis` class is initialized using `SU2` AIM to compute both primal and adjoint solutions. The input for `SU2` AIM is the volume mesh generated in the previous step, which is achieved by passing `Pointwise AIM`'s output to `SU2` AIM's input. `SU2` AIM takes additional input from configuration file definitions. These inputs are then used to write configuration files for direct and adjoint simulations by calling the `preAnalysis` function in `caps.Analysis`.

The execution of both direct and discrete adjoint calculations in `SU2` is initiated by `SU2`'s Python interface, namely `pysu2` and `pysu2ad` [57]. The API for `pysu2` and `pysu2ad` provide users with easier access to adjoint flow states, objective functional values, and adjoint flow sensitivity of objective functionals to surface coordinates $\mathrm{d}I/\mathrm{d}x_s$. Once this sensitivity is obtained, it is passed to `SU2` AIM by calling the `postAnalysis` function in `caps.Analysis`. The overall process, as well as inputs and outputs, are shown in Fig. 1, with three central `SU2` blocks (`SU2_CFD`, `SU2_CFD_AD`, and `SU2_DEF_AD`).

4. Obtaining Parametric Sensitivities

Once the adjoint flow sensitivities have been computed, a dot product between the adjoint flow sensitivities $\mathrm{d}I/\mathrm{d}x_s$ and the geometric sensitivities $\mathrm{d}x_s/\mathrm{d}p$ must be computed to obtain the parametric sensitivities, as shown in Eq. 11. This was achieved by utilizing the `dynout.deriv` function in `caps.Problem` through `SU2` AIM. This step is illustrated by the merging of two field data in the `SU2_AIM` block shown in Fig. 1.

IV. Results

To demonstrate the capability of the developed workflow in obtaining functional sensitivities with respect to geometric parameters, several test cases are presented in this section. First, the calculation of geometric sensitivity for a simple geometry is validated with an analytical expression. Then, the parametric sensitivity is computed for more realistic application problems, one incompressible and one compressible test case, culminating in a representative design optimization of a finite wing.

A. Block

1. Description

For the purpose of validating geometric sensitivity, a rectangular box that mimics a rectangular fin under uniform flow is selected. The geometry is shown in Fig. 2, while the coordinate system is represented by dashed arrows with the

x and z axes. The rectangular fin is defined by three geometric parameters: the length l, width w, and height h of the three edges along the x, y, and z coordinates, respectively. The lengths of l, w, and h for unperturbed geometry are 0.5 m, 1.0 m, and 1.0 m, respectively. The origin of the relative coordinate for CAD is defined at $[0, -0.5, 0]$ and is shown with a blue circle at the bottom-left vertex, which is fixed while all other vertices are defined with a relative displacement from $[0, -0.5, 0]$ using the design parameters l, w, and h.

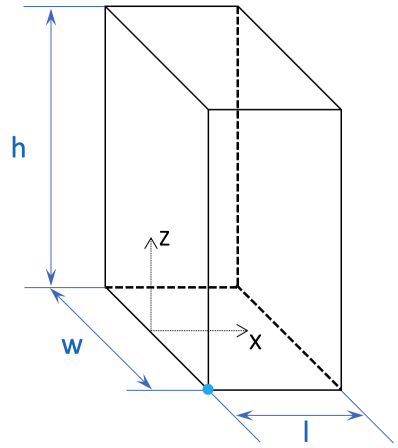


Fig. 2 Schematic of a rectangular fin, with geometric parameters h, w, and l shown in blue and the coordinate system defined with the $x - z$ plane shown in black dashed arrow.

2. Validation

The geometric sensitivities of the discretized surface vertex locations with respect to CAD parameters $\mathrm{d}x_s/\mathrm{d}p$ are verified against analytical expressions. For the z-direction, doubling the design parameter h induces a velocity of 1 ($\partial s/\partial h = 1$), where s is the top surface, on the top surface due to the construction process of the CSG. The surface normal projection of this velocity, according to 1, for the top surface is $[\partial x_{s,x}/\partial h,\ \partial x_{s,y}/\partial h,\ \partial x_{s,z}/\partial h]^T = [0,\ 0,\ 1]^T$. Since modifying h does not result in any deformation of the top surface, it remains flat even after the displacement, with the same inner nodal coordinate $f_{i,j}$ as the original surface. Therefore, we have $\partial f/\partial h = 0$ for the top surface. According to Eq. 1, the geometric sensitivity is the sum of the normal projection of surface velocity and changes of face coordinates with respect to the design parameter along the face [45]. Therefore, the analytical geometric sensitivity for the top surface will be $[0,\ 0,\ 1]^T$.

For all four side surfaces, changing the parameter h induces no normal displacement such that the surface normal projection of surface velocity is zero $\partial s/\partial p = 0$. However, the nodal locations of face parametric coordinates were stretched along the surfaces. The magnitude of the induced velocity $\partial x_s/\partial p$ is proportional to the nodal z-coordinate of the original geometry. Summing both effects, the predicted geometric sensitivity at point $[x,\ y,\ z]$ is then $[\partial x_{s,x}/\partial h,\ \partial x_{s,y}/\partial h,\ \partial x_{s,z}/\partial h]^T = [0,\ 0,\ z/h_0]^T$, where h_0 is the initial height of the block. The analytical geometric sensitivity for the other design parameters w and l can be derived in a similar manner, and the result for point $[x,\ y,\ z]^T = [i,\ j,\ k]^T$ is computed as

$$\left(\frac{\mathrm{d}x_s}{\mathrm{d}p}\right)_{i,j,k} = \begin{bmatrix} \frac{\partial x}{\partial l} & \frac{\partial x}{\partial w} & \frac{\partial x}{\partial h} \\ \frac{\partial y}{\partial l} & \frac{\partial y}{\partial w} & \frac{\partial y}{\partial h} \\ \frac{\partial z}{\partial l} & \frac{\partial z}{\partial w} & \frac{\partial z}{\partial h} \end{bmatrix}_{i,j,k} = \begin{bmatrix} \frac{x}{l_0} & 0 & 0 \\ 0 & \frac{y+0.5}{w_0} & 0 \\ 0 & 0 & \frac{z}{h_0} \end{bmatrix}_{i,j,k} . \tag{12}$$

The analytical geometric sensitivity is compared with the output from `OpenCSM` and the results are shown in Fig. 3. The discretized surface geometric sensitivity is represented by the color scale at the corresponding spatial location. Geometric sensitivities of $\partial x/\partial w$, $\partial x/\partial h$, $\partial y/\partial l$, $\partial y/\partial h$, $\partial z/\partial l$, and $\partial z/\partial w$ are all zero and therefore not shown. The analytical sensitivity values are in good agreement with the `OpenCSM` geometric sensitivity output.

