Highlights

Adjoint-Based Aerodynamic Shape Optimization with Hybridized Discontinuous Galerkin Methods

Joachim Balis, Frederik Jacobs, Georg May

- First shape optimization tool in a high-order HDG finite element framework
- Gradient computed through discrete adjoint and finite difference
- 2D mesh motion interpolated from the body surface displacement
- In-house steepest descent or external optimization algorithms from pyOpt
- Coupling possible with various physical models, notably high-enthalpy

Adjoint-Based Aerodynamic Shape Optimization with Hybridized Discontinuous Galerkin Methods

Joachim Balis[a,*], Frederik Jacobs[a], Georg May[a]

[a] *Von Karman Institute for Fluid Dynamics, Waterloosesteenweg 72, 1640, Sint-Genesius-Rode, Belgium*

Abstract

We present a discrete adjoint approach to aerodynamic shape optimization (ASO) based on a hybridized discontinuous Galerkin (HDG) discretization. Our implementation is designed to tie in as seamlessly as possible into a solver architecture written for general balance laws, thus adding design capability to a tool with a wide range of applicability. Design variables are introduced on designated surfaces using the knots of a spline-based geometry representation, while gradients are computed from the adjoint solution using a difference approximation of residual perturbations. A suitable optimization algorithm, such as an in-house steepest descent or the Preconditioned Sequential Quadratic Programming (PSQP) approach from the pyOpt framework, is then employed to find an improved geometry. The resulting ASO module is currently set up for 2D test cases governed by balance laws, including linear scalar equations or nonlinear systems of equations. We present verification of the implementation, including drag or heat flux minimization

*Corresponding author

Email addresses: joachim.balis@aeronomie.be (Joachim Balis), frederik.jacobs@live.be (Frederik Jacobs), georg.may@vki.ac.be (Georg May)

in compressible flows, as well as inverse design.

Keywords: Aerodynamic shape optimization, Hybridized Discontinuous Galerkin, Discrete adjoint, Rational splines parameterization, Inverse design, Compressible flows

---

1. Introduction

Aerodynamic shape optimization (ASO) has become increasingly important over recent years and is now used for the design of many different technological products such as aircraft, trains, cars, turbomachinery or pipes [1, 2]. In ASO, a geometry to be improved is parametrized through a set of design variables. A target objective (also called cost function) to be optimized is subsequently chosen, and the fluid flow equations are expressed as a constraint linking the target objective to the design variables [3]. Within the context of gradient-based optimization, the gradient is then defined as the derivative of the target objective with respect to the design variables. The gradient computation represents a critical part of the optimization process: it can be particularly costly to evaluate if the number of design variables is high and it can also lead to bad geometries if it is not computed accurately enough [4]. The adjoint approach, as first proposed by Pironneau [5], adequately combines low computational cost and high accuracy. In fact, its cost is virtually independent of the number of design variables.

In parallel, promising results in flow simulation have been obtained with high-order numerical methods [6, 7]. These methods potentially achieve higher accuracy per computational degree of freedom (DOF), compared to their low-order counterparts. This prospect of increased computational effi-

ciency has led to an increased research effort towards further improvement and wider application of high-order schemes. A well-known high-order finite element (FE) technique is the discontinuous Galerkin (DG) method, firstly introduced by Reed and Hill [8]. Several extensions of the DG method have been developed, most notably the hybridized discontinuous Galerkin (HDG) variant [9], which we use in our approach.

However, only a handful of research efforts implementing ASO in a high-order DG framework are reported in the literature: a discrete adjoint formulation in a DG framework for time-dependent flow problems has notably been developed, with applications in aeroacoustic noise minimization [10]. More recently, a unified strategy to perform shape optimization in both conforming FE and DG contexts, using a certified descent algorithm [11], has been proposed. Finally, an adjoint-based ASO tool has been developed for airfoil drag minimization within an isogeometric DG framework [12]. No pairing of adjoint-based ASO with HDG schemes are known to the authors.

The tool that will be used throughout this project has been developed by Woopen, Balan, and May [13]. It includes an HDG solver, which has been validated with respect to various compressible flow models such as Euler, Navier-Stokes and RANS. The framework also includes a discrete adjoint solver, currently used for goal-oriented anisotropic mesh refinement [14, 15]. It has already been shown that the HDG discretization of the primal problem is adjoint consistent once some simple rules are respected [16, 17]. The solver is written in C++, has an object-oriented structure, and is designed to be modular. The different components of the framework have independent implementations that are connected through general interfaces. This

decoupling makes the solver very flexible, and new features can be added without extensive modification of the existing code. Moreover, the HDG tool interfaces to Netgen/NGSolve, an open source FE library which handles geometries and provides a mesh generation tool [18]. The HDG solver also interfaces to the library `Mutation++` for high-enthalpy simulations [19]. An overview of the HDG framework structure is given in Fig. 1.

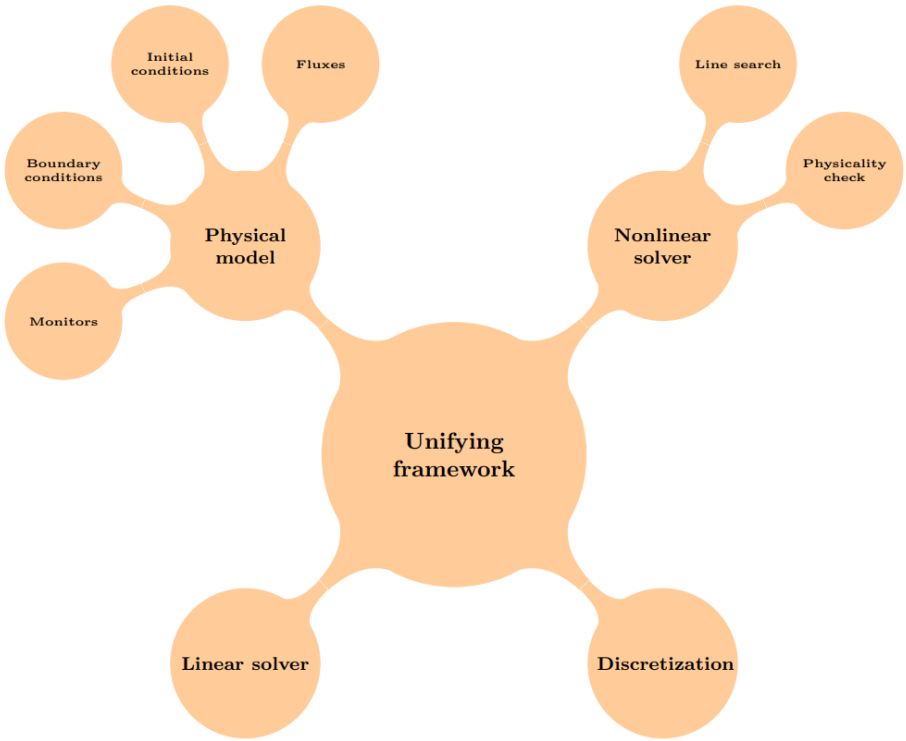

Figure 1: Structure of the HDG framework, adapted from [20]

Within this context, the present paper is based on a technical report presented at the AIAA SciTech Forum 2023 [21]. The objective of this work is actually threefold:

- Implement an efficient and modular method for computing the gradient, using the existing discrete adjoint solver.
- Develop a robust shape and mesh deformation strategy which will preserve the validity of the cells and the quality of the initial grid.
- Create and interface to optimizers able to perform automatically several optimization cycles, leading to an improved geometry.

In Section 2, the HDG approach to primal and adjoint discretization is described. In Section 3, a detailed discussion about our design choices in terms of shape parametrization and gradient computation is given. In Section 4, the inverse distance weighted procedure for the mesh deformation is described. In Section 5, we discuss the smoothed gradient descent approach [3] implemented within our code, as well as the coupling with the pyOpt package [22]. In Section 6, results on several test cases are presented, ranging from thermal and aerodynamic shape optimization to inverse design.

2. HDG Discretization

Among the different types of high-order schemes, DG methods have received considerable attention since their introduction by Reed and Hill in 1973 [8]. However, DG methods often lead to high computational cost due to the large number of DOF. Moreover, the global coupling of equations across elements interfaces leads to large stencils and hence significant memory requirements, especially with implicit methods. To alleviate these drawbacks, a significant effort has been directed towards the development of HDG schemes

in the past decade [9]. These schemes are based on hybridization, i.e., decoupling the system through the introduction of trace unknowns at the element interfaces. This idea is depicted in Fig. 2.

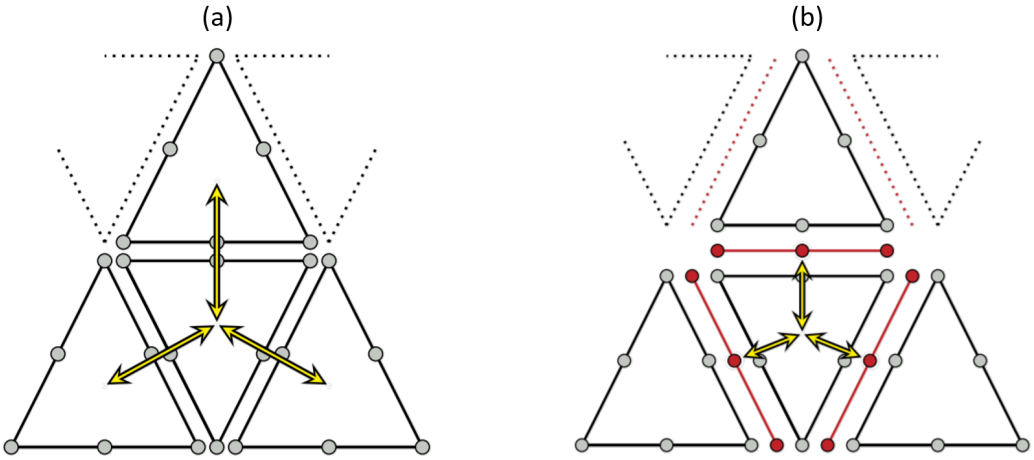

Figure 2: Comparison of DG (a) and HDG (b) methods. The triangles represent the elements, and the circles are the local unknowns. The red circles represent the trace. The yellow arrows symbolize the numerical fluxes for DG in (a) and the trace acting as boundary conditions for HDG in (b).

