Oral presentation | Mesh generation/adaptation
Mesh generation/adaptation-I
Wed. Jul 17, 2024 10:45 AM - 12:15 PM Room C

[7-C-03] Application of High-order Discontinuous Galerkin Methods for Aerodynamic Shape Optimization with Adaptive Meshing

*Kush Pandya[1], Aravind Balan[1], Georg May[2] (1. Indian Institute of Science, 2. Von Karman Institute for Fluid Dynamics)

Keywords: Aerodynamic Shape Optimization, Discontinuous Galerkin Method, Mesh Adaptation

Twelfth International Conference on
Computational Fluid Dynamics (ICCFD12),
Kobe, Japan, July 14-19, 2024

Application of High-order Discontinuous Galerkin Methods for Aerodynamic Shape Optimization with Adaptive Meshing.

Kush Pandya*, Aravind Balan* and Georg May**
Corresponding author: kushpandya@iisc.ac.in
* Indian Institute of Science, Bengaluru, India.
** Von Karman Institute for Fluid Dynamics, Sint-Genesius-Rode, Belgium.

Abstract: This paper integrates adjoint-based aerodynamic shape optimization within a unified framework employing the Hybridized Discontinuous Galerkin (HDG) flow solver and uses adjoint-based adapted meshes to control the discretization errors. Results are presented for the geometry-constrained inviscid transonic drag minimization benchmark case 1, defined by the Aerodynamic Design Optimization Discussion Group (ADODG). The accuracy of the objective function is crucial for achieving optimal results, necessitating the control of discretization errors. To address this, metric-based anisotropic mesh adaptation using adjoint-based error estimates is employed. Through an inverse design problem focusing solely on the angle of attack as the design variable, the study demonstrates the effectiveness of mesh adaptation in enhancing shape optimization outcomes.

Keywords: Aerodynamic Shape Optimization, High-Order methods, Discontinuous Galerkin method, Mesh Adaptation.

1 Introduction

Driven by the continued progress in numerical optimization techniques and computational fluid dynamics, Aerodynamic Shape Optimization (ASO) has become integral to modern aircraft design. ASO modifies the geometry, defined by a set of design variables, to achieve specific objectives. These objectives include minimizing the drag coefficient [1], achieving a target pressure distribution [2], etc.

ASO methods are classified as either gradient-based or non-gradient-based. Non-gradient-based methods can converge to global optima, while gradient-based methods are more adept at converging to local optima. When dealing with many design variables, the non-gradient-based methods require many function evaluations, making them computationally prohibitive [3, 4]. An adjoint method to calculate gradients [5, 6] allows a gradient-based optimizer to deal with several design variables effectively [7].

Calculating the objective function and its gradient requires solving a set of PDEs, such as the Euler or the Navier-Stokes equations, depending on the flow type. Such PDEs do not exhibit an analytic solution and are required to be solved numerically. High-order methods such as the DG methods and their variations [8, 9, 10] have the potential to solve such equations in an efficient manner. These methods offer better accuracy per degree of freedom than lower-order methods such as the finite volume method [11, 12].

The discretization error in the objective function and its gradients can pollute the optimization results. Controlling the size of the meshes used for discretization can control such errors. A few authors have presented strategies to ensure the reliability of optimization by controlling the mesh size [13, 14]. In the context of aerodynamic shape optimization, Dadone and Grossman[15, 16, 17] have presented methods for shape optimization using a set of uniformly refined meshes.

Mesh adaptation effectively controls discretization errors by targeting regions in the domain that contribute most to the errors while keeping the mesh coarse elsewhere. Adjoint-based error estimates [18, 19], which rely on the adjoint solution of the flow equation, provide localized error estimates for specific objective functions. In aerodynamic shape optimization (ASO), where accurate objective function values are crucial, adjoint-based output error estimators are particularly attractive for mesh adaptation. Significant progress has been made in mesh adaptation strategies, including h-adaptation in finite volume methods and h-, p-, and hp-adaptation for high-order methods. Fidkowski and Darmofal [20] provides a comprehensive review of these strategies. Within ASO, a few authors have implemented strategies

to integrate adjoint-based mesh adaptation [21, 22, 23, 24, 25, 26]. Such strategies have been shown to offer more reliability on the optimization output. These works use finite volume or DG methods or their variations as the flow solvers. A comparative study signifies the effectiveness of DG flow solver over finite volume solver [27] for aerodynamic shape optimization. A variation of the DG method, the Hybridized Discontinuous Galerkin (HDG) method [10], offers the DG schemes' accuracy using lower degrees of freedom. Balis, Jacobs, and May [28] recently integrated a shape optimization module within an HDG solver.

The present work integrates a shape optimization module into a unified framework utilizing the Hybridized Discontinuous Galerkin (HDG) method as the solver and controls the discretization errors using mesh adaptation. The HDG solver we employ was developed by Woopen, Balan, and May [29]. It is implemented in C++ and is structured upon a unifying framework [30] that facilitates the seamless integration of various physical models, discretizations, and iterative solvers. The solver already has an implementation of a mesh adaptation strategy based on the global mesh optimization approach proposed by Rangarajan, May, and Dolejsi [31]. This strategy employs adjoint-weighted residuals for error estimation and can generate highly anisotropic mesh elements.

We utilize a constrained optimizer, the Preconditioned Sequential Quadratic Programming (PSQP) method, from the external optimization library `pyOptSparse` [32, 33], within the HDG solver. Our framework can handle constrained optimization problems for drag minimization and inverse design problems involving multiple design variables. We verify the implementation of our optimizer by solving the benchmark test case for inviscid transonic drag minimization with geometric constraints laid down by the AIAA Aerodynamic Design Optimization Discussion Group (ADODG) [34].

To understand the effect of adaptive mesh, we present results for a single design variable inverse design problem. A target value of pressure distribution at a selected angle of attack is known apriori, and the optimizer's goal is to reach that value of angle of attack using the information of the current pressure distribution and the target distribution. Adaptive meshes help ensure that the angle of attack reaches its target value.

The structure of this paper is as follows: Section 2 discusses the optimization problem, sets the context, and provides the mathematical formulation. Section 3 details the calculation of the objective function and the HDG discretization. Section 4 explains the gradient calculation using the adjoint solution. Section 5 addresses adjoint-based error estimates and the mesh adaptation strategy. Finally, Section 6 discusses the results.

