Variable-fidelity Multipoint Aerodynamic Shape Optimization with Output-based Adapted Meshes

Guodong Chen[a,*], Krzysztof J. Fidkowski[a]

[a] *Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109*

Abstract

This paper presents a method to control the discretization error in multipoint aerodynamic shape optimization using output-based adapted meshes. The meshes are adapted via adjoint-based error estimates, taking into account both the objective and constraint output errors. A multi-fidelity optimization framework is then developed by taking advantage of the variable fidelity offered by adaptive meshes. The objective functional and its sensitivity at each design point (operating condition) are first evaluated on the same initial coarse mesh, which is then subsequently adapted for each design point individually as the shape optimization proceeds. The effort to set up the optimization is minimal since the initial mesh can be fairly coarse and easy to generate. As the shape approaches the optimal design, the mesh at each design point becomes finer, in regions necessary for that particular operating condition. The multi-fidelity framework is tightly coupled with the objective error estimation to ensure the optimization accuracy at each fidelity. Computational savings arise from a reduction of the mesh size when the design is far from optimal and avoiding an exhaustive search on low-fidelity meshes. The proposed method is demonstrated on multipoint drag minimization problems of a transonic airfoil with lift and volume constraints. Improved accuracy and efficiency are shown compared to traditional fixed-fidelity optimization with a fixed computational mesh.

Keywords: Multipoint optimization, Variable-fidelity optimization, Discretization error, Adjoint-based error estimation, Mesh adaptation

1. Introduction

Over the past several decades, Computational Fluid Dynamics (CFD) has become increasingly prevalent in aerospace design and analysis. The fast turnaround time, high degree of geometric flexibility, and almost arbitrary test conditions offered by CFD have made it an attractive tool in aerodynamic design process. Successful use of CFD in practical design problems requires both accurate simulations for a given configuration and efficient optimization methods to improve design configurations. Gradient-free methods such as genetic algorithms may be made robust for non-smooth or non-convex problems [1], but they are generally not as efficient as gradient-based methods, especially for problems with a large number of design parameters. Specifically, gradient-based algorithms converge to the optimum with fewer evaluations of the objective function and lower cost, even when taking into account the gradient calculations. Moreover, with the development of adjoint-based sensitivity analysis [2, 3, 4, 5, 6, 7, 8], the computational cost of gradient-based optimization has been dramatically reduced.

Ideally, the design is expected to retain favorable performance over a wide range of operating conditions. Optimization at one specific cruise condition can lead to mediocre performance on the overall mission profile, and/or poor performance at off-design conditions. Therefore, aerodynamic optimization in practice must take into account various flight conditions in both the objective and constraints [9, 10, 11, 12, 13], making

[*] Corresponding author
Email address: cgderic@umich.edu (Guodong Chen)

the setup for high-fidelity aerodynamic optimization challenging. In order to achieve high accuracy for every design point, the single mesh used for all points has to be able to capture all of the important flow features over a wide range of operating conditions. The designer, either expert in meshing or not, cannot reliably generate a mesh appropriate for all cruise conditions. This situation can be even worse when the geometry is complex or as the number of design points increases. Also, the resulting mesh may be quite fine, making the computational cost needed for high-fidelity multipoint aerodynamic optimization prohibitive in practice. On the other hand, numerical errors are typically only investigated via grid convergence studies for the initial and final designs, before and after the optimization, which can potentially lead to inaccurate or spurious optima [14, 15]. These are the problems that we tackle in the present work.

In order to aid practical multipoint aerodynamic design and to reduce the optimization cost, automated adapted meshes are introduced into the design process using gradient-based optimization algorithms. The designer only needs to provide a relatively coarse background mesh to start the optimization run. Then, the computational mesh is adapted individually in necessary regions based on the output error estimates, with active control of numerical error at various operating conditions. The fact that both the output error estimation and objective sensitivity calculations rely on adjoint solutions makes the incorporation of mesh adaptation into optimization more efficient. After showing many successes in a wide range of aerospace computational applications [16, 17, 18, 19, 20, 21, 22, 23], output-based error estimation and mesh adaptation methods have been demonstrated in several single-point aerodynamic shape optimization problems [24, 25, 26, 27, 15, 28]. However, to the authors' knowledge, no attention has been paid to more sophisticated multipoint optimization problems.