The elements are only indirectly coupled and the degrees of freedom corresponding to the elements can be efficiently eliminated from the global system by the Schur complement technique. As a result, the global system is much less expensive to solve, due to both a decrease in size and better sparsity.

In the next section, we summarize the HDG-discretized equations, similar to the account given by Woopen et al. [23]. The adjoint presentation given in [14] is slightly adapted and presented afterward. The general system of balance laws can be written as:

$$\nabla \cdot (\mathbf{f}_c(\mathbf{w}) - \mathbf{f}_v(\mathbf{w}, \nabla \mathbf{w})) = \mathbf{s}(\mathbf{w}, \nabla \mathbf{w}) \tag{1}$$

with given convective and diffusive fluxes,

$$\mathbf{f}_c : \mathbb{R}^m \to \mathbb{R}^{m\times d}, \quad \mathbf{f}_v : \mathbb{R}^m \times \mathbb{R}^{m\times d} \to \mathbb{R}^{m\times d}, \tag{2}$$

and a state-dependent source term,

$$\mathbf{s} : \mathbb{R}^m \times \mathbb{R}^{m\times d} \to \mathbb{R}^m, \tag{3}$$

where the spatial dimension is denoted by d and the number of conservative variables by m. For $\mathbf{f}_v \neq 0$, or a gradient-dependent source term $\mathbf{s}$, Eq. (1) can be rewritten as:

$$\begin{aligned} \mathbf{q} &= \nabla \mathbf{w}, \\ \nabla \cdot (\mathbf{f}_c(\mathbf{w}) - \mathbf{f}_v(\mathbf{w}, \mathbf{q})) &= \mathbf{s}(\mathbf{w}, \mathbf{q}). \end{aligned} \tag{4}$$

The mixed formulation (4) is frequently applied when motivating viscous discontinuous Galerkin discretization [24, 25].

2.1. Notation

The domain Ω is tessellated into a collection of non-overlapping elements, $\mathcal{T}_h = \{K\}$, such that $\bigcup_{K\in\mathcal{T}_h} K = \overline{\Omega}$. We define two sets on the element edges. They are, respectively, element-oriented and edge- or face-oriented:

$$\partial\mathcal{T}_h := \{\partial K \backslash \partial\Omega : K \in \mathcal{T}_h\}, \tag{5}$$

$$\Gamma_h := \{e : e = K \cap K' \text{ for } K, K' \in \mathcal{T}_h; \text{meas}_{d-1}(e) \neq 0\}. \tag{6}$$

Note that neither of these sets includes edges or faces lying on the domain boundary: the set of boundary edges is instead denoted by Γ_h^b.

Let $\Pi^p(\mathrm{D})$ be the set of polynomials of degree at most p on some domain D. Discontinuous function spaces are defined as:

$$V_h = \{v \in L^2(\Omega) : v|_K \in \Pi^{p_K}(K), \quad K \in \mathcal{T}_h\}^{m\times d}, \tag{7}$$

$$W_h = \{w \in L^2(\Omega) : w|_K \in \Pi^{p_K}(K), \quad K \in \mathcal{T}_h\}^{m}, \tag{8}$$

$$M_h = \{\mu \in L^2(\Gamma_h) : \mu|_e \in \Pi^{p_e}(e), \quad e \in \Gamma_h\}^{m}. \tag{9}$$

In the present work, $p_K = p$ for all $K \in \mathcal{T}_h$. Thus, $\mathbf{v} \in V_h, \mathbf{w} \in W_h$ and $\boldsymbol{\mu} \in M_h$ are piecewise polynomials of degree p, which are discontinuous across edges (for $\mathbf{v}$, $\mathbf{w}$) or vertices (for $\boldsymbol{\mu}$), respectively.

A distinction is made between element-oriented integration (defined with $\mathcal{T}_h$ or $\partial\mathcal{T}_h$),

$$\int_{\mathcal{T}_h} \cdot \; dx := \sum_{K\in\mathcal{T}_h} \int_K \cdot \; dx, \qquad \int_{\partial\mathcal{T}_h} \cdot \; ds := \sum_{K\in\mathcal{T}_h} \int_{\partial K\setminus\Gamma_h^b} \cdot \; ds,$$

and edge- or face-oriented integrals (defined with Γ_h),

$$\int_{\Gamma_h} \cdot \; ds := \sum_{e\in\Gamma_h} \int_e \cdot \; ds. \tag{10}$$

For boundary edges or faces, no such distinction is necessary, as each of these is on the boundary of only one element. We write boundary integrals as in (10) with Γ_h replaced by Γ_h^b.

2.2. Method

Defining $\mathbb{X}_h := V_h \times W_h \times M_h$, the HDG method is written as a variational equation

$$\mathbf{x}_h \in \mathbb{X}_h : \qquad N_h(\mathbf{x}_h; \mathbf{y}_h) = 0 \qquad \forall \mathbf{y}_h \in \mathbb{X}_h. \tag{11}$$

The semilinear form N_h is obtained from DG discretization of (4) on each element, using the trace DOF as a boundary condition, and enforcing conservation by continuity of the normal flux across elements. Setting $\mathbf{x}_h = (\mathbf{q}_h, \mathbf{w}_h, \boldsymbol{\lambda}_h)$ and $\mathbf{y}_h = (\boldsymbol{\tau}_h, \boldsymbol{\varphi}_h, \boldsymbol{\mu}_h)$, one obtains

$$\begin{aligned} N_h(\mathbf{x}_h; \mathbf{y}_h) := & \int_{\mathcal{T}_h} \boldsymbol{\tau}_h : \boldsymbol{q}_h \, dx + \int_{\mathcal{T}_h} (\mathbf{w}_h \otimes \nabla) : \boldsymbol{\tau}_h \, dx - \int_{\Gamma_h} (\boldsymbol{\lambda}_h \otimes \mathbf{n}) : \boldsymbol{\tau}_h \, ds \\ & - \int_{\mathcal{T}_h} \nabla \boldsymbol{\varphi}_h : (\mathbf{f}_c(\mathbf{w}_h) - \mathbf{f}_v(\mathbf{w}_h, \mathbf{q}_h)) \, dx - \int_{\mathcal{T}_h} \boldsymbol{\varphi}_h \cdot \mathbf{s}(\mathbf{w}_h, \mathbf{q}_h) \, dx \\ & + \int_{\partial \mathcal{T}_h} \boldsymbol{\varphi}_h \cdot (\widehat{\mathbf{f}}_c - \widehat{\mathbf{f}}_v) \, ds + \int_{\Gamma_h} \boldsymbol{\mu}_h \cdot [\![\hat{\mathbf{f}}_c - \hat{\mathbf{f}}_v]\!] \, ds \\ & + N_{h,\partial\Omega}(\mathbf{q}_h, \mathbf{w}_h; \boldsymbol{\tau}_h, \boldsymbol{\varphi}_h) + N_{h,\mathrm{sc}}(\mathbf{q}_h, \mathbf{w}_h; \boldsymbol{\varphi}_h) \end{aligned} \tag{12}$$

In the present work, the numerical (normal) convective flux $\hat{\mathbf{f}}_c$ is of the Lax-Friedrichs type, while a local discontinuous Galerkin scheme is employed for the diffusive flux $\hat{\mathbf{f}}_v$.

In order to obtain an adjoint-consistent scheme, the boundary terms have to be discretized properly. Following an approach proposed by Schütz and May in [16], the boundary terms are discretized using boundary states $\mathbf{w}_{\partial\Omega}(\mathbf{w}_h)$ and $\mathbf{q}_{h,\partial\Omega}(\mathbf{w}_h, \mathbf{q}_h)$:

$$\begin{aligned} N_{h,\partial\Omega}(\mathbf{q}_h, \mathbf{w}_h; \boldsymbol{\tau}_h, \boldsymbol{\varphi}_h) := & \int_{\Gamma_h^b} (\mathbf{w}_{\partial\Omega} \otimes \mathbf{n}) : \boldsymbol{\tau}_h \, ds \\ & + \int_{\Gamma_h^b} (\boldsymbol{\varphi}_h \otimes \mathbf{n}) : (\mathbf{f}_c(\mathbf{w}_{\partial\Omega}) - \mathbf{f}_v(\mathbf{w}_{\partial\Omega}, \mathbf{q}_{h,\partial\Omega})) \, ds. \end{aligned} \tag{13}$$

It is important to note that $\boldsymbol{\lambda}_h$ does not appear in the boundary terms, as it is only defined on interior faces. We omit discussion of the shock-capturing operator and refer instead to [20].