2 Optimization Problem Formulation

Shape optimization can be formulated as a minimization problem to find optimal design variables $\boldsymbol{\mathcal{F}}$ that minimize an objective function $\mathcal{J}$ subject to constraints $\boldsymbol{\mathcal{H}}$. Mathematically, it can be expressed as follows:

$$\begin{aligned} &\underset{\boldsymbol{\mathcal{F}}}{\text{minimize}}\ \mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u}) \\ &\text{subject to}\ \boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{F}}, \mathbf{u}) = 0 \\ &\qquad\qquad \boldsymbol{\mathcal{H}}(\boldsymbol{\mathcal{F}}, \mathbf{u}) \leq 0 \end{aligned} \tag{1}$$

The objective function $\mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u})$ depends on the flow variable $\mathbf{u}$, which is obtained by solving a set of partial differential equations (PDEs), such as the Euler or Navier-Stokes equations. These PDEs can be considered an equality constraint, denoted by $\boldsymbol{\mathcal{R}}$, that must be satisfied at every optimization step. Additional equality and inequality constraints, denoted by $\boldsymbol{\mathcal{H}}$, may also be present.

It is not possible to obtain an analytic solution to the PDEs under consideration, thus necessitating a discretization to achieve a finite-dimensional approximation of the PDEs. By discretizing the domain and focusing solely on the PDE constraints, we can reformulate the optimization problem as follows:

$$\begin{aligned} &\underset{\boldsymbol{\mathcal{F}}}{\text{minimize}}\ \mathcal{J}_h(\boldsymbol{\mathcal{F}}, \mathbf{u}_h) \\ &\text{subject to}\ \boldsymbol{\mathcal{R}}_h(\boldsymbol{\mathcal{F}}, \mathbf{u}_h) = 0 \end{aligned} \tag{2}$$

Here, the subscript h indicates the discretization of the continuous problem. $\mathcal{J}_h(\boldsymbol{\mathcal{F}}, \mathbf{u}_h)$ represents the discretized objective function, and $\boldsymbol{\mathcal{R}}_h(\boldsymbol{\mathcal{F}}, \mathbf{u}_h)$ denotes the discretized residuals of the PDEs, ensuring that the PDE constraints are satisfied within the discretized framework.

For the remainder of this paper, $\mathcal{J}$ represents $\mathcal{J}_h$ unless otherwise specified about the continuous problem.

We solve the optimization problem 2 using a Sequential Quadratic Programming (SQP) type of constrained optimizer. To achieve this, we interface with the external library `pyOptSparse` [33], which provides access to various optimizers, including the Preconditioned Sequential Quadratic Programming (PSQP) optimizer. This optimizer requires the Hessian matrix of the objective function. `pyOptSparse` approximates this Hessian matrix using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which iteratively updates the Hessian approximation based on gradient evaluations.

3 Evaluating the Objective Function

In aerodynamic shape optimization, we evaluate the objective function and constraints using physical parameters such as the drag coefficient and the drag-to-lift ratio. We can represent such an objective function as a surface integral of a scalar function of the flow solutions and their gradients.

We express the objective functional $\mathcal{J}$ mathematically as:

$$\mathcal{J} = \int_{\partial\Omega} j_{\partial\Omega}(\mathbf{u}, \nabla\mathbf{u})\, dx, \tag{3}$$

where $\mathbf{u}$ represents the flow solutions, and $\nabla\mathbf{u}$ denotes their gradients. We take the integral over the boundary $\partial\Omega$, corresponding to the geometry.

3.1 Governing Equations

We can write the flow equations in a general conservation form as:

$$\begin{aligned} \nabla\cdot\mathbf{f}(\mathbf{u},\nabla\mathbf{u}) &= \mathbf{s}(\mathbf{u},\nabla\mathbf{u}) \quad \text{in } \Omega\subset\mathbb{R}^d,\\ \mathbf{u} &= g_{_D} \quad \text{on } \Gamma_{_D},\\ \mathbf{n}\cdot\nabla\mathbf{u} &= g_{_N} \quad \text{on } \Gamma_{_N}. \end{aligned} \tag{4}$$

Here, the total flux and the source term are defined as:

$$\begin{aligned} \mathbf{f} &: \mathbb{R}^m \times \mathbb{R}^{m\times d} \to \mathbb{R}^{m\times d},\\ \mathbf{s} &: \mathbb{R}^m \times \mathbb{R}^{m\times d} \to \mathbb{R}^m, \end{aligned}$$

where m denotes the number of conserved variables and d denotes the spatial dimension. We can decompose the total flux $\mathbf{f}(\mathbf{u},\nabla\mathbf{u})$ into convective and viscous components as:

$$\mathbf{f}(\mathbf{u},\nabla\mathbf{u}) = \mathbf{f}^c(\mathbf{u}) - \mathbf{f}^v(\mathbf{u},\nabla\mathbf{u}). \tag{5}$$

Based on this decomposition and using the mixed formulation, we can write the conservation law as:

$$\begin{aligned} \nabla\cdot\mathbf{f}^c(\mathbf{u}) - \nabla\cdot\mathbf{f}^v(\mathbf{u},\nabla\mathbf{u}) &= \mathbf{s}(\mathbf{u},\nabla\mathbf{u}) \quad \text{in } \Omega\subset\mathbb{R}^d,\\ \mathbf{q} &= \nabla\mathbf{u} \quad \text{in } \Omega\subset\mathbb{R}^d,\\ \mathbf{u} &= g_{_D} \quad \text{on } \Gamma_{_D},\\ \mathbf{n}\cdot\mathbf{q} &= g_{_N} \quad \text{on } \Gamma_{_N}. \end{aligned} \tag{6}$$

The mixed formulation provided by these equations is frequently applied in viscous discontinuous Galerkin discretizations because it constitutes a system of only first-order partial differential equations.

3.2 HDG Discretization

Discontinuous Galerkin (DG) methods [8] have recently gained considerable interest for their high-order accuracy. However, they have high degrees of freedom (DOF), necessitating significant memory and computational cost. Hybridized Discontinuous Galerkin (HDG) methods [9] are a variation of the standard DG methods that reduce the number of globally coupled degrees of freedom.

The main advantage of hybridizing finite element discretizations is that the resulting set of algebraic equations has globally coupled degrees of freedom only on the skeleton of the computational mesh.

Consequently, solving for these degrees of freedom involves solving a much smaller system. This reduces storage requirements and allows for a faster solution with iterative solvers.