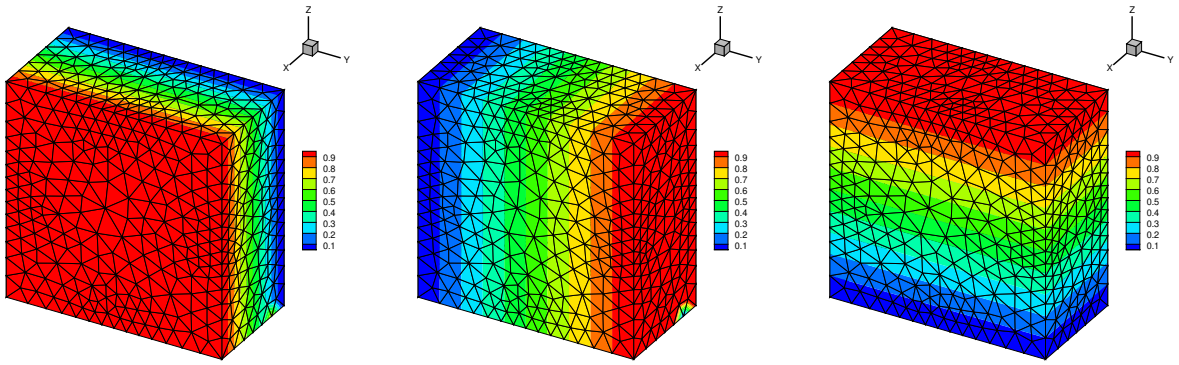


Fig. 3 Geometric sensitivities of surface point locations to CAD parameters $\mathrm{d}x_s/\mathrm{d}p$**.**

B. Heat Exchanger

1. Description

With the geometric sensitivities validated, the parametric sensitivity are computed using the workflow described in Section III. The first test case utilized SU2's incompressible Navier-Stokes solver, where the geometry is inspired by a typical cross-flow heat exchanger in a gas boiler. Only one cell of the heat exchanger's gas channel is modeled in this study. The fluid domain represents the hot air exiting from the burner, entering from the left side of the domain. The free-stream boundary condition is set to a hot incompressible ideal gas at 1900 K and a velocity of 0.2 m/s in the x-direction. The hot gas passes through a pencil-shaped cross-flow fin with a wall temperature of 600 K, set with an isothermal boundary condition. The cooled gas exits the computational domain to the right, which is a pressure outlet boundary condition with a 0 Pa back pressure.

Four CAD parameters are used to construct the pencil-shaped fin, which are defined in Fig. 4. The origin is defined at the center of the nose arc with a diameter of d_c. A ramp connects the nose arc to the aft-section of the fin with horizontal length $l_r = 1.5$ m. The aft-section is defined by a flat plate that begins at $x = l_r$ and extends to $x = l_r + l_f$ with $l_f = 2$ m. The tail of the fin is constructed with another arc with a diameter of $d_t = 0.5$ m centered at $x = l_r + l_f$. The total length of the fin L is defined as $L = \frac{1}{2}(d_c + d_t) + l_r + l_f$. The sensitivity of the integrated total heat flux Q on the fin surface to changes in the nose diameter d_c is studied in this test case.

The fluid domain extends 3 m upstream (2 l_r) and 8 m downstream (4 l_r). The width of the fluid domain is 0.1 m (1/15 l_r), extending along the y-axis into the $x - z$ plane. The $x - z$ cross-section of the fluid domain is shown in Fig. 4. The size of this fluid domain is chosen to ensure that no reflective waves interfere with the boundary layer flow and to mimic a burner gas channel.

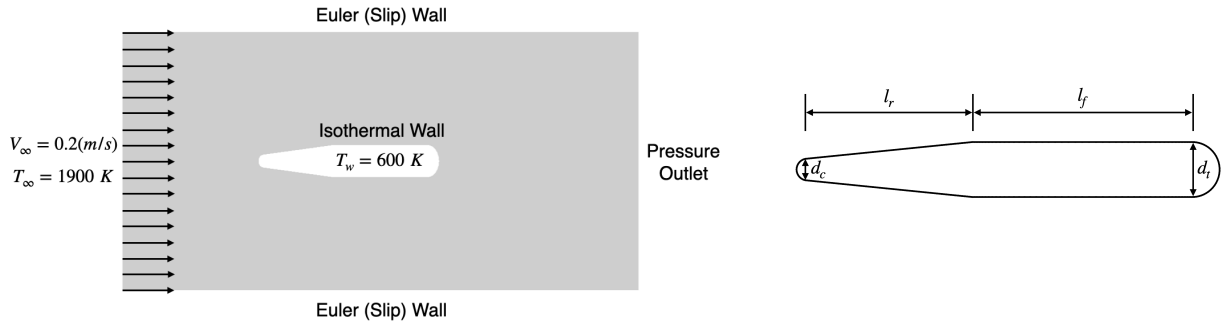


Fig. 4 The $x - z$ **cross-section of the computational domain and a schematic of the fin with CAD parameters.**

2. Validation

A fully unstructured, boundary layer resolved mesh is generated using `Pointwise`® [49] via the `Pointwise` AIM in `pyCAPS`. Meshing parameters such as the boundary layer growth rate and the average edge and face spacing were defined in the `pyCAPS` script and tuned based on a grid refinement study of the integrated heat flux Q along the fin surface and the drag coefficient C_d. Figure 5 shows the convergence history of the grid refinement study and Fig. 6 shows an $x-z$ cross-section of the final mesh used in this study. A surface grid size of 7244 elements with an average element size of $0.16d_c$ is sufficient for convergence of C_d to three significant digits. The boundary layer cell near the isothermal wall has a wall spacing of 0.0001 m with a growth rate of 1.2. The average value of $y^+ = 0.1$ and the maximum value of $y^+ = 0.21$ at the nose.

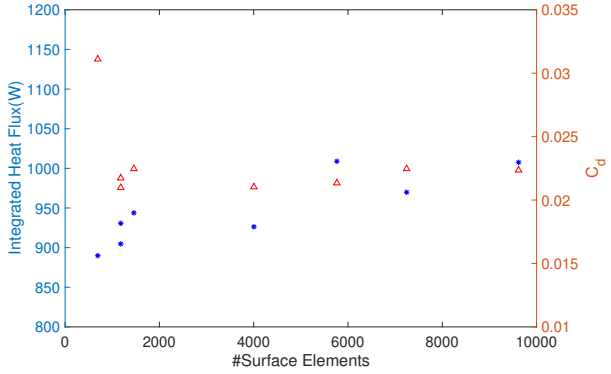


Fig. 5 Mesh refinement study.