2.3. Adjoint system

The discrete adjoint system of equations is derived from the presented primal system. The adjoint may be used for computing gradients, notably in the context of mesh adaptation or shape optimization. The exact Jacobian of the nonlinear primal system is available from our Newton solver and, upon transposition, may be used directly to solve the adjoint problem. This is suitable, as our HDG discretization is adjoint consistent with respect to admissible target functionals [17]. However, in the present HDG framework, the adjoint is already used for goal-oriented anisotropic mesh refinement. In this context, the adjoint is solved in a richer space to avoid identically vanishing error estimates. For the sake of maximizing code re-usage, this approach has been taken for shape optimization as well.

More precisely, we prolongate a given primal solution $\mathbf{x}_h \in \mathbb{X}_h$ to the space $\widetilde{\mathbb{X}}_h$, obtained by incrementing all polynomial degrees by one. Then, for given admissible target functional $\mathcal{J}_h$, we solve the adjoint problem

$$\mathcal{N}_h' \left[\mathbf{x}_h\right] \left(\mathbf{y}_h, \boldsymbol{\psi}_h\right) = \mathcal{J}_h' \left[\mathbf{x}_h\right] \left(\mathbf{y}_h\right) \quad \forall \mathbf{y}_h \in \widetilde{\mathbb{X}}_h. \tag{14}$$

The adjoint solution $\boldsymbol{\psi}_h \in \tilde{\mathrm{X}}_h$ represents how residual variations lead to target functional variations.

For more information about the HDG solver, the reader is encouraged to read [13]. For further discussion about the adjoint and its applications in mesh adaptation, [17, 23] can be consulted. Here, we discuss the use of the adjoint solution in an ASO context, i.e., the computation of gradients with respect to the design variables.

3. Design Variables and Gradient Computation

Before implementing the required ASO routines, it is important to make a choice regarding shape parametrization. Netgen offers several geometry formats to represent 2D and 3D geometries, such as Constructive Solid Geometry (CSG), StereoLithography (STL), or 2D spline geometries [18]. In this work, the focus is directed toward curved 2D profiles described by piecewise rational splines.

Each rational spline built by Netgen is described by 3 control points: the two endpoints of the spline, denoted $\vec{p}_i = (x_i, y_i)$ and $\vec{p}_f = (x_f, y_f)$, and the point at which the tangents to the geometry at these two end points meet, denoted $\vec{p}_t = (x_t, y_t)$. This latter point will be called "mid point" in the rest of this section. The situation is represented for one spline in Fig. 3.

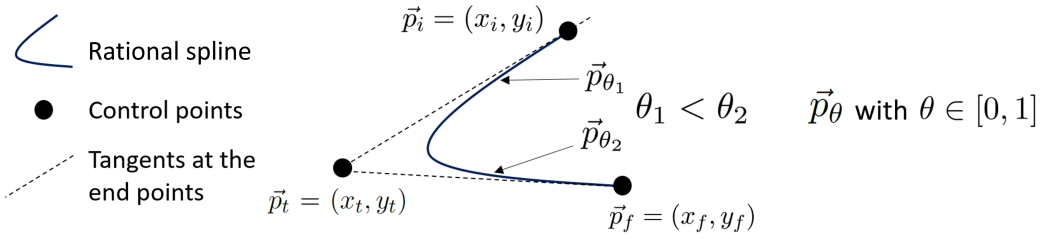

Figure 3: Rational spline represented by two end points and one mid point

With this body representation, each point $\vec{p}_\theta$ located on the geometry surface between $\vec{p}_i$ and $\vec{p}_f$ can be uniquely identified by a parameter θ varying between 0 and 1 through:

$$\vec{p}_\theta = \frac{((1-\theta)^2\vec{p}_i + w\theta(1-\theta)\vec{p}_t + \theta^2\vec{p}_f)}{(1-\theta)^2 + w\theta(1-\theta) + \theta^2}, \tag{15}$$

where

$$w = \frac{\|\vec{p}_i - \vec{p}_f\|}{\sqrt{0.5\left(\|\vec{p}_i - \vec{p}_t\|^2 + \|\vec{p}_f - \vec{p}_t\|^2\right)}}, \tag{16}$$

and $\|\cdot\|$ is the usual Euclidean distance between two points. Along with its simplicity, a significant advantage of such a geometry representation is that the spline endpoints can be directly chosen as design variables: they uniquely represent the geometry as the midpoint locations can be deduced from the knowledge of the endpoint positions and from smoothness constraints.

The gradient $\mathcal{G}$ represents the sensitivity of the target objective $\mathcal{J}$ with respect to the design variables $\mathcal{F}$, i.e., $\delta\mathcal{J} = \mathcal{G}\delta\mathcal{F}$. Since a discrete adjoint solver is already available in the HDG framework, the gradient can be evaluated using the adjoint solution:

$$\mathcal{G} = \left[\frac{\partial \mathcal{J}_h}{\partial \mathcal{F}}\right]^T - \psi^T \left[\frac{\partial R}{\partial \mathcal{F}}\right]. \tag{17}$$

Here ψ is the discrete adjoint coefficient vector and R contains the flow field residuals. The mathematical derivation of this this expression can be found in [3].

A modification of the body geometry (by changing the design variables) necessitates a corresponding modification of the mesh. When the mesh topology is perturbed, the volumes of the cells increase or decrease and the integral weights change. Moreover, the face normals are impacted and the quadrature points move along with the mesh vertices. These successive modifications can be summarized with the well-known chain rule of differentiation:

$$\frac{\partial}{\partial \mathcal{F}} = \frac{\partial}{\partial X_V} \cdot \frac{\partial X_V}{\partial X_S} \cdot \frac{\partial X_S}{\partial \mathcal{F}}, \tag{18}$$

where X_S are the surface vertices and X_V are the volume vertices. Hence, the gradient can be obtained by evaluating each partial derivative one by one and then by multiplying them.

In the context of this project, a significant implementation difficulty is that these derivatives concern the residuals and the target objective (which are evaluated in the HDG framework) with respect to the design variables, which represent the geometry defined in Netgen. Therefore, the method of automatic differentiation used inside the HDG framework, albeit attractive for its efficiency and exactness properties [20], was dismissed after a thorough analysis. It was deemed against our objective of modularity, raising obvious maintainability issues, to adapt routines inside external tools (Netgen in this case) to the needs of the design module. More recently, the Netgen library has provided a way to differentiate the geometry and mesh routines [26], which may be considered in the future.

With these considerations in mind, the method preferred for this project is to use the adjoint solution and to evaluate the partial derivatives with respect to the design variables of Eq. (17) with a finite difference scheme[1]. This approach gives the following expression for the differentiation with respect to the design variable m:

$$\frac{\delta \mathcal{J}_h}{\delta \mathcal{F}_m} = \frac{\mathcal{J}_h\left(\mathbf{x}, \boldsymbol{\alpha}^{(+\delta F_m)}\right) - \mathcal{J}_h(\mathbf{x}, \boldsymbol{\alpha})}{\delta F_m} - \psi^T \frac{R\left(\mathbf{x}, \boldsymbol{\alpha}^{(+\delta \mathcal{F}_m)}\right) - R(\mathbf{x}, \boldsymbol{\alpha})}{\delta F_m}. \quad (19)$$

Here we have written $\mathcal{J}_h = \mathcal{J}_h(\mathbf{x}, \boldsymbol{\alpha})$, where $\mathbf{x}$ is the HDG approximation of the flow field, see Section 2, $\boldsymbol{\alpha}$ represents the geometry and mesh topology,

[1]Complex step methods were rejected for similar reasons as automatic differentiation, as implementation would have required to change Netgen routines.

and δ is a small perturbation step.

It is important to note that the numerical solution $\mathbf{x}$ is not reevaluated for computing the gradient. The target objective and the residuals are evaluated on both initial and perturbed configurations with the same flow field. Therefore, despite the fact that the cost of this approach scales linearly with the number of design variables, it is still relatively cheap.

4. Mesh Deformation

Once the geometry is modified, the mesh needs to be adapted accordingly. Netgen offers the possibility to regenerate a mesh from an updated spline geometry. However, mesh regeneration algorithms can change the mesh topology and connectivity. This is an undesirable feature as it could impact the convergence of the optimization loop.

Therefore, a mesh deformation approach is preferred. For that purpose, the surface mesh needs to be adapted first. In fact, once the geometry is modified, the mesh vertices which belonged to the body must remain on the body. This projection can be achieved in two ways: the first is to perform a minimum distance projection which is already implemented in Netgen. The second possibility is to get the θ value of each surface mesh point and to keep it constant on the deformed configuration. Indeed, since the weight of the spline will be modified after the geometry perturbation, keeping constant θ value will nonetheless lead to new coordinates through Eq. (15). Both ways have been tested and compared: the differences appeared to be rather small for small geometry deformations. Consequently, the simplicity of the latter approach is preferred for this work.

Once the surface mesh has been projected onto the deformed geometry, the volume has to be modified accordingly. For that purpose, a direct interpolation-based method inspired from [27] and later reworked in [28] is employed. In that algorithm, each surface node displacement is represented as a rigid body motion, and the volume mesh is deformed as an inverse distance weighted average of the surface motion.

If the position of surface mesh point i in the initial configuration is noted $\mathbf{x}_{s,i}^{(0)}$, its position $\mathbf{x}_{s,i}$ in the deformed configuration can be written as:

$$\mathbf{x}_{s,i} = \mathbf{x}_{s,i}^{(0)} + \mathbf{s}_i, \tag{20}$$

where $\mathbf{s}_i$ defines the translation applied to this surface mesh point from the initial to the deformed configuration.