The domain Ω is discretized into non-overlapping elements, $\mathcal{T}_h$, such that $\cup_{K\in\mathcal{T}_h} K = \Omega$. Two distinct sets are defined: $\partial\mathcal{T}_h$ for element edges and Γ_h for element faces:

$$\partial\mathcal{T}_h := \{\partial K \setminus \partial\Omega : K \in \mathcal{T}_h\}, \tag{7}$$

$$\Gamma_h := \{e : e = K \cap K' \text{ for } K, K' \in \mathcal{T}_h;\ \text{meas}_{d-1}(e) \neq 0\}. \tag{8}$$

These sets do not include domain boundary edges, denoted by Γ_h^b. Let $\Pi^p(D)$ denote polynomials of degree at most p on domain D. Discontinuous function spaces are defined as:

$$V_h = \{v \in L^2(\Omega) : v|_K \in \Pi^{p_K}(K),\ K \in \mathcal{T}_h\}^{m\times d}, \tag{9}$$

$$W_h = \{u \in L^2(\Omega) : u|_K \in \Pi^{p_K}(K),\ K \in \mathcal{T}_h\}^{m}, \tag{10}$$

$$M_h = \{\mu \in L^2(\Gamma_h) : \mu|_e \in \Pi^{p_e}(e),\ e \in \Gamma_h\}^{m}. \tag{11}$$

Assuming uniform polynomial orders across elements and edges ($p_K = p_e = p$), $\mathbf{v} \in V_h$, $\mathbf{u} \in W_h$, and $\boldsymbol{\mu} \in M_h$ are piecewise polynomials of degree p, exhibiting discontinuities across edges (for $\mathbf{v}$ and $\mathbf{u}$) or vertices (for $\boldsymbol{\mu}$).

The HDG method is expressed as a variational equation within $\mathbb{X}_h = V_h \times W_h \times M_h$:

$$\mathbf{x}_h \in \mathbb{X}_h : \quad N_h(\mathbf{x}_h; \mathbf{y}_h) = 0 \quad \forall \mathbf{y}_h \in \mathbb{X}_h, \tag{12}$$

where the semilinear form N_h is derived from DG discretization on each element, ensuring conservation through continuity of normal flux across elements.

For more details of the semi-linear form, we refer to [30].

4 Evaluating the Gradients

In gradient-based optimization, achieving efficient and precise gradient calculations is essential. The adjoint method provides a practical approach for computing these gradients. Adjoints can be discrete or continuous [35], depending on how we formulate them. Although the continuous adjoint method is recognized for its computational efficiency, we chose the discrete one due to its straightforward implementation [7]. The HDG framework already integrates a discrete adjoint solver for mesh adaptation. We make use of the adjoint solver for gradient calculation.

4.1 Gradient Calculation Using Discrete Adjoint Solution

We present a summary for formulating the gradients using the adjoint solution similar to [6]. To compute the gradients of the objective function $\mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u})$ with respect to the design variable $\boldsymbol{\mathcal{F}}$, we begin with the flow solution $\mathbf{u}$, obtained by driving the discrete residual vector $\mathcal{R}(\boldsymbol{\mathcal{F}}, \mathbf{u})$ to zero. Hence, we have:

$$\mathcal{J} = \mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u}), \tag{13}$$

$$\mathcal{R}(\boldsymbol{\mathcal{F}}, \mathbf{u}) = 0. \tag{14}$$

The gradient of the objective function can be calculated using the chain rule:

$$\frac{d\mathcal{J}}{d\boldsymbol{\mathcal{F}}} = \frac{\partial\mathcal{J}}{\partial\boldsymbol{\mathcal{F}}} + \frac{\partial\mathcal{J}}{\partial\mathbf{u}}\frac{\partial\mathbf{u}}{\partial\boldsymbol{\mathcal{F}}}. \tag{15}$$

Applying the chain rule to the total derivative of the residual $\mathcal{R}$ with respect to the design variable $\boldsymbol{\mathcal{F}}$ and setting it to zero, as the residual must be zero at each design iteration, we obtain:

$$\frac{d\mathcal{R}}{d\boldsymbol{\mathcal{F}}} = \frac{\partial\mathcal{R}}{\partial\boldsymbol{\mathcal{F}}} + \frac{\partial\mathcal{R}}{\partial\mathbf{u}}\frac{\partial\mathbf{u}}{\partial\boldsymbol{\mathcal{F}}} = 0, \tag{16}$$

$$\frac{\partial\mathbf{u}}{\partial\boldsymbol{\mathcal{F}}} = -\left(\frac{\partial\mathcal{R}}{\partial\mathbf{u}}\right)^{-1}\frac{\partial\mathcal{R}}{\partial\boldsymbol{\mathcal{F}}}. \tag{17}$$

Substituting the expression for $\frac{\partial \mathbf{u}}{\partial \boldsymbol{\mathcal{F}}}$ into the equation for the gradient of $\mathcal{J}$, we get:

$$\frac{d\mathcal{J}}{d\boldsymbol{\mathcal{F}}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\mathcal{F}}} - \frac{\partial \mathcal{J}}{\partial \mathbf{u}} \left(\frac{\partial \mathcal{R}}{\partial \mathbf{u}} \right)^{-1} \frac{\partial \mathcal{R}}{\partial \boldsymbol{\mathcal{F}}}. \tag{18}$$

We solve the system of equations $\frac{\partial \mathcal{J}}{\partial \mathbf{u}} \left(\frac{\partial \mathcal{R}}{\partial \mathbf{u}} \right)^{-1} \frac{\partial \mathcal{R}}{\partial \boldsymbol{\mathcal{F}}}$ using the adjoint method, defining the adjoint vector $\boldsymbol{\psi}$ as:

$$\boldsymbol{\psi} = \left(\frac{\partial \mathcal{R}^T}{\partial \mathbf{u}} \right)^{-1} \left(\frac{\partial \mathcal{J}}{\partial \mathbf{u}} \right)^T. \tag{19}$$

This linear system scales with the number of functions of interest rather than the number of design variables. Hence, the total derivative can be expressed as:

$$\frac{d\mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u})}{d\boldsymbol{\mathcal{F}}} = \frac{\partial \mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u})}{\partial \boldsymbol{\mathcal{F}}} - \boldsymbol{\psi}^T \frac{\partial \mathcal{R}(\boldsymbol{\mathcal{F}}, \mathbf{u})}{\partial \boldsymbol{\mathcal{F}}}. \tag{20}$$

For a primal solution $\mathbf{u}_h \in \tilde{\mathbb{X}}_h$, the adjoint solution vector $\boldsymbol{\psi}_h \in \tilde{\mathbb{X}}_h$ is represented by:

$$\mathbf{N}'_h[\mathbf{u}_h](\mathbf{y}_h, \boldsymbol{\psi}_h) = \mathcal{J}'_h[\mathbf{u}_h](\mathbf{y}_h) \quad \forall \mathbf{y}_h \in \tilde{\mathbb{X}}_h, \tag{21}$$

where $\mathbf{N}'_h[\mathbf{x}_h]$ and $\mathcal{J}'_h[\mathbf{x}_h]$ denote the linearizations of $\mathbf{N}_h$ and $\mathcal{J}_h$ with respect to $\mathbf{x}_h$. Within the HDG framework, the discrete adjoint solution is implemented for error estimation and mesh adaptation. To prevent error estimates from vanishing identically, the adjoint solution is computed in a richer space. Thus, the space $\tilde{\mathbb{X}}_h$ is defined with a polynomial order of $p + 1$.