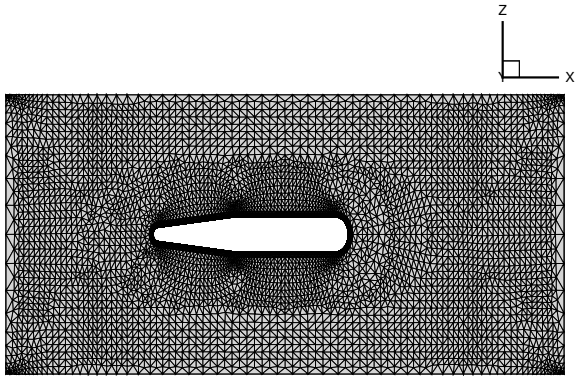


Fig. 6 Final heat exchanger mesh.

Figure 7 shows the history of the pressure residual during the simulation. The convergence criterion for the simulation is a root-mean-square (RMS) pressure residual below -8. The chosen convergence criterion is aligns with typical industrial usage of these numerical tools in the early conceptual design phase, where rapid and efficient exploration of the design space often takes precedence over accuracy. In Fig. 8, the temperature field sampled at the central $x-z$ plane at $y = 0$ is presented. The temperature gradients are found to be concentrated within the boundary layer of the fin, with the highest gradient occurring around the nose. The right-hand side of Fig. 8 displays the surface heat flux across the fin as a function of the normalized x coordinate. The surface heat flux is observed to be maximum at the nose of the fin.

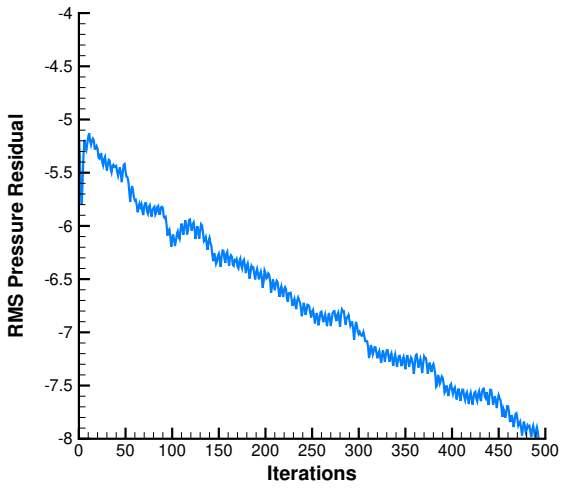


Fig. 7 Convergence history of root-mean-square pressure residual.

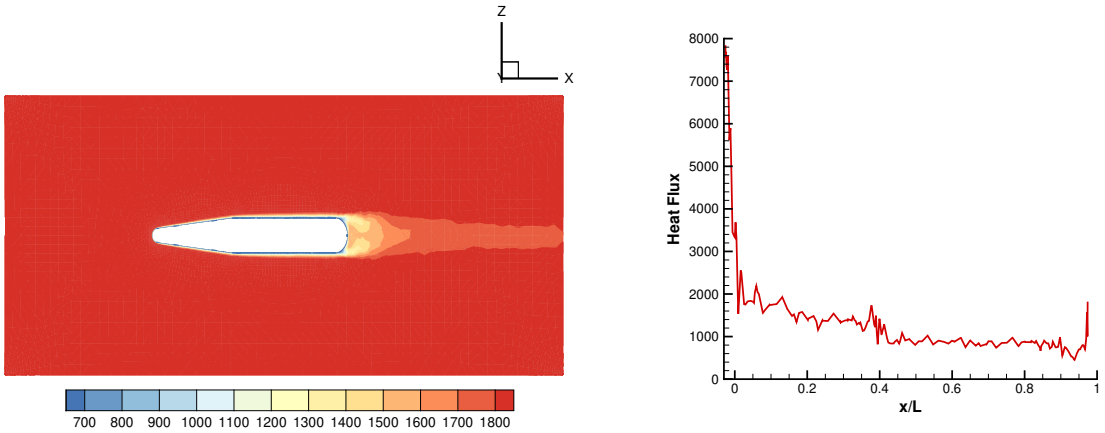


Fig. 8 Temperature field (K) in the $x-z$ plane sampled at $y=0$ and corresponding surface heat flux ($\mathrm{W/m^2}$).

Parametric sensitivity is computed by taking the dot product between the geometric sensitivity and the adjoint flow sensitivity on the fin surface. To verify the parametric sensitivity computed from this workflow, the sensitivities are validated against finite-difference values using the `pyCAPS` framework to change the design parameter value by a finite amount, regenerate the mesh, and rerun the CFD calculation.

The parametric sensitivity of the nose diameter d_c is calculated for nose diameter at 0.25 m. Ideally, the finite-difference sensitivity can be computed using central difference. However, since the mesh regeneration step changes the point discretization and inevitably leads to different truncation error, selecting a proper step size for central difference is daunting. Alternatively, one can construct a linear fit for multiple design points with finite perturbation of design parameters to estimate the sensitivity. Using this approach, the integrated surface heat flux with $d_c = d_c \pm 0.005$ m, $d_c \pm 0.01$ m, and $d_c \pm 0.015$ m are computed. A linear fit of the data series is shown in Fig. 9, which is used to estimate the sensitivity of the integrated heat flux to the design parameter d_c. The observed $\mathrm{d}Q/\mathrm{d}d_c$ is 112.48 W/m, with a R-squared value of 0.4524. The results are shown in Table 1.

d_c	Integrated Heat Flux $[W]$	Analytic Adjoint + CAD	Finite-Difference	Relative Difference [%]
0.25	937.24	102.90	112.48	8.52

Table 1 Validation of sensitivities of integrated heat flux to design parameters d_c ($\mathrm{d}Q/\mathrm{d}d_c$).

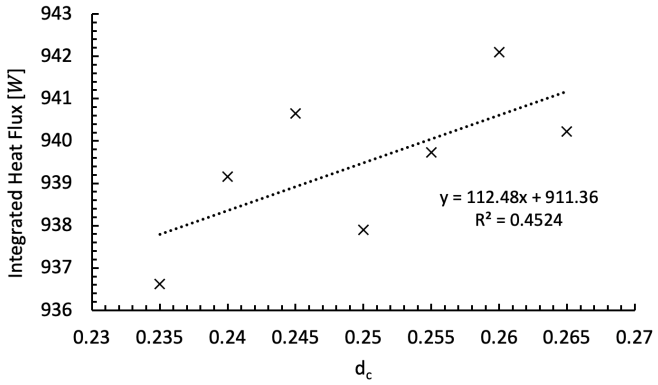


Fig. 9 Finite-difference results for d_c around 0.25 m and linear fit for slope quantification.

A linear fit had to be used instead of directly taking central difference in finite-difference parametric sensitivity quantification due to insufficient convergence of the primal solution and mesh dependency from the deformation process. It should be noted that the authors recognize the order of 3 in RMS pressure residual reduction may not be adequate for adjoint calculation, potentially leading to imprecise surface sensitivity calculation. Further convergence of the primal solution may yield differences in adjoint surface sensitivity. Nevertheless, in the context of tool development, our aim is to demonstrate and validate the coupling of geometric and adjoint flow sensitivities using this test case as a crawl-walk-run approach, laying the foundation for future improvements.