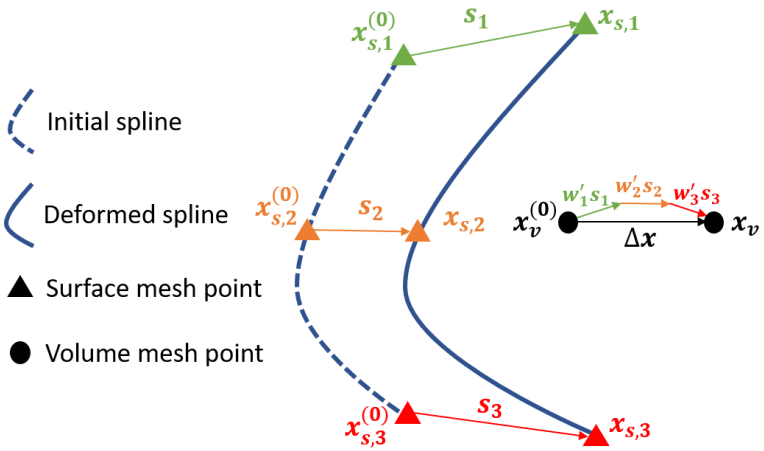

Figure 4: Volume mesh deformation as a weighted average of the surface mesh deformation, adapted from [28].

Then, each volume point motion is computed as a weighted average of the deformation of all surface mesh points, as illustrated in Fig. 4. Denoting $\mathbf{x_v}^{(0)}$

the position of the volume point in the initial configuration and $\mathbf{x_v}$ its position in the deformed configuration, the deformation $\Delta\mathbf{x_v}$ can be computed with:

$$\Delta\mathbf{x}\left(\mathbf{x_v}^{(0)}\right) = \mathbf{x_v} - \mathbf{x_v}^{(0)} = \frac{\sum_i w_{\mathrm{i}}\left(\mathbf{x_v}^{(0)}\right)\cdot\mathbf{s}_i}{\sum_i w_{\mathrm{i}}\left(\mathbf{x_v}^{(0)}\right)}. \tag{21}$$

The weights w_{i} used in this process are inversely proportional to the distance between the volume point and the surface point:

$$w_i\left(\mathbf{x_v}^{(0)}\right) = \left[\left(\frac{L_{\mathrm{def}}}{\left|\mathbf{x_v}^{(0)} - \mathbf{x}_{s,i}^{(0)}\right|}\right)^a + \left(\frac{\alpha L_{\mathrm{def}}}{\left|\mathbf{x_v}^{(0)} - \mathbf{x}_{s,i}^{(0)}\right|}\right)^b\right]. \tag{22}$$

The parameter L_{def} controls how far surface deformations propagate in the computational domain. It is automatically set to the maximum distance between the mesh centroid and the furthest surface mesh point. Finally, α, a, and b are tuned parameters that create a near-field body influence where nearby surface mesh points have a much more significant impact on the deformation process than distant mesh points. Following the guidelines of [27, 28], the following values are chosen: $\alpha = 0.25$, $a = 3$ and $b = 5$.

5. Optimization Algorithms

5.1. Smoothed gradient descent

We implement a smoothed gradient descent method similar to the approach described by Jameson et al. [3]. The gradient descent method takes a small step in the direction of the negative gradient, such that

$$\delta\mathcal{F} = -\lambda\mathcal{G}. \tag{23}$$

However, this raw approach is generally unsatisfying, because the gradient usually has a lower smoothness than the initial shape. This can lead to the

creation of irregular geometries as the optimization process is performed, or to a crash of the optimization process at worst. To avoid this, the core idea is to define a smoothed gradient $\overline{\mathcal{G}}$ such that:

$$\delta \mathcal{J}_h = \langle \overline{\mathcal{G}}, \delta \mathcal{F} \rangle, \tag{24}$$

where $\langle u, v \rangle$ indicates a weighted Sobolev inner product defined as:

$$\langle u, v \rangle = \int \left(uv + \varepsilon \frac{\partial u}{\partial \xi} \frac{\partial v}{\partial \xi} \right) \mathrm{d}\xi. \tag{25}$$

In our case, all the design variables live on a 1D curve, and the smoothed gradient $\overline{\mathcal{G}}$ is determined through the smoothing equation:

$$\overline{\mathcal{G}} - \frac{\partial}{\partial \xi_1} \varepsilon \frac{\partial}{\partial \xi_1} \overline{\mathcal{G}} = \mathcal{G}, \tag{26}$$

where ε is a smoothing parameter, ξ_1 is the variable describing the position on the design surface, and $\mathcal{G}$ is the gradient computed through Eq. (19). Then, by setting

$$\delta \mathcal{F} = -\lambda \overline{\mathcal{G}}, \tag{27}$$

the variation in target functional after an optimization cycle is:

$$\delta \mathcal{J}_h = -\lambda \langle \overline{\mathcal{G}}, \overline{\mathcal{G}} \rangle, \tag{28}$$

which is strictly negative except when $\overline{\mathcal{G}}$ and $\mathcal{G}$ are equal to 0.

We solve Eq. (26) for the smoothed gradient using a simple second-order central difference approximation

$$\overline{\mathcal{G}}_i - \epsilon \left(\overline{\mathcal{G}}_{i+1} - 2\overline{\mathcal{G}}_i + \overline{\mathcal{G}}_{i-1} \right) = \mathcal{G}_i \quad 1 \leq i \leq n, \tag{29}$$

where $\overline{\mathcal{G}_i}$ and $\mathcal{G}_i$ are the smoothed and unsmoothed gradient at the design variable i and n is the number of design variables used on the chosen design surface. The discrete smoothing parameter ϵ is an input to the design module.

5.2. Coupling with pyOpt

Despite its simplicity, steepest descent is generally not the preferred approach: other optimization methodologies are better suited for aerodynamic shape optimization [29]. We use the open-source software pyOpt [22] to couple our solver to external optimization algorithms. The gradient is thus calculated within the HDG framework and then passed to the optimizer, together with the current target value.

The external optimizers can use very large deformation during line searches. This might result in a non-converging solution, in which case a flag is passed to the optimizer, indicating that the flow solution is not successful. For less sensitive cases, such as Euler solutions described in Section 6, no step limit is used. However, for the Navier-Stokes solutions, the step size taken by the optimizer is limited based on a case-by-case analysis.

For the steepest descent methods, the solution from the previous optimization step is reused. This allows starting the computation of the primal solution from the previous high-order solution. Because the external optimizers can use large deformations to find the optimum, we observed that it was not possible to reuse the flow field solution of the previous iteration as the initial state. Therefore, each optimization step takes more time to be performed. This is however countered by the fact that the convergence rate of the external optimizers is usually faster, as we discuss in Section 6.

6. Results

6.1. Scalar heat transfer

This test case simulates a 2D steady heat equation in a square domain. The external walls are set at a constant temperature of 100°C. An ellipsoid body with an aspect ratio of 2 is placed at the center of the domain, and its walls are set at a constant temperature of 1°C. The temperature field inside the computational domain is governed by Laplace's equation, i.e. there is no heat production nor heat extraction. The objective chosen for this test case is to minimize the heat transfer across the walls of the ellipsoid body. The advantage of this configuration is that the theoretical solution of this problem is known: if we fix the area of the 2D body, the minimization of heat transfer is obtained when the geometry is a circle. This result has already been verified for scalar transport problems in [30].

The target functional is defined as the temperature gradient (which is proportional to the heat flux) across the body walls. However, an area penalty is added as to avoid the geometry simply shrinking to a single point to minimize the target. The objective is thus

$$\mathcal{J} = \int_{body} |\nabla T| \cdot \mathbf{n} ds + c(\mathrm{Vol}(\Omega) - \mathrm{Vol}\,(\Omega_0))^2, \tag{30}$$

where Ω is the computational domain at a given optimization cycle, Ω_0 is the initial domain, and c is the penalty coefficient that has to be fine-tuned. A too-low value of c makes the ASO routine reduce the size of the ellipsoid to a point. On the other hand, a too-high value of c makes the process unstable as the gradient become very large due to a disproportionate impact of the area penalty compared to the main target.

A mesh of 397 elements is used, as depicted in Fig. 5. The initial temperature field is shown in Fig. 6. The 40 geometry points of the ellipse are chosen as design variables.