The partial derivative terms on the right-hand side of Equation (20) are calculated using finite differences:

$$\frac{\partial \mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u})}{\partial \boldsymbol{\mathcal{F}}} = \frac{\mathcal{J}(\boldsymbol{\mathcal{F}} + \delta\boldsymbol{\mathcal{F}}, \mathbf{u}) - \mathcal{J}(\boldsymbol{\mathcal{F}}, \mathbf{u})}{\delta \boldsymbol{\mathcal{F}}}, \tag{22}$$

$$\frac{\partial \mathcal{R}(\boldsymbol{\mathcal{F}}, \mathbf{u})}{\partial \boldsymbol{\mathcal{F}}} = \frac{\mathcal{R}(\boldsymbol{\mathcal{F}} + \delta\boldsymbol{\mathcal{F}}, \mathbf{u}) - \mathcal{R}(\boldsymbol{\mathcal{F}}, \mathbf{u})}{\delta \boldsymbol{\mathcal{F}}}. \tag{23}$$

No additional flow solution is required to calculate these partial derivatives. The design variable and mesh are deformed by a small perturbation $\delta\boldsymbol{\mathcal{F}}$, and the solution is reconstructed on this deformed mesh to evaluate the perturbed objective function and its residuals.

5 Mesh Adaptation and Error Estimation

Owing to the necessity of accurately computing the objective function and its gradient, adaptive meshes offer a compelling approach. Mesh adaptation refines the domain in areas that significantly impact the calculation of the objective function. Adjoint-based error estimates provide indicators for error distribution within the domain. Within the optimization context, the adjoint solution, which is already available for calculating gradients, can be leveraged for mesh adaptation. This use of the adjoint solution makes adjoint-based error estimates particularly advantageous in optimization processes.

5.1 Adjoint-based Error Estimation

Consider the functional $\mathcal{J}_h$ representing the objective function for shape optimization. To frame adjoint-based error estimation for $\mathcal{J}_h$, consider the error in this quantity,

$$e_h := \mathcal{J}_h(\mathbf{u}) - \mathcal{J}_h(\mathbf{u}_h). \tag{24}$$

For the derivation of the adjoint-based error estimate, we expand the target functional in a Taylor series:

$$\mathcal{J}_h(\mathbf{u}) - \mathcal{J}_h(\mathbf{u}_h) = \mathcal{J}_h'[\mathbf{u}_h]\delta\mathbf{u}_h + \mathcal{O}(\|\delta\mathbf{u}_h\|^2), \tag{25}$$

where $\delta\mathbf{u}_h := \mathbf{u} - \mathbf{u}_h$, $\mathbf{u}_h \in \tilde{\mathbb{X}}$ represents the numerical solution, and $\mathbf{u}$ represents the actual solution.

We proceed similarly with the error in the residual:

$$N_h(\mathbf{u};\mathbf{y}_h) - N_h(\mathbf{u}_h;\mathbf{y}_h) = N_h'[\mathbf{u}_h](\delta\mathbf{u}_h;\mathbf{y}_h) + \mathcal{O}(\|\delta\mathbf{u}_h\|^2). \tag{26}$$

Since our discretization is consistent, the first term $N_h(\mathbf{u};\mathbf{y}_h)$ vanishes. Substituting the above equation into the error expression and neglecting quadratic terms yields:

$$e_h \approx \eta := -N_h(\mathbf{u}_h;\boldsymbol{\psi}_h), \tag{27}$$

where $\boldsymbol{\psi}_h$ is defined by the adjoint equation:

$$N_h'[\mathbf{u}_h](\mathbf{y}_h;\boldsymbol{\psi}_h) = \mathcal{J}_h'[\mathbf{u}_h](\mathbf{y}_h) \quad \forall\mathbf{y}_h \in \tilde{\mathbb{X}}_h. \tag{28}$$

The adjoint solution $\boldsymbol{\psi}_h \in \tilde{\mathbb{X}}_h$ provides the connection between variations in the residual and the target functional. The global error estimate η can be localized to individual elements to yield local indicators for driving an adaptation procedure:

$$\eta_K := \|N_h(\mathbf{u}_h;\boldsymbol{\psi}_h)\|_K. \tag{29}$$

It is crucial that the functionals N_h and $\mathcal{J}_h$ and their Jacobians be evaluated in a space richer than $\mathbb{X}_h$, specifically $\tilde{\mathbb{X}}_h \supset \mathbb{X}_h$. Otherwise, the weighted residual $N_h(\mathbf{u}_h;\boldsymbol{\psi}_h)$ would be zero because:

$$N_h(\mathbf{u}_h;\mathbf{y}_h) = 0 \quad \forall\mathbf{u}_h \in \mathbb{X}_h. \tag{30}$$

This requirement can be satisfied either through mesh refinement or by increasing the polynomial degree of the ansatz functions. In our setting, the latter approach is more advantageous in terms of implementation effort.

5.2 Mesh Adaptation Strategy

For our study, we employ the goal-oriented anisotropic mesh adaptation strategy proposed by Rangarajan[31]. This method is based on a novel output-based mesh adaptation approach that incorporates adjoint-based error estimates within the continuous mesh model framework. This strategy effectively refines the mesh in regions where high accuracy is required, thereby optimizing computational resources and improving the overall solution quality. We direct interested readers to the original work by Rangarajan for a comprehensive explanation and implementation details [31].

6 Results and Discussion

6.1 Drag minimization case

AIAA's Aerodynamic Design Optimization Discussion Group (ADODG) has laid down a set of benchmark problems [34] to help evaluate the utility of optimization methods and processes. The first problem is the inviscid transonic drag minimization problem, which is subjected to geometric constraints. Here, the optimization searches the design space starting from a modified NACA0012 airfoil with a zero thickness trailing edge. A geometric constraint is imposed that restricts the final thickness of the airfoil to be more than the baseline geometry's thickness. The airfoil is subjected to an inviscid flow at a Mach number of 0.85 at an angle of attack (α) of zero degrees. It is a zero-lifting airfoil. Mathematically, the problem can be formulated as,

$$\begin{aligned} &\underset{\boldsymbol{\mathcal{F}}}{\text{minimize}}\ C_d(\boldsymbol{\mathcal{F}},\mathbf{u}) \\ &\text{subject to } y \geq y_{baseline}, \end{aligned} \tag{31}$$

Where y indicates the thickness of the airfoil.