To quantify mesh dependency uncertainty, we compared the integrated heat flux using multiple meshes generated for a geometry while perturbing the mesh generation parameters by a finite amount. Specifically, we perturbed the average node spacing, denoted by Δ, from baseline values to $\Delta \pm 0.01\Delta$ and $\Delta \pm 0.05\Delta$. All generated meshes satisfied an average y^+ below 0.2. The resultant standard deviation for integrated heat flux is 12.306 W, and for parametric sensitivity is 9.972 W/m.

Although a mesh refinement study was carried out for the un-deformed mesh, changing a geometric parameter required regenerating the mesh under the current workflow. For the smallest design parameter perturbation of $d_c = d_c \pm 0.005$ m, the arc length of the fin nose is extended by $0.03d_c$ or 0.0079 m, which is on the order of the average mesh size used to discretize the nose and is larger than boundary layer spacing, which is $0.0004d_c$ or 0.0001 m. Consequently, regenerating the mesh on the deformed shape would change the connectivity of mesh nodes, which is not ideal because maintaining the same grid connectivity is considered to be of paramount importance for sensitivity calculation by multiple researchers [58, 59]. To remedy this problem, one could either use a mesh deformation solver to deform the mesh for parametric geometry changes or keep the geometry changes small enough that the mesh connectivity is minimally affected.

C. Finite Wing

1. Description

As a second test case, a rectangular wing with a NACA0012 airfoil is tested. This test case is one of the benchmark test cases in ADODG [15], with the purpose of testing optimizer performance in MDO. We selected this test case to demonstrate obtaining geometric sensitivity using SU2's Euler solver.

The characteristic length is the chord length c shown in Fig. 10 (a), and the span of the wing is $3c$. The origin of the coordinate system is located at the leading edge at the root of the wing, with the positive x-axis extending toward the trailing edge and the positive y-axis extending toward the tip. The twist angle θ is defined as the angle of twist with respect to the un-deformed airfoil and is positive in the clockwise direction. Two design parameters θ_r and θ_t are defined, corresponding with twist angle at root ($y = 0$) and twist angle at tip ($y = 3c$). Fig. 10 (b) shows the computational domain and coordinate system used in this study. We used a C-type domain with the far-field located $25c$ from the leading edge, $25c$ in the direction of the wing tip, and $50c$ in the downstream direction.

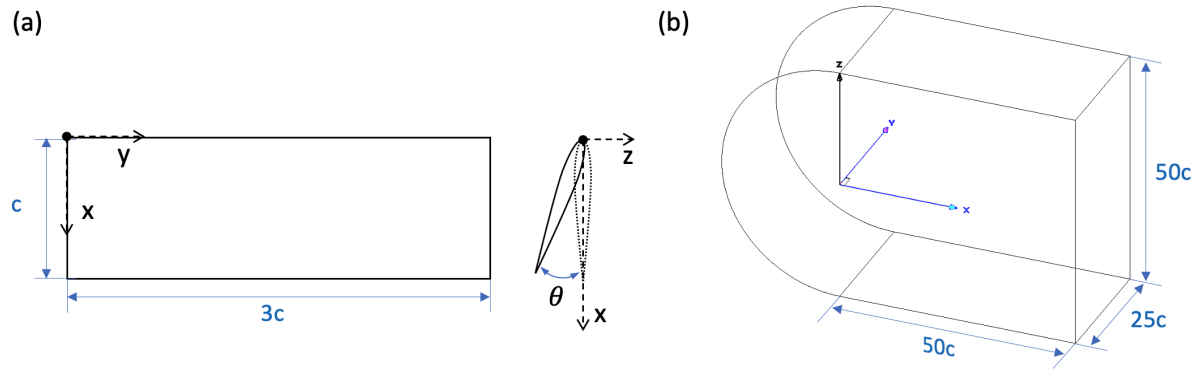


Fig. 10 Schematic drawing for (a) rectangular wing viewing from $x - y$ plane and $x - z$ plane, with twist angle θ defined, and (b) fluid domain used.

2. Validation

A `Pointwise`® [49] Glyph script is utilized to generate a fully unstructured mesh by defining meshing parameters, such as average edge and face spacing, in `pyCAPS`. The `Pointwise`® AIM tool is employed to facilitate this process. To evaluate the quality of the mesh, a grid refinement study is conducted based on the convergence of C_l and C_d. Mesh convergence is illustrated in Fig. 11. Based on the study, it is determined that a wing surface grid consisting of 8175 elements, with an average element size of $0.06c$, is sufficient for converging C_d to 2 significant digits.

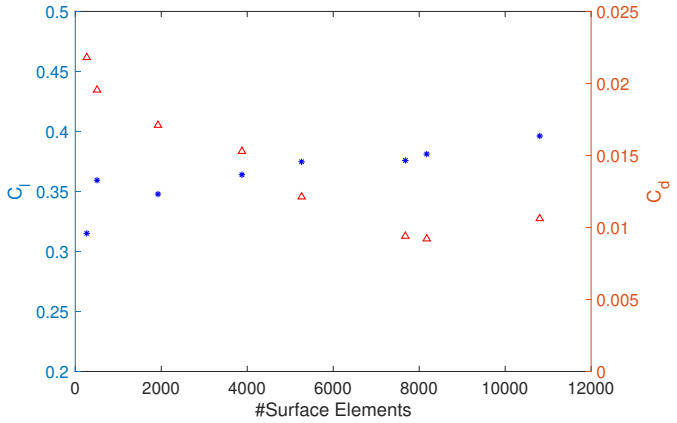


Fig. 11 Mesh independent study for test case 2, with C_l plotted on the left axis and C_d plotted on the right axis.

The far-field Mach number is set to $M_\infty = 0.3$, with a free-stream pressure of 101 300 Pa and a free-stream temperature of 288.15 K. To ensure $C_l \geq 0.375$, the angle of attack is set to 4.3 degrees. The $x - z$ plane is set to be a symmetry boundary condition, and the wing surface is set with an Euler wall boundary condition. The remaining boundaries are set to be far-field boundary conditions. Asymptotic convergence to a steady-state solution is obtained for all cases. A linear convergence for the primal and adjoint density residuals is illustrated in Fig. 12. The convergence criterion for the simulation is an RMS density or adjoint density residual below -8 based on a similar justifications as the Heat Exchanger section. The pressure coefficient contour for the baseline un-deformed wing ($\theta_r = \theta_t = 0$ deg) at centerline at $y = 1.5c$ is shown in Fig. 13 to illustrate the converged pressure distribution.