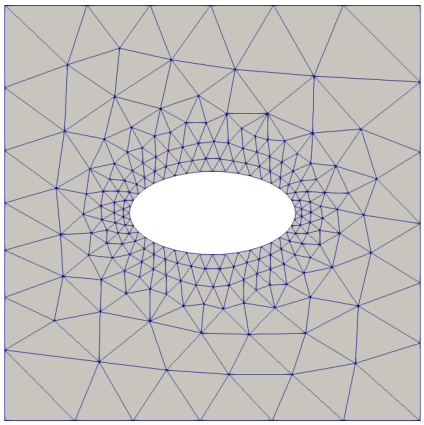

Figure 5: Initial mesh (397 elements)

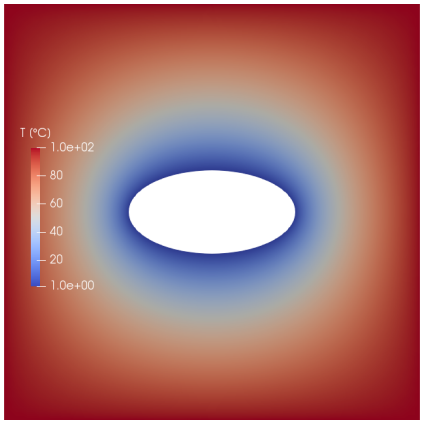

Figure 6: Initial temperature field

6.1.1. Gradient validation

The verification of the gradient with respect to the design variables is made by comparing the gradient obtained with the adjoint approach given in Eq. (19) with a total finite difference approach where the gradient is computed as

$$\frac{\delta \mathcal{J}_h}{\delta F_m} = \frac{\mathcal{J}_h\left(\mathbf{x}^{(+\delta F_m)}, \boldsymbol{\alpha}^{(+\delta F_m)}\right) - \mathcal{J}_h(\mathbf{x}, \boldsymbol{\alpha})}{\delta F_m}. \tag{31}$$

The difference between the gradients computed with Eq. (19) and (31) for different perturbation steps δ from 10^{-15} to 10^{0} is shown in Fig. 7. The gradients obtained at each design variable for $\delta = 10^{-2}$, 10^{-6}, and 10^{-12} are shown in Fig. 8, Fig. 9, and Fig. 10, respectively.

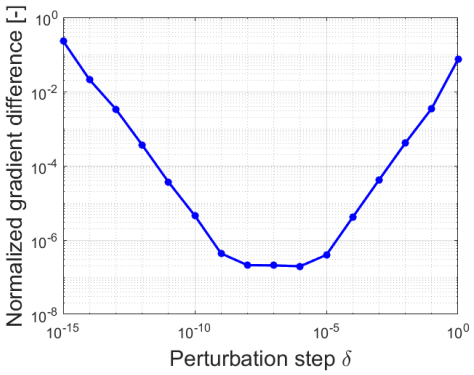

Figure 7: Difference between Eq. (19) and (31)

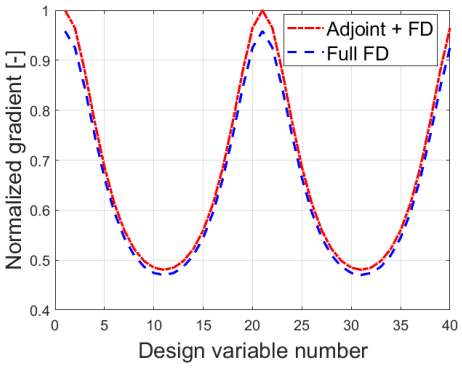

Figure 8: Gradients for $\delta = 10^{-2}$

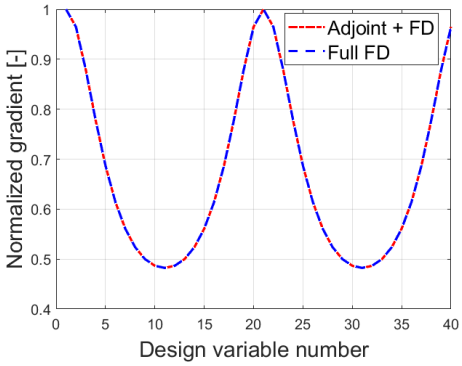

Figure 9: Gradients for $\delta = 10^{-6}$

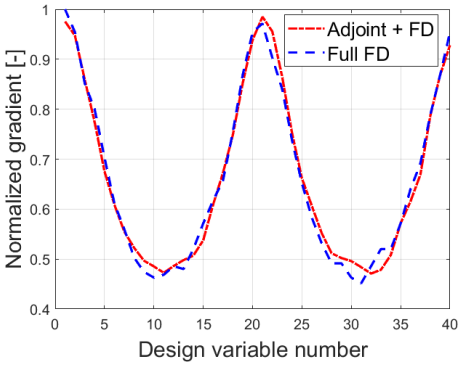

Figure 10: Gradients for $\delta = 10^{-12}$

From Fig. 7, it appears that the optimal range of perturbation steps δ is between 10^{-6} and 10^{-8} for this test case. This can be easily understood as this interval gives the best trade-off between the round-off error (whose relative impact increases when δ decreases) and the finite difference discretization error (which increases when δ increases). This observation is further reinforced by the actual plot of the gradients: the overlap is perfect for a perturbation

step equal to 10^{-6} while the agreement is much less convincing for 10^{-2} and 10^{-12}.

6.1.2. Heat transfer minimization

The final mesh obtained with grid deformation and regeneration are respectively shown in Fig. 11 and Fig. 12. A comparison between the initial ellipsoid geometry and the converged design is also given in Fig. 13, while the final temperature field is displayed in Fig. 14. These were obtained after 40 optimization cycles on a temperature field described by quadratic polynomials ($p = 2$), using steepest descent with a smoothing parameter $\epsilon = 10$ and a constant step size $\lambda = 0.02$.

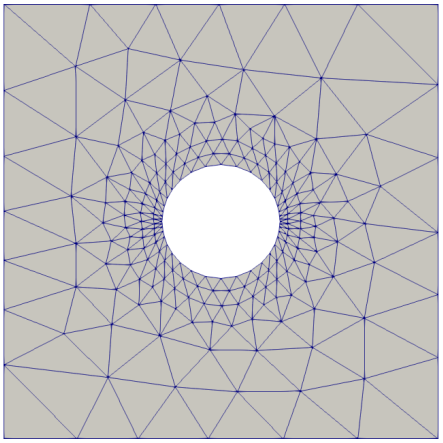

Figure 11: Final deformed mesh (397 elements)

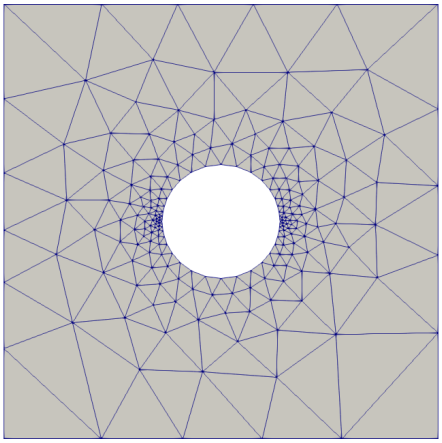

Figure 12: Final regenerated mesh (375 elements)

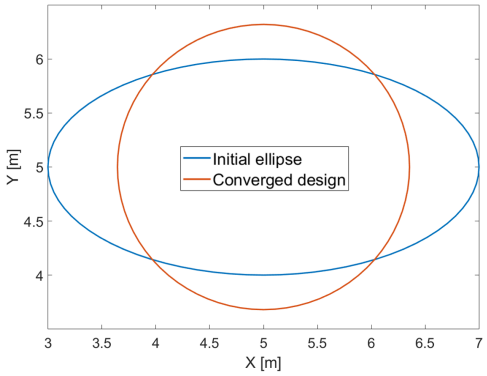

Figure 13: Initial ellipse and final design

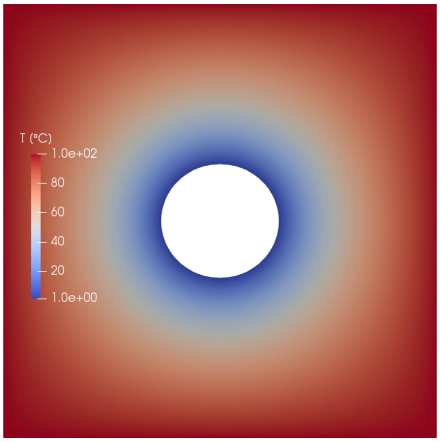

Figure 14: Final temperature field

The ASO routine demonstrates its ability to yield the correct geometry after 40 optimization cycles. Despite the required large geometry deformations, the mesh manipulation algorithms manage to preserve the validity of the elements. It is interesting to note that both the mesh deformation and regeneration approaches lead to a non-uniformly distributed mesh over the geometry. This is to be expected as the initial ellipsoid is not radially symmetric and as the design variables are only allowed to move normally to the surface. Nonetheless, this variation in nodal density along the geometry does not cause any issues for the ASO routine as it is able to converge toward the optimal geometry.

The evolution of the dimensionless target objective with mesh deformation and mesh regeneration is shown in Fig. 15. The shape modification reduces the heat flux across the body by almost 7%. It also appears that both approaches exhibit the same convergence behavior. Therefore, in the following test cases, only the mesh deformation strategy is employed. Indeed,

as the gradient is computed through this approach, it will also be favored for performing the actual shape update.

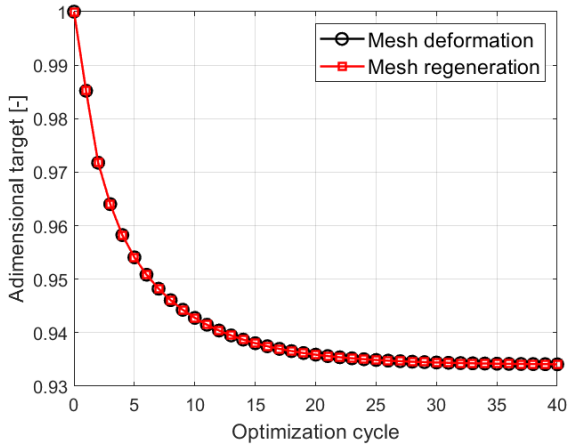

Figure 15: Evolution of the target objective with mesh deformation and with mesh regeneration

6.2. Inverse airfoil design

This test case simulates a NACA 0012 airfoil in a Mach 0.4 flow governed by the inviscid Euler equations. The airfoil is at an angle of attack $\alpha = 2°$ with respect to the incoming flow. The mesh is made of 2560 elements and the simulation uses $p = 2$. For this inverse problem, the objective is to retrieve a NACA 0013 geometry, starting from this initial NACA 0012. This is done by minimizing the pressure difference from a target pressure distribution.