Since the flow is inviscid and transonic, the governing equations are the 2D compressible Euler equations. In the form of the standard conservation laws (4), the flow variables and fluxes for these

equations can be written as:

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \qquad \mathbf{f} = \mathbf{f}^c \begin{bmatrix} \rho u & \rho v \\ \rho u^2 & \rho uv \\ \rho uv & \rho v^2 \\ u(E+p) & v(E+p) \end{bmatrix}, \tag{32}$$

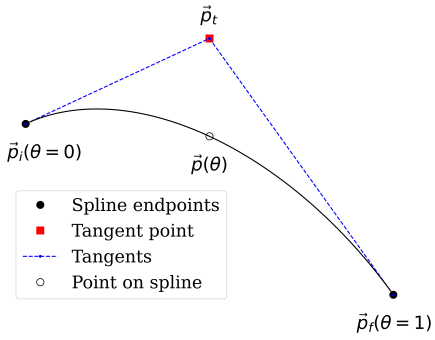
Where ρ is the density, u is the x-velocity, v is the y-velocity, E is the total energy and p is the pressure.

We use the state equation relating pressure and Energy to close the above system of equations.

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho(u^2 + v^2)\right), \tag{33}$$

where γ is the ratio of specific heat capacities.

The HDG framework utilizes Netgen, an open-source geometry and mesh generation tool part of the finite element package NGSolve. Netgen represents 2D geometries using piecewise rational quadratic splines. In our study, we represent the airfoil as a set of points, which we then close using these splines. We impose first-order continuity at each point, except at the endpoints where we apply second-order continuity or a 'natural' boundary condition. We use the points defining the splines as design points, allowing them to move in a direction normal to the geometry in the 2D plane. Owing to the symmetry of the problem, we only consider the upper half of the airfoil points as the design points and replicate the mirror value to obtain the lower surface. Figure (1) describes the spline and the points used to define it.



$$\vec{p}(\theta) = \frac{(1-\theta)^2\vec{p}_i + 2w\theta(1-\theta)\vec{p}_t + \theta^2\vec{p}_f}{(1-\theta)^2 + 2w\theta(1-\theta) + \theta^2}$$
$$w = \frac{\|\vec{p}_f - \vec{p}_i\|}{\sqrt{0.5(\|\vec{p}_f - \vec{p}_t\|^2 + \|\vec{p}_t - \vec{p}_i\|^2)}} \tag{34}$$

Figure 1: Spline geometry

As the geometry changes, the mesh must be correspondingly deformed. We achieve this through an interpolation-based mesh deformation strategy. For the detailed implementation of this strategy, we refer to [36].

Figure (2) compares the initial airfoil and the optimized airfoil. The markers on the geometry represent the design variables. The leading edge develops a bluntness, and the geometry constraint is satisfied at every design point. Figure (3) shows the objective convergence over more than 550 optimization iterations. The initial geometry had a drag coefficient value of 435.8 drag count (dc), where 1 drag count = 0.0001. The optimized design has a drag coefficient of 121.4 dc.

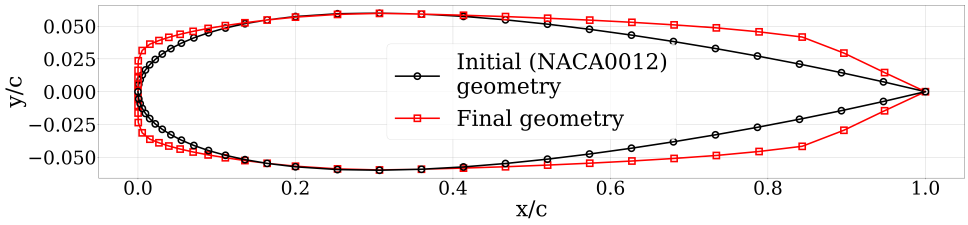


Figure 2: Initial and optimized geometries

Figure (4) shows the coefficient of pressure on initial and optimized airfoils. Figure (5) and Figure (6) show the plot of Mach number on the initial and final design, respectively. The initial geometry shows a strong shock at a location extending towards the trailing edge. Owing to the thickness constraint, the optimized geometry is a thick airfoil with a flat surface. The optimized airfoil shows two shock features:

a strong shock is evident near the leading edge, while a weaker shock is observed near the trailing edge. The results are comparable to those obtained by other authors for the same case.