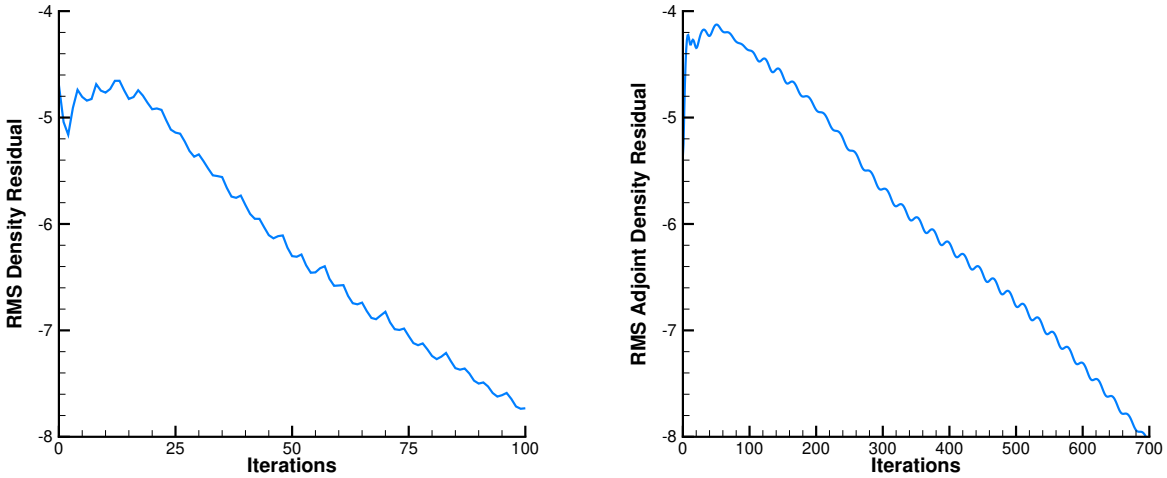


Fig. 12 Convergence history of root-mean-square primal density and adjoint density residuals.

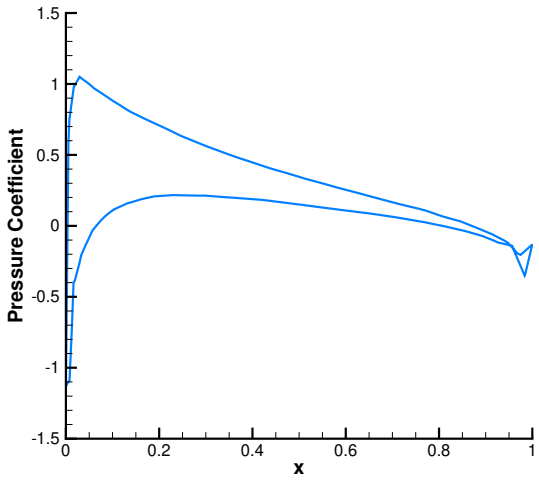


Fig. 13 Pressure coefficient for un-deformed wing.

The sensitivity of the drag coefficient (C_d) to the tip twist angle (θ_t) is computed using an analytic adjoint workflow and compared against the result obtained using finite-difference. The finite-difference result is generated by perturbing θ_t by a finite degree, performing primal CFD computation until the same pressure residual convergence is achieved, calculating the drag coefficient C_d, and using a central difference scheme to compute the sensitivity $\mathrm{d}C_d/\mathrm{d}\theta_t$. Fig. 14 shows the predicted sensitivity at various step sizes $\Delta\theta_t$, which are defined as finite perturbations applied to the tip twist angle. From this step sizing study, it is observed that for $\Delta\theta_t \leq 0.1$ deg, the step size is too small, as the prediction error has a log-linear correlation with the step size, indicating that the source of error originates from numerical subtractions. The predicted sensitivity is of the same order for step sizes between 0.2 deg and 1.2 deg. The sensitivity prediction using the smallest possible step size, corresponding to $\Delta\theta_t = 0.2$ deg, is selected as the finite-difference reference value for validating the parametric sensitivity computed with the current analytic adjoint workflow. The finite-difference and the computed analytic adjoint with CAD parameter sensitivity $\mathrm{d}C_d/\mathrm{d}\theta_t$ are reported in Table 2. The relative error between the finite-difference and analytic sensitivities is 3.68%.

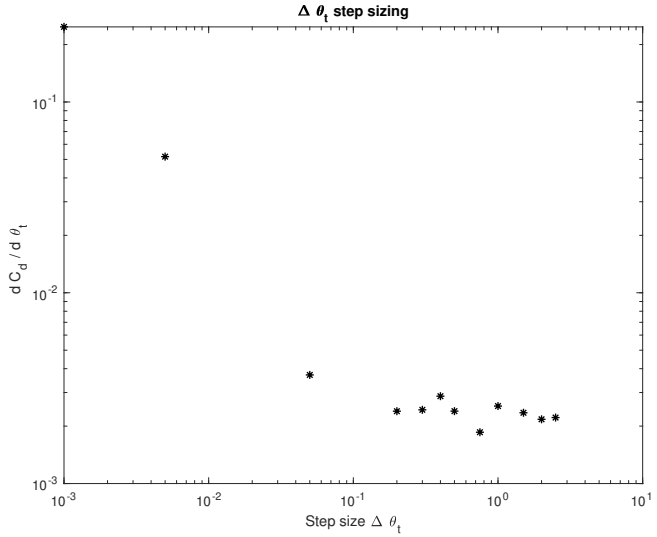


Fig. 14 Step sizing on finite-difference step size $\Delta\theta_t$ versus predicted sensitivity $\mathrm{d}C_d/\mathrm{d}\theta_t$.

θ_t	C_l	C_d	CAD $\mathrm{d}C_d/\mathrm{d}\theta_t$	FD $\mathrm{d}C_d/\mathrm{d}\theta_t$	Relative difference [%]
0	0.377	0.00944	0.00223	0.00232	3.68

Table 2 Sensitivity of C_d to tip twist angle θ_t ($\mathrm{d}C_d/\mathrm{d}\theta_t$), computed with current workflow (CAD) and with finite-difference (FD).

The improvement in relative error between the sensitivity computed by the finite-difference method and the workflow in the previous test case is noteworthy. The authors believe that the improved mesh quality on the geometry surface and the relatively smaller geometric deformation, as compared to the minimal mesh size, contributed to this improvement. Regarding the first claim, the mesh used in this test case had finely refined elements with an average cell size of $0.01c$ near the tip of the wing, where deformation is the largest. When θ_t is perturbed by 0.2 deg, it induced a maximum displacement of $0.003c$ at the trailing edge of the wing at the tip. However, this deformation is still smaller than the cell size used on the edge. Hence, the induced error across the simulations of mesh regeneration is considered to be smaller than that reported in the previous test case. This improvement supports our hypothesis regarding mesh-regeneration induced error hypothesis in the previous test case. Further investigation, however, shall be conducted on converging both the primal and adjoint residual further to reduce uncertainty in adjoint flow sensitivity.