To obtain the target pressure distribution, a forward problem is first run on the NACA 0013 with an order $p = 2$ on a fine mesh made of 13,464 elements. This gives the pressure coefficient curve defined as

$$C_p(x) = \frac{p(x) - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2}, \tag{32}$$

where x is the direction along the chord. In this expression, $p(x)$ is the local pressure at the surface of the airfoil while p_∞, ρ_∞ and V_∞ are respectively the farfield pressure, density, and velocity.

Then, the inverse design problem is run, starting from the NACA 0012 geometry. This initial shape is described by 80 design variables. The target objective to minimize is computed as:

$$\mathcal{J} = \int_{airfoil} (C_{p,target}(x) - C_p(x))^2 dx. \tag{33}$$

The simulations are performed with a steepest descent step $\lambda = 0.05$ for the 25 first cycles. Then, to avoid oscillations around the optimal geometry, the descent step is reduced to $\lambda = 0.04$ for the rest of the 400 cycles. The other parameters chosen for the optimization loop are a quadratic polynomial reconstruction, a finite difference perturbation $\delta = 10^{-6}$ and a smoothing $\epsilon = 100$.

The evolution of this pressure difference throughout the optimization cycles is shown in Fig. 16 and compared with the results obtained with the Preconditioned Sequential Quadratic Programming (PSQP) optimizer from pyOpt [22]. It can be seen that both approaches exhibit a similar convergence behavior. However, the best step size for the steepest descent is determined manually, which is very time-consuming. On the other hand, the standard settings of the PSQP optimizer are used, leading to significant time-saving in setting up the test case.

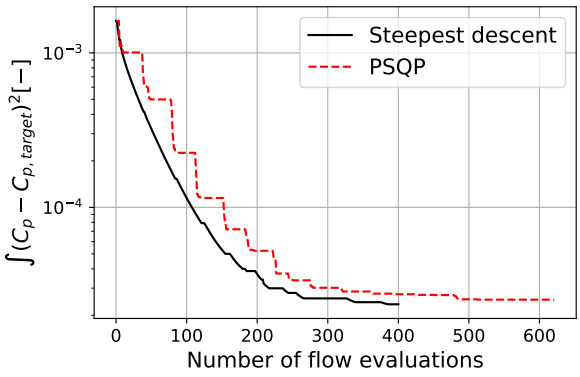

Figure 16: Evolution of the inverse target objective for steepest descent and PSQP

In Fig. 17, the initial, final, and target pressure coefficients along the airfoil are plotted. Good convergence is reached, which was already suggested by the low residual. Some difference is still present near the trailing edge, possibly due to the fast change in pressure between the top and the bottom at that location. Taking a look at Fig. 18, it can be seen that the geometrical shape shows a high degree of similarity: the end geometry is a redesign of the NACA 0013 as the maximum thickness has been adequately increased from 12% to 13% of the chord.

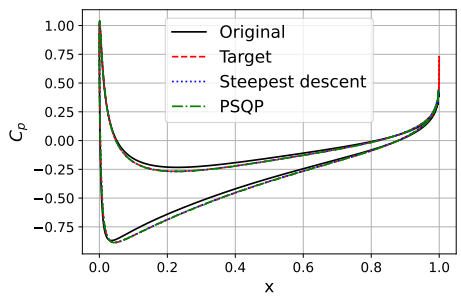

Figure 17: Initial C_p curve and target C_p curve

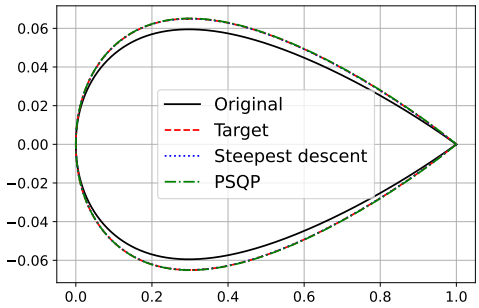

Figure 18: Initial NACA 0012 and converged designs

An inverse design performed with PSQP towards the NACA 0013, but starting from the NACA 1408, is shown in Figures 19 and 20. It can be seen that a small difference on the bottom is still present, but the pressure coefficient distribution is showing good agreement with the target pressure distribution. This demonstrates once again the potential of the coupling with external optimizers.

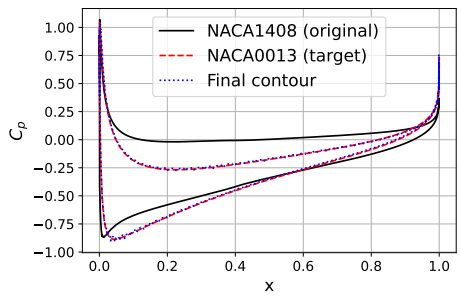

Figure 19: Initial C_p curve and target C_p curve

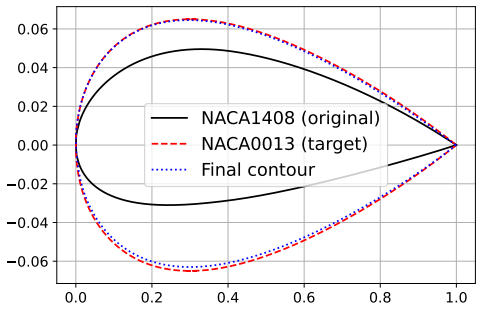

Figure 20: Initial NACA 1408 and converged designs

6.3. Transonic drag minimization

For this test case, a NACA 0012 airfoil is now placed at an angle of attack $\alpha = 0^\circ$ with respect to an inviscid and transonic Mach 0.8 flow, described by quadratic polynomials. In these transonic conditions, a shock wave develops on both the suction and pressure side. These shock waves lead to a considerable amount of drag despite the flow being inviscid.

As a result, the target objective chosen for this test case is the drag coefficient:

$$C_d = \frac{D}{\frac{1}{2}\rho_\infty V_\infty^2 c} \tag{34}$$

where D is the drag experienced by the airfoil and c is its chord. The ASO routine has to find a novel geometry configuration which should get rid of the shock waves or at least attenuate their strength.

In Fig. 21, the convergence behavior of steepest descent and PSQP is shown. For the steepest descent case, an adaptive step size is used to initially drive down the remaining drag quickly, and afterwards reduce it to prevent taking too large steps.

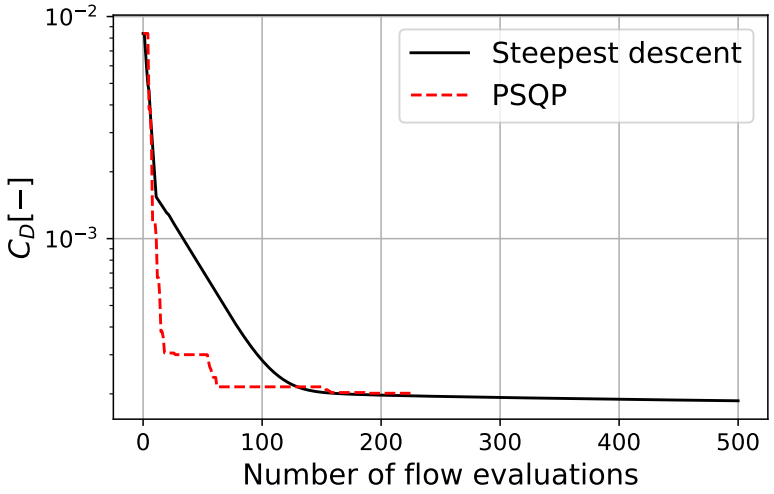

Figure 21: Evolution of the transonic drag of a NACA0012 airfoil for steepest descent and PSQP

The remaining drag is not going to zero when the airfoil is becoming shock-free, which is due to the addition of artificial viscosity. For this transonic flow, a fine-tuned artificial viscosity is indeed used for shock-capturing purposes: decreasing its value reduces the artificial diffusion around the shocks and therefore makes the convergence of the simulations more difficult to reach. A too-high value, on the other hand, leads to a highly stable simulation but to a numerical solution further away from the true physical solution.

It can be seen that the PSQP algorithm has a final drag coefficient similar to the steepest descent. This method makes the airfoil completely shock-free, as depicted in Fig. 22. There, the initial pressure distribution is compared with the final distributions obtained with steepest descent and PSQP.

The remaining small oscillations in the PSQP airfoil pressure distribution

can be explained by the fact that the mesh is deformed from the initial mesh in every step. The large deformations of the geometry can indeed lead to a degradation of the mesh quality. An interesting area of improvement would be to completely restart the optimization with a regenerated mesh based on the final obtained geometry, in order to further optimize the airfoil shape.

While the flow does only speed up before the shock on the initial configuration, it speeds up more at the start for the final configuration, but then decelerates. This deceleration before mid-chord allows for a slow pressure build-up on the final airfoil. As a result, instead of the shock that we observe for the initial geometry, it is rather a steep pressure recovery at the mid-chord for the final geometry. This modification allows reducing the total drag coefficient by a factor of 54, from $C_D = 8.69 \cdot 10^{-3}$ to $C_D = 1.60 \cdot 10^{-4}$.