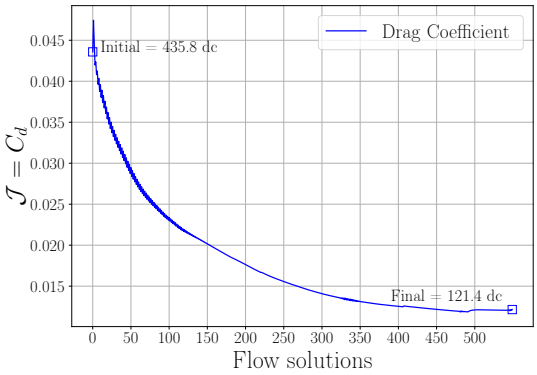


Figure 3: Objective convergence history

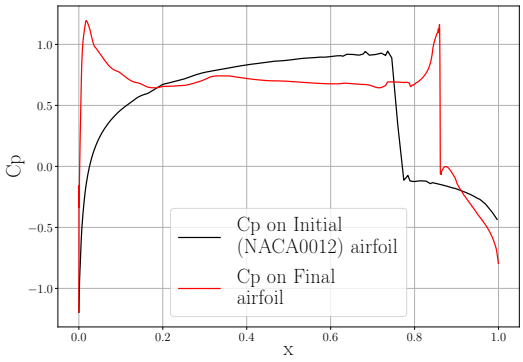


Figure 4: Pressure coefficient distribution on initial and final geometry

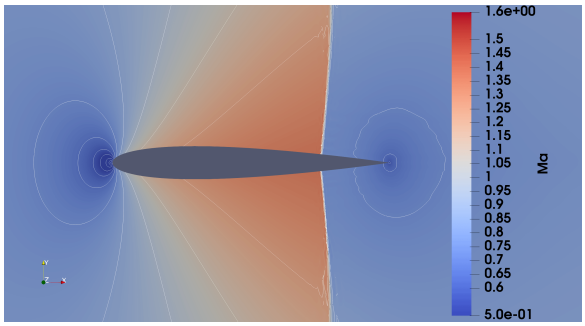


Figure 5: Mach number distribution on the initial geometry

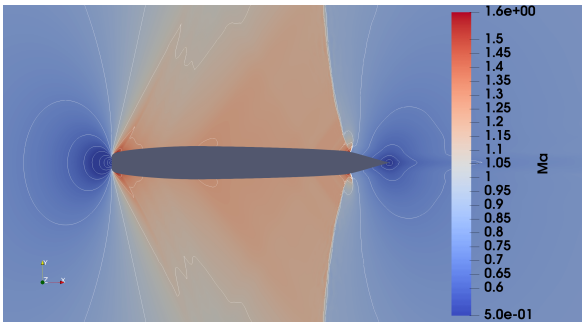


Figure 6: Mach number distribution on the final geometry

6.2 Inverse Design case

We consider an inverse design problem to analyze the impact of adaptive meshes on optimization convergence. In this scenario, the design variable is the angle of attack. A NACA0012 airfoil is subjected to inviscid transonic flow at a Mach number 0.8. The target pressure coefficient is specified at an angle of attack of two degrees. The objective of the optimization problem is to adjust the angle of attack to achieve the desired value of two degrees, starting from an arbitrary initial angle of attack. For this study, the initial angle of attack is set to zero degrees. Compressible Euler equations are the governing equations.

The objective function is defined as follows:

$$\mathcal{J} = \int_{\text{airfoil}} \left(C_{p,\text{target}}(x) - C_p(x)\right)^2 \, dx. \tag{35}$$

The pressure coefficient is defined as:

$$C_p(x) = \frac{p(x) - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2}, \tag{36}$$

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

A fine mesh with 233,472 elements is employed to obtain the target pressure distribution. The Hybridized Discontinuous Galerkin (HDG) solver is used with a polynomial order of two. Figure (7) shows the target $C_{p,\text{target}}$ distribution.

Optimization starts with an initial angle of attack of zero degrees. The gradient of the objective function with respect to the angle of attack, is calculated using the adjoint method:

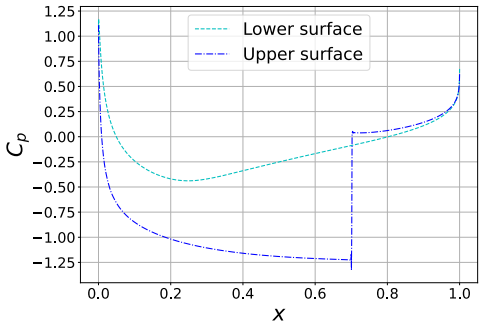


Figure 7: Target value of the coefficient of pressure $C_{p,\text{target}}$ obtained by solving the flow on a NACA0012 airfoil at Mach 0.8 and $\alpha = 2$ using a fine mesh with 233472 mesh elements

$$\frac{d\mathcal{J}(\alpha, \mathbf{u})}{d\alpha} = \frac{\mathcal{J}(\alpha + \delta\alpha, \mathbf{u}) - \mathcal{J}(\alpha, \mathbf{u})}{\delta\alpha} - \boldsymbol{\psi}^T \frac{\boldsymbol{\mathcal{R}}(\alpha + \delta\alpha, \mathbf{u}) - \boldsymbol{\mathcal{R}}(\alpha, \mathbf{u})}{\delta\alpha}, \tag{37}$$

We use finite differences similar to equations (22) and (23) to calculate the partial derivatives on the right-hand side of the above equation. For calculating $\mathcal{J}(\alpha + \delta\alpha, \mathbf{u})$ and $\boldsymbol{\mathcal{R}}(\alpha + \delta\alpha, \mathbf{u})$, a new flow solution is not calculated. Instead, the geometry is tilted by the amount $\delta\alpha$, and the mesh is deformed accordingly. The older solution is reconstructed on this tilted mesh and geometry, and the perturbed objective function and residuals are recalculated. We use an interpolation-based approach similar to [36] for mesh deformation.

6.2.1 Comparison between optimized results for coarse and fine discretization

To understand the impact of mesh quality on the optimized result, we solve the aforementioned optimization problem using a coarse and a fine meshes. The coarse mesh comprises 438 elements, while the fine mesh comprises 40,960. The flow is solved using a polynomial order of two. For an initial angle of attack, α, of zero degrees, Figures (8) and (9) shows the Mach number distribution.

Using these meshes, we run the optimizer, and the evolution of the angle of attack with respect to optimization iterations is shown in Figure (10). The target α, represented by the dashed blue line, is two degrees. For the fine mesh, the angle of attack converged to a value very close to the target value. Conversely, α converged to 1.828 degrees for the coarse mesh, which is not as close to the target value. The non-convergence observed in the coarse mesh can be attributed to its inability to resolve the flow solution adequately. Figure (11) shows the plot of coefficient of pressure for the target configuration and the optimal design values obtained using fine and coarse mesh. The fine mesh plot almost aligns with the target plot. While we can see diffusion of shock for the plot corresponding to the coarse mesh.

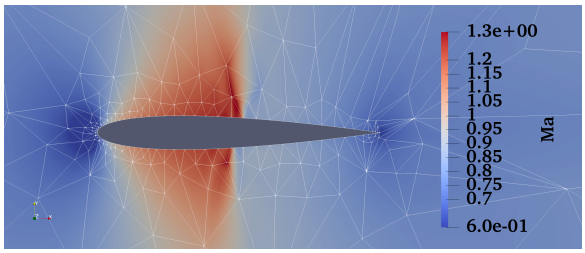


Figure 8: Mach number distribution on the coarse mesh

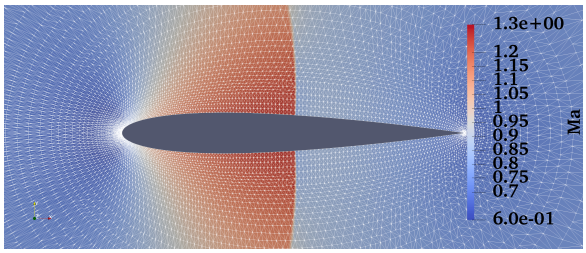


Figure 9: Mach number distribution on the fine mesh

The comparison clearly illustrates that the choice of mesh significantly influences the optimization results. With its higher resolution, the fine mesh provides a more accurate flow solution, leading to better convergence of both the design variable and the objective function. In contrast, the coarse mesh fails to capture critical flow features, resulting in suboptimal convergence. Controlling the error through finer discretization can improve the accuracy and reliability of the optimization process.