The present work proposes a variable-fidelity implementation of multipoint aerodynamic optimization, integrating output-based error estimation and mesh adaptation with a gradient-based algorithm to actively control the numerical error during optimization. We adopt the error estimation and mesh adaptation strategy developed in our previous work [15], taking the errors in the constraint outputs into account during error estimation and mesh adaptation, since the errors in the constraints can indirectly affect the calculation of the objectives [29]. Two unstructured mesh adaptation methods are considered in this paper: (a) mesh adaptation with Hessian-based anisotropy, and (b) mesh optimization via error sampling and synthesis (MOESS). In addition to the time saving on the optimization setup, this method also has the potential to reduce the run-time computational cost of the optimization. As the physics of various operating conditions can differ substantially, the meshes required to accurately predict the outputs also differ. With the proposed method, meshes for the points whose physics are relatively simple (e.g. low speed, laminar flow) can be coarse, whereas substantial mesh refinement can be added to those points governed by more complex physics (e.g. shocks, turbulent flows). Furthermore, the refinement can differ among the complex physics points, e.g. as shocks and wakes move to different positions. A variable-fidelity optimization framework is built into the proposed method, taking advantage of the variable fidelity offered by adaptive meshes. The variable-fidelity framework reduces the computational cost when the shape is far from the optimum, thus avoiding over-refining on an undesired configuration. On the other hand, the error estimation prevents optimization directions from being polluted by discretization errors and over-optimization on a coarse mesh.

The remainder of this paper proceeds as follows. We describe the general aerodynamic optimization problem in Section 2 and the discontinuous Galerkin discretization in Section 3. Details of the error estimation and mesh adaptation are given in Section 4 and Section 5. Section 6 presents the coupling of gradient-based optimization with error estimation and mesh adaptation. The primary results are shown in Section 7, and Section 8 concludes the present work and discusses potential future work.

2. Problem Formulation

2.1. Multipoint Aerodynamic Optimization

In general, the aerodynamic shape optimization problem can be stated as a search for the design variables $\mathbf{x}$ over the design space $\mathcal{X}$ that minimize a given objective function J:

$$\begin{aligned} \min_{\mathbf{x}} \quad & J(\mathbf{U},\mathbf{x}), \quad \mathbf{U}\in\mathcal{U},\ \mathbf{x}\in\mathcal{X}, \\ \text{s.t.} \quad & \mathbf{R}^{\mathrm{e}}(\mathbf{U},\mathbf{x}) = \mathbf{0}, \\ & \mathbf{R}^{\mathrm{ie}}(\mathbf{U},\mathbf{x}) \geq \mathbf{0}, \end{aligned} \tag{1}$$

where $J : \mathcal{U}\times\mathcal{X} \rightarrow \mathbb{R}$ represents a scalar objective function, always defined by aerodynamic outputs, for example lift or drag or a combination of these for multi-objective optimization. $\mathbf{U} \in \mathcal{U} \subset \mathbb{R}^{N_f}$ denotes the flow variable vector of dimension N_f, and $\mathbf{R}^{\mathrm{e}} : \mathcal{U}\times\mathcal{X} \rightarrow \mathbb{R}^{N_{\mathrm{e}}}$ and $\mathbf{R}^{\mathrm{ie}} : \mathcal{U}\times\mathcal{X} \rightarrow \mathbb{R}^{N_{\mathrm{ie}}}$ are the N_{e} equality and N_{ie} inequality constraints, respectively. The flow variables $\mathbf{U}$ are solved within a feasible space $\mathcal{U}$ given a design $\mathbf{x}$ to satisfy the flow equations, often the Euler or Navier-Stokes equations. In discretized form, these can be represented by a set of nonlinear equations,

$$\mathbf{R}(\mathbf{U},\mathbf{x}) = \mathbf{0}, \quad \mathbf{U}\in\mathcal{U},\ \forall\ \mathbf{x}\in\mathcal{X}, \tag{2}$$

where $\mathbf{R} : \mathcal{U}\times\mathcal{X} \rightarrow \mathbb{R}^{N_f}$ is the nonlinear flow residual vector, which when driven to zero implicitly defines $\mathbf{U}$ as a function of $\mathbf{x}$.