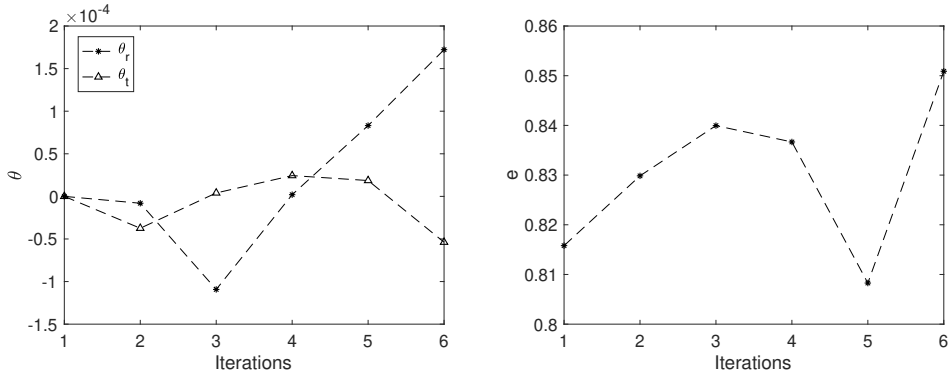
3. Optimization

As discussed in the Introduction section, the purpose of developing this workflow is to use the computed parametric sensitivity to drive shape optimization through a gradient-based optimizer. Despite the limitations demonstrated in previous heat exchanger and finite wing test cases, it remains unclear whether the computed sensitivities, accounting for the uncertainty induced in the mesh regeneration step during each update of design variables, can be used to drive gradient-based design optimization. To test this capability, we perform a shape optimization test case using a gradient descent optimizer to serve as a preliminary assessment of the stability of the gradient computed during shape optimization iterations.

The objective of this test case is to minimize C_d under the constraint that $C_l \geq 0.375$. Both design parameters θ_r and θ_t are allowed to vary.

$$\begin{aligned} \underset{\theta_t,\theta_r}{\text{minimize}} \quad & C_d(\theta_t, \theta_r) \\ \text{subject to} \quad & C_l(\theta_t, \theta_r) \geq 0.375 \end{aligned}$$

To optimize the design variables, we employed a fixed-step gradient descent optimizer, which updates the variable vector at each iteration according to $[\theta_r,\ \theta_t]^{i+1} = [\theta_r,\ \theta_t]^i - \alpha g([\theta_r,\ \theta_t]^i)$, where g denotes the parametric gradient and α represents the learning rate. We imposed the constraint $C_l \geq 0.375$ by defining the objective function as $I = C_d + \max((0.375 - C_l)^2,\ 0)$. We set the learning rate α to 0.02 to ensure the stability of the optimizer. The choice of α is to ensure the stability of the optimizer and is not optimized in this study.



-0

Fig. 15 Twist optimization for finite wing with C_l and C_d evolution along with elliptical lift efficiency factor e

The results for the optimization for 6 iterations are shown in Fig. 15. The non-monotonic behavior in the twist angle evolution, as indicated by a decrease in θ_r at iteration 3, is caused by occasional parametric sensitivity divergence, most likely induced by mesh regeneration errors. However, as an overall trend, the wing tip is twisted towards reducing the angle of attack, while the root is twisted to increase the angle of attack to satisfy the lift constraint. The right portion of Fig. 15 shows the evolution of the elliptical lift efficiency factor, defined as $e = C_l^2/(\pi ARC_d)$, where AR is the aspect ratio of the wing, which is 6 for this test case. The results show an increasing elliptical lift efficiency factor after the wing is optimized, indicating that drag reduction is achieved after 6 iterations. Further improvement should focus on improving the mesh regeneration routine between iterations or using parametric surface velocity to deform the original mesh to improve the stability of parametric sensitivity computation.

V. Conclusion

This study presents a workflow for computing the parametric sensitivity of objective functionals to CAD design parameters using the `pyCAPS` framework and `SU2` multi-physics solver. The goal is to provide an effective tool for industry to explore design space while taking manufacturing constraints into account through the use of a CAD software. The workflow begins by obtaining geometric sensitivity using `OpenCSM`, which is then validated against analytical results for a simple 3-dimensional block. The geometric sensitivity is then combined with adjoint surface sensitivity obtained from the `SU2` adjoint solver through a dot product, resulting in a parametric sensitivity. Two test cases are considered, one using `SU2`'s incompressible Navier-Stokes solver and the other using `SU2`'s Euler solver, to validate the computed analytic parametric sensitivities with finite-difference results. The results demonstrate reasonable agreement, indicating the conceptual effectiveness of the proposed workflow.

To demonstrate the potential for using the computed parametric sensitivities to drive design space exploration, the parametric sensitivity computed using this workflow is then applied to a finite wing shape optimization problem. In addition, the study also explored the effects of regenerating the mesh during optimization iterations. The authors believe that mesh regeneration between updates of design parameters can result in inconsistencies in physical problem modeling, due to both discretization and the changes in mesh connectivity. Therefore, for future work, mesh deformation is preferred for design parameter changes when the deformation is small. This can be achieved by exporting the surface velocity induced by changing design parameters using `OpenCSM` and using this velocity to drive a mesh deformation solver.

VI. Acknowledgements

We would like to express our sincerest gratitude to the following companies and individuals for their invaluable assistance. This work has been made possible by the generous support of Robert Bosch LLC. Additionally, the authors would like to extend special thanks to Dr. John F. Dannenhoffer from Syracuse University for his insightful discussions with `EGADS`, `OpenCSM`, and `CAPS`.

References

[1] Martins, J. R., "A Coupled-Adjoint Method for High-fidelity Aero-structural Optimization," Ph.D. thesis, Stanford University, 2002.

[2] Kenway, G. K., and Martins, J. R., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," Journal of Aircraft, Vol. 51, No. 1, 2014, pp. 144–160. https://doi.org/10.2514/1.C032150.

[3] Brooks, T. R., Kenway, G. K., and Martins, J. R., "Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings," AIAA Journal, Vol. 56, No. 7, 2018, pp. 2840–2855. https://doi.org/10.2514/1.J056603.

[4] Bons, N., Mader, C. A., Martins, J. R., Cuco, A., and Odaguil, F., "High-fidelity aerodynamic shape optimization of a full configuration regional jet," 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2018, p. 0106. https://doi.org/10.2514/6.2018-0106.

[5] Wang, L., Diskin, B., Biedron, R. T., Nielsen, E. J., and Bauchau, O. A., "High-Fidelity Multidisciplinary Sensitivity Analysis and Design Optimization for Rotorcraft Applications," AIAA Journal, Vol. 57, No. 8, 2019, pp. 3117–3131. https://doi.org/10.2514/1.J056587.