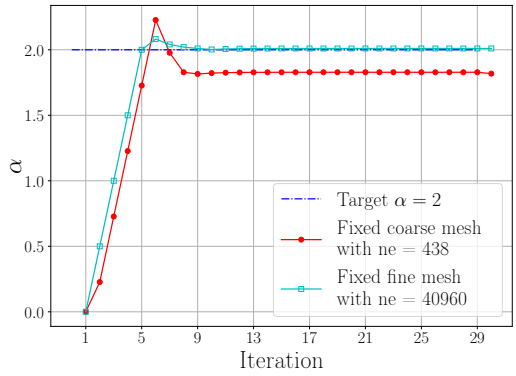


Figure 10: Evolution of the design variable α for coarse and fine mesh

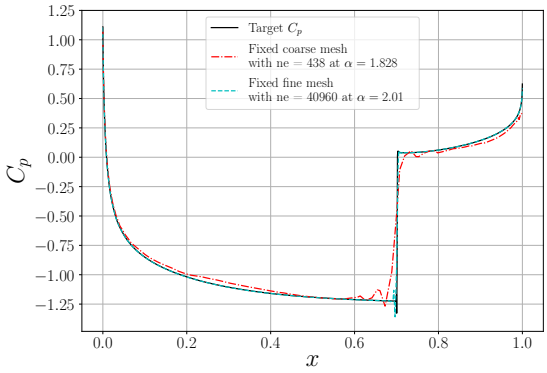


Figure 11: C_p plot on the upper surface of the airfoil

6.2.2 Optimization Using Adapted Meshes

Previous results highlight the influence of discretization errors on the outcomes of shape optimization. Here, we present results using adapted meshes instead of a fine mesh for optimization.

Mesh adaptation is guided by error estimates in the target functional, calculated using the adjoint method. The objective function of optimization is typically a suitable candidate for mesh adaptation. However, in the case of inverse design, the objective function diminishes as it approaches its target value. Therefore, we define the target functional for mesh adaptation as the integral of the square of the coefficient of pressure:

$$\mathcal{J}_{adapt} = \int_{\Omega} C_p(x)^2 \, dx. \tag{38}$$

The optimization begins with an initial angle of attack $\alpha = 0$ degrees. Initially, a very coarse mesh consisting of 438 elements is employed. The optimizer converges to an angle of attack $\alpha = 1.828$ degrees, which deviates from the target value of 2 degrees. To address this, the mesh is adapted at $\alpha = 0$ while maintaining a fixed number of degrees of freedom, focusing on reorienting elements to better capture flow characteristics. This adaptation results in a modified mesh, depicted in Figure (12), composed of 370 elements.

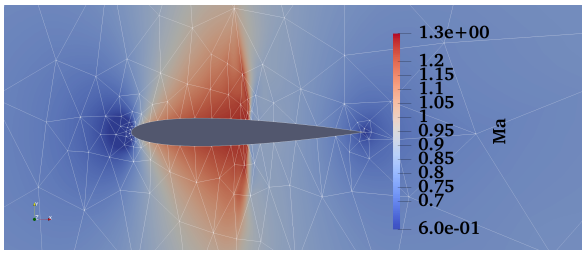


Figure 12: Mach number plot on the adapted mesh for $\alpha = 0$

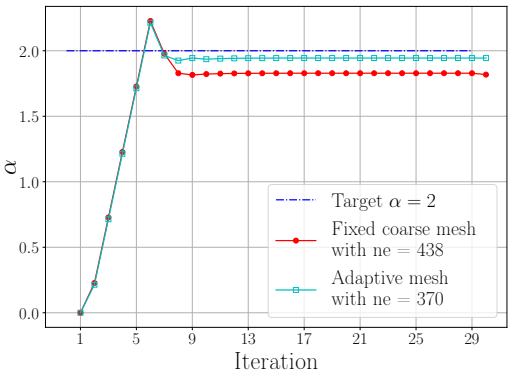


Figure 13: Evolution of the design variable α for coarse and adapted meshes

Figure (13) compares the evolution of α during optimization iterations between the coarse mesh and the adapted mesh for $\alpha = 0$. The adapted mesh demonstrates convergence towards $\alpha = 1.944$ degrees, closer to the target value compared to the coarse mesh but still not achieving it. However, as α increases, the adaptation designed for $\alpha = 0$ becomes less effective in reducing errors, highlighting the need for multiple adapted meshes in optimization. Subsequently, the mesh is adapted a second time for $\alpha = 1.944$, resulting in a slight increase in the number of elements to 553. Using this further adapted mesh, optimization proceeds and converges closely to the target value, illustrated in Figure (14). The

distribution of C_p for this optimal α compared to the target value is shown in Figure (15).

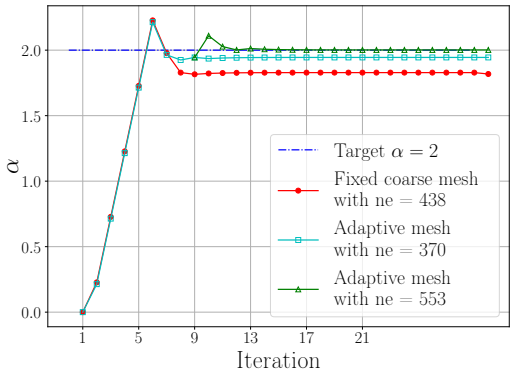


Figure 14: Evolution of the design variable α using multiple adapted meshes

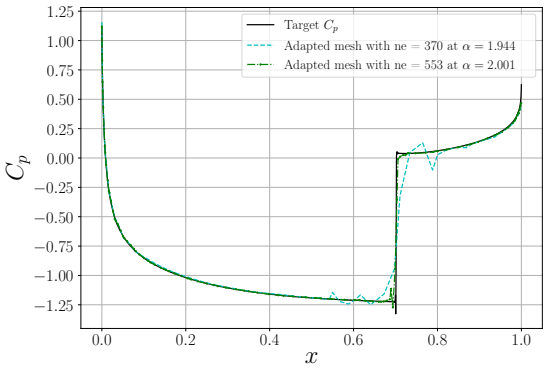


Figure 15: C_p plot on the upper surface of the airfoil

6.3 Summary and future work

This paper implements a shape optimization module within a Hybridized Discontinuous Galerkin (HDG) solver, utilizing adapted meshes to control discretization errors. We apply shape optimization to solve the ADODG benchmark case-1 and demonstrate the necessity of using adapted meshes through an inverse design test case. Due to variations in the flow field throughout the optimization cycle, a single adapted mesh becomes inadequate for effectively capturing changed flow features. This necessitates multiple mesh adaptations within optimization cycles. However, using adapted meshes introduces additional computational costs associated with generating the adapted meshes. Therefore, adapting the mesh at every iteration is not computationally viable. Consequently, strategies must be devised to automatically determine optimal instances within the optimization cycle for implementing mesh adaptation. A few such strategies are explored in the literature [21, 22, 25]. For future research, we intend to develop a strategy to decide over adaptation and shape optimization utilizing the mesh adaptation algorithm proposed by Rangarajan et al. [31] within the HDG framework.