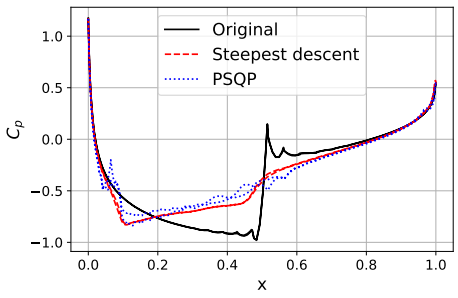

Figure 22: Comparison of the pressure distribution over the airfoil between steepest descent and PSQP

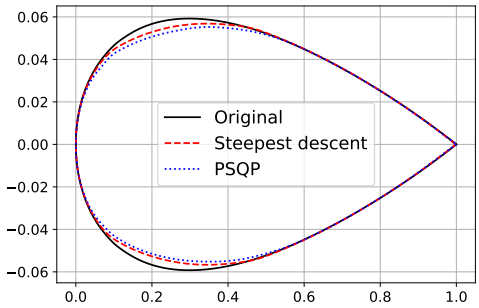

Figure 23: Comparison of the final shape of the airfoil between steepest descent and PSQP

The initial and final geometries obtained with steepest descent and PSQP are shown in Fig. 23, while Fig. 24 and Fig. 25 show respectively the initial and final meshes obtained with the steepest descent method. Looking at

both meshes, it appears that the ASO loop has preserved the quality of the initial mesh. It is interesting to note that the ASO loop has led to practically no modification on the aft part of the airfoil (from mid-chord to the trailing edge). The reason for this modification can be better understood by looking at Fig. 26 and Fig. 27, which show the Mach contours on the initial and final geometries.

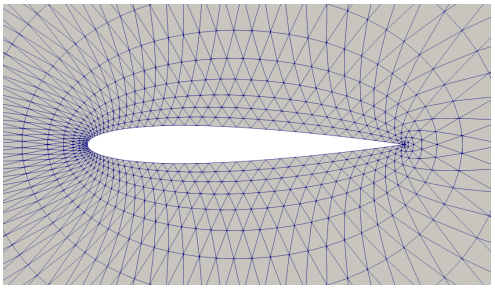

Figure 24: Initial transonic mesh

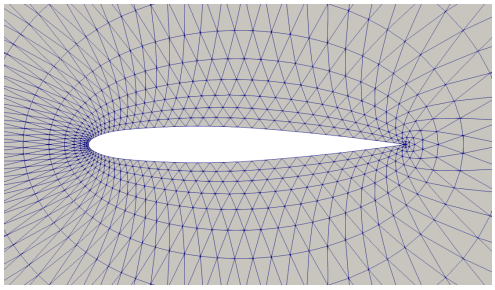

Figure 25: Final transonic mesh

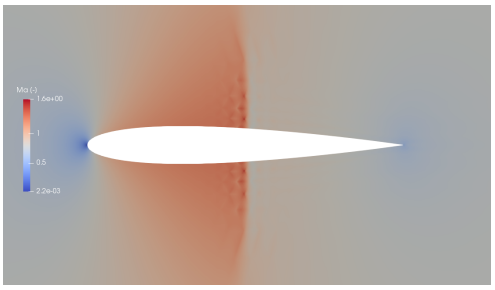

Figure 26: Initial transonic Mach contours

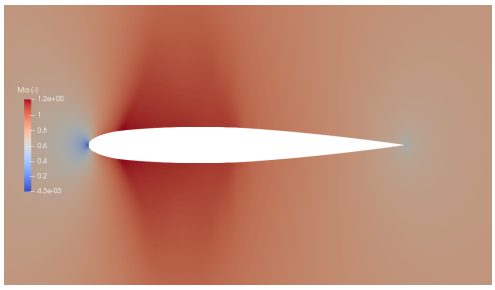

Figure 27: Final transonic Mach contours

Despite the seemingly small geometry changes, the flow field appears to have drastically changed. The initial Mach contours show a shock wave a bit downstream of mid-chord, creating a large amount of wave drag. Thanks to the reduction of thickness in the first 40% of the airfoil chord length, expansion waves develop near the leading edge. These expansion waves increase

the flow speed in this region but lead to strong attenuation of the shock wave downstream.

6.4. Heat transfer

In this section, the analysis of heat transfer into two different initial bodies is performed. We study subsonic flows governed by the steady Navier-Stokes equations. The optimizer used for this section is PSQP.

6.4.1. Blunt wedge

The first test case is a blunt wedge with an opening angle of 9°. The nose is rounded with a radius of 1 and the total length is equal to 6. An initial mesh of 3068 elements (as shown in Fig. 28) is used in a Mach 0.5 flow, with a Reynolds number equal to 100. The wall temperature of the body is taken as 75% of the freestream temperature. Both endpoints of the geometry are fixed throughout the optimization cycles. Three target objectives to minimize are used: the heat transfer $\frac{Q}{Q_0}$ (where the subscript 0 indicates initial values), the heat transfer minus the drag $\frac{Q}{Q_0} - \frac{D}{D_0}$ and the heat-over-drag-ratio $\frac{Q}{D}/\frac{Q_0}{D_0}$. The motivation for this simultaneous minimization of heat transfer and maximization of drag is taken from planetary entry problems. However, since we lack the shock-capturing capabilities for such applications, we content ourselves with subsonic computations.

The target evolution for the heat minimization, as well as the heat-over-drag-ratio, are shown in Fig. 29. It can be seen that the heat is reduced by more than 50% and the heat-over-drag-ratio by more than 30%. For the heat-minus-drag case, the value starts at 0 (due to the target definition) and goes to -0.32.

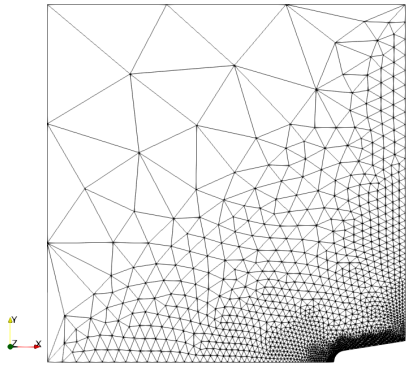

Figure 28: Initial mesh for the blunt wedge

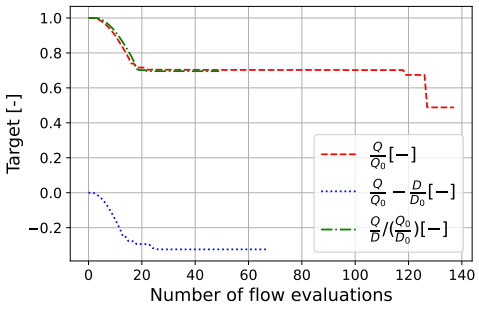

Figure 29: Targets evolution for the blunt wedge

In Fig. 30, we can observe that the three targets result in very similar final geometries. This can be easily understood since minimizing the heat already increases the drag. Therefore, adding the drag contribution to the target objective only drives the target further down. This heat minimization is notably achieved through the expansion that appears after the location of maximum thickness, as well as increased nose radius.

It is important to mention that these results are in agreement with literature [31]. They compare a similar shape in hypersonic flow using second-order finite volume methods and minimize the heat-over-drag-ratio. Due to the shock being present in front of the hypersonic body, a large subsonic region is present downstream, allowing a comparison with our Mach 0.5 flow.

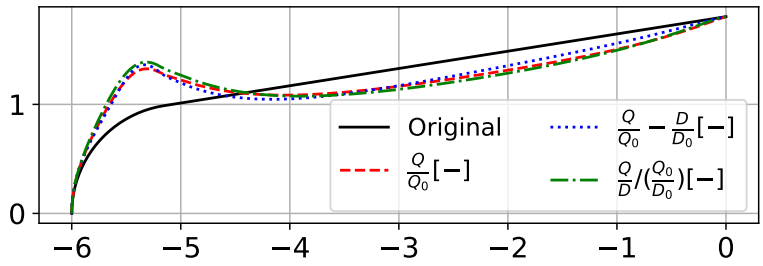

Figure 30: Optimized designs for the blunt wedge

6.4.2. Droplet

A second test case, using a completely closed shape, is performed using a droplet as the initial shape. The mesh consists of $1,515$ elements and the total length of the droplet is equal to 5. The same flow and wall conditions as for the blunt wedge are used. The initial mesh and Mach contours can be seen in Fig. 31. Here, two minimization targets are used: $\frac{Q}{Q_0} - \frac{D}{D_0}$ and $\frac{Q}{D}/\frac{Q_0}{D_0}$. Their evolution is shown in Fig. 32, where it appears that the target objective is decreased by about 35% for both cases.

The final geometries for the heat-minus-drag and the heat-over-drag-ratio are displayed in Fig. 33. It can be seen that the converged design obtained with both targets is again very similar.