[6] Crovato, A., Almeida, H. S., Vio, G., Silva, G. H., Prado, A. P., Breviglieri, C., Güner, H., Cabral, P. H., Boman, R., Terrapon, V. E., and Dimitriadis, G., "Effect of Levels of Fidelity on Steady Aerodynamic and Static Aeroelastic Computations," Aerospace, Vol. 7, No. 4, 2020, pp. 1–22. https://doi.org/10.3390/aerospace7040042.

[7] Gray, A. C., and Martins, J. R., "Geometrically Nonlinear High-fidelity Aerostructural Optimization for Highly Flexible Wings," AIAA Scitech 2021 Forum, , No. January, 2021, pp. 1–24. https://doi.org/10.2514/6.2021-0283.

[8] Nadarajah, S. K., and Jameson, A., "A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization," 38th Aerospace Sciences Meeting and Exhibit, 2000. https://doi.org/10.2514/6.2000-667.

[9] Martins, J. R., Alonso, J. J., and Reuther, J. J., "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," Optimization and Engineering, Vol. 6, No. 1, 2005, pp. 33–62. https://doi.org/10.1023/B:OPTE.0000048536.47956.62.

[10] 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, pp. 828–846. https://doi.org/10.2514/1.J053813.

[11] Venkatesan-Crome, C., and Palacios, R., "A Discrete Adjoint Solver for Time-Domain Fluid-Structure Interaction Problems with Large Deformations," AIAA Scitech 2020 Forum, 2020. https://doi.org/10.2514/6.2020-0406.

[12] Albring, T. A., Sagebaum, M., and Gauger, N. R., "Efficient Aerodynamic Design using the Discrete Adjoint Method in SU2," AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, 2016. https://doi.org/10.2514/6.2016-3518, 0.

[13] Giles, M. B., and Pierce, N. A., "An Introduction to the Adjoint Approach to Design," Flow, Turbulence and Combustion, Vol. 65, No. 3, 2000, pp. 393–415. https://doi.org/10.1023/A:1011430410075.

[14] Giles, M. B., Duta, M. C., Müller, J. D., and Pierce, N. A., "Algorithm developments for discrete adjoint methods," AIAA Journal, Vol. 41, No. 2, 2003, pp. 198–205. https://doi.org/10.2514/2.1961.

[15] Yang, G., Da Ronch, A., and Drofelnik, J., "Wing Twist Optimisation of a Benchmark Problem Using Two Aerodynamic Solvers," 31st Congress of the International Council of the Aeronautical Sciences (ICAS 2018), 2018, pp. 1–8. URL http://eprints.soton.ac.uk/id/eprint/423722.

[16] Gamba, E. P., and Palharini, R. C., "Design Method for Wind Turbine Airfoils by Adjoint Optimization Considering Climate Conditions." AIAA Scitech 2021 Forum, 2021. https://doi.org/10.2514/6.2021-0464.

[17] Zhou, B. Y., Ryong Koh, S., Gauger, N. R., Meinke, M., and Schöder, W., "A discrete adjoint framework for trailing-edge noise minimization via porous material," Computers & Fluids, Vol. 172, 2018, pp. 97–108. https://doi.org/https://doi.org/10.1016/j.compfluid.2018.06.017.

[18] Masters, D. A., Taylor, N. J., Rendall, T. C., Allen, C. B., and Poole, D. J., "A geometric comparison of aerofoil shape parameterisation methods," 54th AIAA Aerospace Sciences Meeting, Vol. 0, No. January, 2016, pp. 1–36. https://doi.org/10.2514/6.2016-0558.

[19] Sobieczky, H., "Geometry Generator for CFD and Applied Aerodynamics," New Design Concepts for High Speed Air Transport, edited by H. Sobieczky, Springer Vienna, Vienna, 1997, pp. 137–157. https://doi.org/10.1007/978-3-7091-2658-5_9.

[20] Braibant, V., and Fleury, C., "Shape optimal design using B-splines," Computer Methods in Applied Mechanics and Engineering, Vol. 44, No. 3, 1984, pp. 247–267. https://doi.org/10.1016/0045-7825(84)90132-4.

[21] Svenningsen, K. H., Madsen, J. I., Hassing, N. H., and Päuker, W. H., "Optimization of flow geometries applying quasianalytical sensitivity analysis," Applied Mathematical Modelling, Vol. 20, No. 3, 1996, pp. 214–224. https://doi.org/10.1016/0307-904X(95)00148-D.

[22] Kulfan, B., and Bussoletti, J., ""Fundamental" Parameteric Geometry Representations for Aircraft Component Shapes," 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2006. https://doi.org/10.2514/6.2006-6948.

[23] Hinchliffe, B., and Qin, N., "Using surface sensitivity from mesh adjoint solution for transonic wing drag reduction," AIAA Scitech 2016 Forum, San Diego, 2016. https://doi.org/10.2514/6.2016-0560.

[24] Bobrowski, K., Ferrer, E., Valero, E., and Barnewitz, H., "CAD-based aerodynamic shape optimization using geometry surrogate model and adjoint methods," 17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, , No. June, 2016. https://doi.org/10.2514/6.2016-3831.

[25] Jameson, A., "Aerodynamic design via control theory," Journal of Scientific Computing, Vol. 3, No. 3, 1988, pp. 233–260. https://doi.org/10.1007/BF01061285.

[26] Hicks, R. M., and Henne, P. A., "Wing Design by Numerical Optimization," Journal of Aircraft, Vol. 15, No. 7, 1978, pp. 407–412. https://doi.org/10.2514/3.58379.

[27] He, X., Li, J., Mader, C. A., Yildirim, A., and Martins, J. R., "Robust aerodynamic shape optimization—From a circle to an airfoil," Aerospace Science and Technology, Vol. 87, 2019, pp. 48–61. https://doi.org/10.1016/j.ast.2019.01.051.

[28] Dhert, T., Ashuri, T., and Martins, J. R., "Aerodynamic shape optimization of wind turbine blades using a Reynolds-averaged Navier–Stokes model and an adjoint method," Wind Energy, Vol. 20, No. 5, 2017, pp. 909–926. https://doi.org/10.1002/we.2070.

[29] Nagawkar, J., Ren, J., Du, X., Leifsson, L., and Koziel, S., "Single- and Multipoint Aerodynamic Shape Optimization Using Multifidelity Models and Manifold Mapping," Journal of Aircraft, Vol. 58, No. 3, 2021, pp. 591–608. https://doi.org/10.2514/1.c035297.

[30] Monfaredi, M., Trompoukis, X., Tsiakas, K., and Giannakoglou, K., "Unsteady continuous adjoint to URANS coupled with FW-H analogy for aeroacoustic shape optimization," Computers and Fluids, Vol. 230, No. April, 2021, p. 105136. https://doi.org/10.1016/j.compfluid.2021.105136.

[31] Xu, S., Jahn, W., and Müller, J., "CAD-based shape optimisation with CFD using a discrete adjoint," International Journal for Numerical Methods in Fluids, Vol. 74, 2014, pp. 153–168. https://doi.org/10.1002/fld.3844.