References

[1] Antony Jameson. Automatic design of transonic airfoils to reduce reduce the shock induced pressure drag. *Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics, Tel Aviv*, 01 1990.

[2] Antony Jameson. Aerodynamic inverse design and shape optimization via control theory. *Florida State University Lecture Series February 15-20*, 2010.

[3] Shaun Skinner and Hossein Zare-Behtash. State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing*, 62, 09 2017.

[4] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, 2021.

[5] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1):97 – 110, 1974. Cited by: 586.

[6] Jameson A. Aerodynamic shape optimization using the adjoint method. *VKI lecture series*, 2003.

[7] Joaquim Martins. Aerodynamic design optimization: Challenges and perspectives. *Computers & Fluids*, 239:105391, 03 2022.

[8] Bernardo Cockburn, George Karniadakis, and Shu (Eds. *Discontinuous Galerkin Methods: Theory, Computation and Application*, volume 11. Springer Berlin, Heidelberg, 12 2000.

[9] Bernardo Cockburn, Jay Gopalakrishnan, and Raytcho D. Lazarov. Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems. *SIAM J. Numer. Anal.*, 47:1319–1365, 2009.

[10] Jaime Peraire, Ngoc Nguyen, and Bernardo Cockburn. A hybridizable discontinuous galerkin method for the compressible euler and navier-stokes equations. *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[11] Z. Wang, Krzysztof Fidkowski, Rémi Abgrall, Francesco Bassi, Doru Caraeni, Andrew Cary, H. Deconinck, Ralf Hartmann, Koen Hillewaert, H.T. Huynh, Norbert Kroll, Georg May, Per-Olof Persson, Bram van Leer, and Miguel Visbal. High-order cfd methods: Current status and perspective. *International Journal for Numerical Methods in Fluids*, 72:811–845, 01 2013.

[12] H.T. Huynh, Z.J. Wang, and P.E. Vincent. High-order methods for computational fluid dynamics: A brief review of compact differential formulations on unstructured grids. *Computers & Fluids*, 98:209–220, 2014.

[13] Elijah Polak. *Optimization: Algorithms and Consistent Approximations*. Springer New York, NY, 1997.

[14] Olivier Pironneau and Elijah Polak. Consistent approximations and approximate functions and gradients in optimal control. *SIAM J. Control. Optim.*, 41:487–510, 2002.

[15] Andrea Dadone and Bernard Grossman. Progressive optimization of inverse fluid dynamic design problems. *Computers & Fluids*, 29:1–32, 2000.

[16] Andrea Dadone and Bernard Grossman. Fast convergence of viscous airfoil design problems. *Aiaa Journal - AIAA J*, 40:1997–2005, 10 2002.

[17] Andrea Dadone and Bernard Grossman. Fast convergence of inviscid fluid dynamic design problems. *Computers & Fluids*, 32:607–627, 05 2003.

[18] Roland Becker and Rolf Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1 – 102, 2001.

[19] Michael B. Giles and Endre Süli. Adjoint methods for pdes: a posteriori error analysis and postprocessing by duality. *Acta Numerica*, 11:145 – 236, 2002.

[20] Krzysztof J. Fidkowski and David L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal*, 49:673–694, 2011.

[21] James Lu. An a posteriori error control framework for adaptive precision optimization using discontinuous galerkin finite element method. *Ph.D. thesis, Massachusetts Institute of Technology*, 2005.

[22] Marian Nemec and Michael Aftosmis. Output error estimates and mesh refinement in aerodynamic shape optimization. *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013.

[23] Derek J. Dalle and Krzysztof Fidkowski. Multifidelity airfoil shape optimization using adaptive meshing. *32nd AIAA Applied Aerodynamics Conference*, 2014.

[24] D. Li and R. Hartmann. Adjoint-based airfoil optimization with discretization error control. *International Journal for Numerical Methods in Fluids*, 77(1):1–17, 2015.

[25] Guodong Chen and Krzysztof J. Fidkowski. Discretization error control for constrained aerodynamic shape optimization. *Journal of Computational Physics*, 387:163–185, 2019.

[26] J.E. Hicken and J.J. Alonso. Pde-constrained optimization with error estimation and control. *Journal of Computational Physics*, 263:136–150, 2014.

[27] Alexander Coppeans, Krzysztof Fidkowski, and Joaquim R. Martins. Comparison of finite volume and high order discontinuous galerkin based aerodynamic shape optimization. *AIAA SCITECH 2023 Forum*, 2023.

[28] Joachim Balis, Frederik Jacobs, and Georg May. Adjoint-based aerodynamic shape optimization with hybridized discontinuous galerkin methods. *Computers & Fluids*, 11 2023.

[29] Michael Woopen, Aravind Balan, Georg May, and Jochen Schütz. A comparison of hybridized and standard dg methods for target-based hp-adaptive simulation of compressible flow. *Computers & Fluids*, 98:3–16, 2014. 12th USNCCM mini-symposium of High-Order Methods for Computational Fluid Dynamics.

[30] Michael Woopen, Aravind Balan, and Georg May. A unifying computational framework for adaptive high-order finite element methods. *22nd AIAA Computational Fluid Dynamics Conference*, 2015.

[31] Ajay Rangarajan, Georg May, and Vit Dolejsi. Adjoint-based anisotropic hp-adaptation for discontinuous galerkin methods using a continuous mesh model. *Journal of Computational Physics*, 409:109321, 2020.

[32] Ruben E. Perez, Peter W. Jansen, and Joaquim R. R. A. Martins. pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. *Structures and Multidisciplinary Optimization*, 45(1):101–118, 2012.

[33] Ella Wu, Gaetan Kenway, Charles A. Mader, John Jasa, and Joaquim R. R. A. Martins. pyoptsparse: A python framework for large-scale constrained nonlinear optimization of sparse systems. *Journal of Open Source Software*, 5(54):2564, 2020.

[34] Siva Nadarajah. Aerodynamic design optimization: Drag minimization of the naca 0012 in transonic

inviscid flow.

[35] Siva Nadarajah and Antony Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *: Proceedings of the 38th AIAA Aerospace Sciences Meeting and exhibit*, 11 2014.

[36] Ney R. Secco, Gaetan K. W. Kenway, Ping He, Charles Mader, and Joaquim R. R. A. Martins. Efficient mesh generation and deformation for aerodynamic shape optimization. *AIAA Journal*, 59(4):1151–1168, 2021.