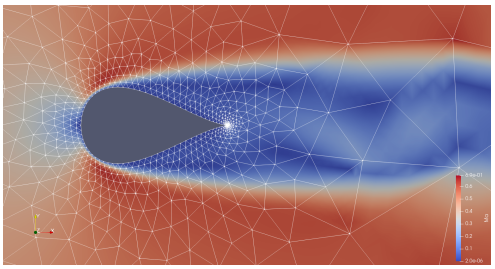

Figure 31: Initial mesh and Mach contours for the droplet

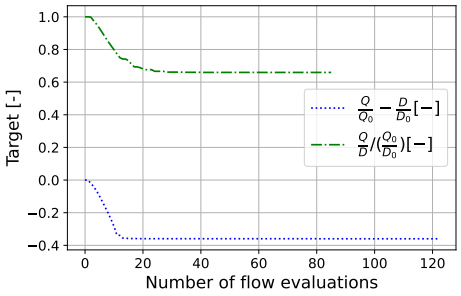

Figure 32: Targets evolution for the droplet

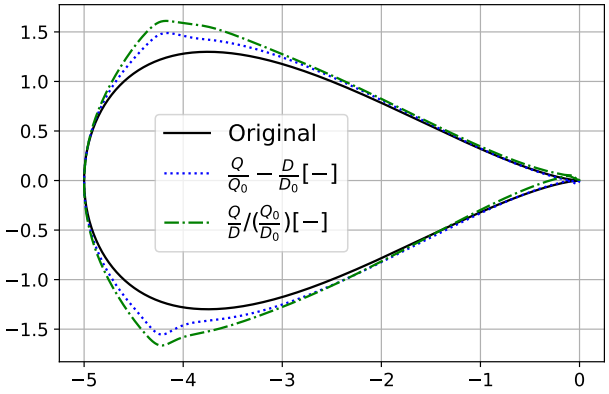

Figure 33: Optimized designs for the droplet

7. Conclusion and Outlook

Only a handful of references have tackled the problem of shape optimization for high-order methods. To the authors' knowledge, this project is the first combination of hybridized discontinuous Galerkin methods with aerodynamic shape optimization routines.

Starting from rational splines for geometry representation, we use a gradient computation based on a combination of the adjoint approach with

finite difference approximations of mesh sensitivities. The volume mesh motion ensuing from the perturbation of design variables is carried out by an interpolation-based approach. An in-house smoothed gradient descent algorithm or an external optimizer from the pyOpt framework [22] are used to find a locally optimal geometry.

Results have been obtained on a variety of test cases. Where applicable, analytical results verify the implementation (Laplacian heat transfer, inverse design, shock-free airfoil). For the remaining cases, plausible results have been obtained, showing a significant reduction in the target functional. The presented tests demonstrate the modularity of our tool. Different sets of conservation laws and compatible target objectives can be chosen.

Future work could be aimed at further improving the gradient computation. Recent references on shape differentiation within Netgen might be useful for this purpose [26]. Moreover, more alternatives to carry out the optimization are available, such as one-shot optimization methods. However, these require a substantial amount of development work. Finally, to use the developed tool in hypersonic flows, an improved shock-capturing scheme is needed.

References

[1] S. Skinner, H. Zare-Behtash, State-of-the-art in aerodynamic shape optimisation methods, Applied Soft Computing 62 (2018) 933–962. doi:10.1016/j.asoc.2017.09.030.

[2] J. Vassberg, A. Jameson, Industrial applications of aerodynamic shape

optimization, VKI Lecture Series on Optimization and Multidisciplinary design (2018). doi:10.35294/ls201804.vassberg2.

[3] A. Jameson, Aerodynamic Shape Optimization Using the Adjoint Method, VKI Lecture Series on Aerodynamic Drag Prediction and Reduction (2003).

[4] M. D. Gunzburger, Perspectives in Flow Control and Optimization, Society for Industrial and Applied Mathematics, 2002.

[5] O. Pironneau, On optimum design in fluid mechanics, Journal of Fluid Mechanics 64 (1974) 97–110.

[6] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, et al., High-order cfd methods: current status and perspective, International Journal for Numerical Methods in Fluids 72 (8) (2013) 811–845. doi:10.1002/fld.3767.

[7] H. T. Huynh, Z. J. Wang, P. E. Vincent, High-order methods for computational fluid dynamics: A brief review of compact differential formulations on unstructured grids, Computers & Fluids 98 (2014) 209–220. doi:10.1016/j.compfluid.2013.12.007.

[8] W. H. Reed, T. R. Hill, Triangular mesh methods for the neutron transport equation, Los Alamos Report LA-UR-73-479 (1973).

[9] B. Cockburn, J. Gopalakrishnan, R. Lazarov, Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for sec-

ond order elliptic problems, SIAM Journal on Numerical Analysis 47 (2) (2009) 1319–1365. doi:10.1137/070706616.

[10] L. Wang, D. J. Mavriplis, W. K. Anderson, Adjoint Sensitivity Formulation for Discontinuous Galerkin Discretizations in Unsteady Inviscid Flow Problems, AIAA Journal 48 (12) (2010) 2867–2883. doi:10.2514/1.J050444.

[11] M. Giacomini, An Equilibrated Fluxes Approach to the Certified Descent Algorithm for Shape Optimization Using Conforming Finite Element and Discontinuous Galerkin Discretizations, Journal of Scientific Computing 75 (1) (2018) 560–595. doi:10.1007/s10915-017-0545-1.

[12] K. Wang, S. Yu, Z. Wang, R. Feng, T. Liu, Adjoint-based airfoil optimization with adaptive isogeometric discontinuous Galerkin method, Computer Methods in Applied Mechanics and Engineering 344 (2019) 602–625. doi:10.1016/j.cma.2018.10.033.

[13] M. Woopen, A. Balan, G. May, J. Schütz, A comparison of hybridized and standard dg methods for target-based hp-adaptive simulation of compressible flow, Computers & Fluids 98 (2014) 3–16. doi:10.1016/j.compfluid.2014.03.023.

[14] M. Woopen, G. May, J. Schütz, Adjoint-based error estimation and mesh adaptation for hybridized discontinuous Galerkin methods, International Journal for Numerical Methods in Fluids 76 (11) (2014) 811–834. doi:10.1002/fld.3959.

[15] A. Balan, M. Woopen, G. May, Adjoint-based hp -adaptivity on anisotropic meshes for high-order compressible flow simulations, Computers & Fluids 139 (2016) 47–67. doi:10.1016/j.compfluid.2016.03.029.

[16] J. Schütz, G. May, An adjoint consistency analysis for a class of hybrid mixed methods, IMA Journal of Numerical Analysis 34 (3) (2014) 1222 – 1239. doi:10.1093/imanum/drt036.

[17] G. May, Hybridized Discontinuous Galerkin Methods : Formulation and Discrete Adjoint, in: VKI 38th Lecture Series on Advanced Computational Fluid Dynamics, 2015, pp. 1–21.

[18] J. Schöberl, NETGEN An advancing front 2D/3D-mesh generator based on abstract rules, Computing and Visualization in Science 1 (1) (1997) 41–52. doi:10.1007/s007910050004.

[19] J. B. Scoggins, T. E. Magin, Development of mutation++ : Multicomponent thermodynamic and transport properties for ionized plasmas written in c++, 11th AIAA/ASME Joint Thermophysics and Heat Transfer Conference (2014). doi:10.2514/6.2014-2966.

[20] G. May, K. Devesse, A. Rangarajan, T. Magin, A Hybridized Discontinuous Galerkin Solver for High-Speed Compressible Flow, Aerospace 8 (11) (2021) 322. doi:10.3390/aerospace8110322.

[21] J. Balis, F. Jacobs, G. May, Aerodynamic shape optimization with hybridized discontinuous galerkin schemes, AIAA SciTech Forum 2023 (2023). doi:10.2514/6.2023-1422.

[22] R. E. Perez, P. W. Jansen, J. R. R. A. Martins, PyOpt: A Python-based object-oriented framework for nonlinear constrained optimization, Structural and Multidisciplinary Optimization 45 (1) (2012) 101–118. doi:10.1007/s00158-011-0666-3.

[23] M. Woopen, A. Balan, G. May, A unifying computational framework for adaptive high-order finite element methods, 22nd AIAA Computational Fluid Dynamics Conference (2015). doi:10.2514/6.2015-2601.

[24] D. N. Arnold, F. Brezzi, B. Cockburn, L. D. Marini, Unified analysis of discontinuous galerkin methods for elliptic problems, SIAM Journal on Numerical Analysis 39 (5) (2002) 1749–1779. doi:10.1137/s0036142901384162.

[25] R. Hartmann, Adaptive discontinuous galerkin methods with shock-capturing for the compressible navier–stokes equations, International Journal for Numerical Methods in Fluids 51 (9-10) (2006) 1131–1156. doi:10.1002/fld.1134.

[26] P. Gangl, K. Sturm, M. Neunteufel, J. Schöberl, Fully and semi-automated shape differentiation in ngsolve, Structural and Multidisciplinary Optimization 63 (3) (2020) 1579–1607. doi:10.1007/s00158-020-02742-w.

[27] E. Luke, E. Collins, E. Blades, A fast mesh deformation method using explicit interpolation, Journal of Computational Physics 231 (2) (2012) 586–601. doi:10.1016/j.jcp.2011.09.021.

[28] N. R. Secco, G. K. W. Kenway, P. He, C. A. Mader, J. R. R. A. Martins, Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization, AIAA Journal (2020). doi:10.2514/1.J059491.

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

[30] M. M. Fyrillas, Shape optimization for 2d diffusive scalar transport, Optimization and Engineering 10 (4) (2008) 477–489. doi:10.1007/s11081-008-9071-1.

[31] S. Eyi, K. M. Hanquist, I. D. Boyd, Shape optimization of reentry vehicles to minimize heat loading, AIAA Scitech 2019 Forum (2019). doi:10.2514/6.2019-0973.