[32] Wu, N., Mader, C. A., and Martins, J. R., "Sensitivity-based Geometric Parameterization for Aerodynamic Shape Optimization," AIAA AVIATION 2022 Forum, 2022, pp. 1–17. https://doi.org/10.2514/6.2022-3931.

[33] Yu, Y., Lyu, Z., Xu, Z., and Martins, J. R., "On the influence of optimization algorithm and initial design on wing aerodynamic shape optimization," Aerospace Science and Technology, Vol. 75, 2018, pp. 183–199. https://doi.org/10.1016/j.ast.2018.01.016.

[34] Han, X., and Zingg, D. W., "An adaptive geometry parametrization for aerodynamic shape optimization," Optimization and Engineering, Vol. 15, No. 1, 2014, pp. 69–91. https://doi.org/10.1007/s11081-013-9213-y.

[35] Taylor, N. J., "Industrial Perspectives on Geometry Handling for Aerodynamics," American Institute of Aeronautics and Astronautics, 2015. https://doi.org/10.2514/6.2015-3408.

[36] Chawner, J. R., Dannenhoffer, J. F., and Taylor, N. J., "Geometry, mesh generation, and the CFD 2030 vision," 46th AIAA Fluid Dynamics Conference, , No. June, 2016, pp. 1–17. https://doi.org/10.2514/6.2016-3485.

[37] Sun, L., Yao, W., Robinson, T., Marques, S., and Armstrong, C., "A Framework of gradient-based shape optimization using feature-based CAD parameterization," AIAA Scitech 2020 Forum, 2020. https://doi.org/10.2514/6.2020-0889.

[38] Samareh, J., "A survey of shape parameterization techniques," NASA Conference Publication, , No. June, 1999, pp. 333–343. URL https://ntrs.nasa.gov/citations/19990050940.

[39] Robinson, T. T., Armstrong, C. G., Chua, H. S., Othmer, C., and Grahs, T., "Optimizing Parameterized CAD Geometries Using Sensitivities Based on Adjoint Functions," Computer-Aided Design and Applications, Vol. 9, No. 3, 2012, pp. 253–268. https://doi.org/10.3722/cadaps.2012.253-268.

[40] Agarwal, D., Robinson, T. T., Armstrong, C. G., Marques, S., Vasilopoulos, I., and Meyer, M., "Parametric design velocity computation for CAD-based design optimization using adjoint methods," Engineering with Computers, Vol. 34, No. 2, 2018, pp. 225–239. https://doi.org/10.1007/s00366-017-0534-x.

[41] Hajdik, H., Yildirim, A., and Martins, J. R., "Aerodynamic Shape Optimization with CAD-based Geometric Parameterization," AIAA Scitech 2023 Forum, 2023. https://doi.org/10.2514/6.2023-0726.

[42] Alyanak, E., Durscher, R., Haimes, R., Dannenhoffer, J. F., Bhagat, N., and Allison, D., "Multi-fidelity geometry-centric multi-disciplinary analysis for design," AIAA Modeling and Simulation Technologies Conference, 2016, , No. June, 2016, pp. 1–24. https://doi.org/10.2514/6.2016-4007.

[43] Haimes, R., Dannenhoffer, J., Bhagat, N. D., and Allison, D. L., "Multi-fidelity Geometry-centric Multi-disciplinary Analysis for Design," AIAA Modeling and Simulation Technologies Conference, 2016. https://doi.org/10.2514/6.2016-4007.

[44] 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, 2013. https://doi.org/10.2514/6.2013-3073.

[45] Dannenhoffer, J., and Haimes, R., "Design Sensitivity Calculations Directly on CAD-based Geometry," 53rd AIAA Aerospace Sciences Meeting, 2015. https://doi.org/10.2514/6.2015-1370.

[46] Haimes, R., and Drela, M., "On the construction of aircraft conceptual geometry for high-fidelity analysis and design," 50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2012, pp. 1–21. https://doi.org/10.2514/6.2012-683.

[47] "Open cascade, part of Capgemini," , retrieved 26 April 2023. URL https://www.opencascade.com/.

[48] Dannenhoffer, J., OpenCSM: An Open-Source Constructive Solid Modeler for MDAO, 2013. https://doi.org/10.2514/6.2013-701.

[49] "Pointwise, Cadence Design Systems," , retrived 26 April 2023. URL http://www.pointwise.com/.

[50] Durscher, R. J., and Reedy, D., "pyCAPS: A Python Interface to the Computational Aircraft Prototype Syntheses," AIAA Scitech 2019 Forum, 2019. https://doi.org/10.2514/6.2019-2226.

[51] Palacios, F., Alonso, J., Duraisamy, K., Colonno, M., Hicken, J., Aranake, A., Campos, A., Copeland, S., Economon, T., Lonkar, A., Lukaczyk, T., and Taylor, T., "Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design," 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, 2013. https://doi.org/10.2514/6.2013-287.

[52] Palacios, F., Economon, T. D., Aranake, A., Copeland, S. R., Lonkar, A. K., Lukaczyk, T. W., Manosalvas, D. E., Naik, K. R., Padron, S., Tracey, B., Variyar, A., and Alonso, J. J., "Stanford University Unstructured (SU2): Analysis and Design Technology for Turbulent Flows," 52nd Aerospace Sciences Meeting, 2014. https://doi.org/10.2514/6.2014-0243.

[53] 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, pp. 828–846. https://doi.org/10.2514/1.J053813.

[54] Jameson, A., Schmidt, W., and Turkel, E. L. I., "Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes," 14th Fluid and Plasma Dynamics Conference, 1981. https://doi.org/10.2514/6.1981-1259.

[55] Si, H., "TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator," ACM Trans. Math. Softw., Vol. 41, No. 2, 2015. https://doi.org/10.1145/2629697, URL https://doi.org/10.1145/2629697.

[56] University, M. S., "SimSys Software — simcenter.msstate.edu," https://www.simcenter.msstate.edu/software/documentation/system/index.html, . [Accessed 09-May-2023].

[57] Patel, H. C., Gastaldi, A. A., Breitsamter, C., Daoud, F., and Alonso, J. J., "Aero-structural discrete adjoint sensitivities in su2 using algorithmic differentiation," AIAA AVIATION 2022 Forum, 2022, pp. 1–19. https://doi.org/10.2514/6.2022-3358.

[58] Truong, A. H., Oldfield, C. A., and Zingg, D. W., "Mesh Movement for a Discrete-Adjoint Newton-Krylov Algorithm for Aerodynamic Optimization," AIAA Journal, Vol. 46, No. 7, 2008, pp. 1695–1704. https://doi.org/10.2514/1.33836.

[59] Samareh, J., "Application of Quaternions for Mesh Deformation," 2002.