Consider a multipoint optimization problem involving N_m design points (typically various Mach numbers). To combine the objective outputs, the weighted-sum method is used in this work,

$$J_m = \sum_{i=1}^{N_m} \omega_i J_i(\mathbf{U}_i,\mathbf{x}) = \sum_{i=1}^{N_m} J_{m,i}, \tag{3}$$

where J_m is the scalar composite objective used in the optimization, a sum of the objective component $J_{m,i}$ at each design point. $J_{m,i}$ is defined as the objective at the i^{th} design point $J_i(\mathbf{U}_i,\mathbf{x})$ weighted by ω_i, which is specified by the user or from the quadrature rules. The objective at each point $J_i(\mathbf{U}_i,\mathbf{x})$ depends on the design $\mathbf{x}$ shared by all the design points and the individual flow state solution $\mathbf{U}_i \in \mathbb{R}^{N_{f,i}}$.

2.2. Adjoint and Design Equations

Inactive inequality constraints $\mathbf{R}^{\mathrm{ie}}_{\mathrm{ia}}(\mathbf{U},\mathbf{x})$, do not affect the optimization explicitly, while the active ones $\mathbf{R}^{\mathrm{ie}}_{\mathrm{a}} = \mathbf{0}$ behave as equality constraints. In general, the inequality constraints can also be transformed into equality constraints with non-negative slack variables [30]. For easier illustration, we only consider the active inequality constraints and equality constraints, put together into one vector of dimension N_t as trim constraints, $(\mathbf{R}^{\mathrm{trim}})^T = [(\mathbf{R}^{\mathrm{e}})^T\ (\mathbf{R}^{\mathrm{ie}}_{\mathrm{a}})^T] \in \mathbb{R}^{N_t}$,

$$\mathbf{R}^{\mathrm{trim}}(\mathbf{U},\mathbf{x}) = \mathbf{J}^{\mathrm{trim}}(\mathbf{U},\mathbf{x}) - \bar{\mathbf{J}}^{\mathrm{trim}} = \mathbf{0}, \tag{4}$$

where $\bar{\mathbf{J}}^{\mathrm{trim}}$ is a set of target trim outputs, for example, the target lift in a lift-constrained problem. In multipoint optimization problems, each operating condition can have the same or different trim constraints. $\mathbf{J}^{\mathrm{trim}}$ is a vector concatenated with the trim outputs at different operating conditions $\mathbf{J}^{\mathrm{trim}}_i \in \mathbb{R}^{N_{t,i}}$, $\sum_i^{N_m} N_{t,i} = N_t$. In order to distinguish the trim outputs from the objective output, we denote the latter by $J^{\mathrm{adapt}}_m = \sum_i^{N_m} \omega_i J^{\mathrm{adapt}}_i = \sum_i^{N_m} J^{\mathrm{adapt}}_{m,i}$, as the objective output is often the direct target of adaptation.

The adjoint-based optimization is equivalent to searching for the stationary point of the Lagrangian function, which augments the flow equations with additional constraints,

$$\mathcal{L} = \sum_{i=1}^{N_m} \omega_i J^{\mathrm{adapt}}_i(\mathbf{U}_i,\mathbf{x}) + \sum_{i=1}^{N_m} \boldsymbol{\lambda_i}^T \mathbf{R}_i(\mathbf{U}_i,\mathbf{x}) + \sum_{i=1}^{N_m} \boldsymbol{\mu_i}^T \mathbf{R}^{\mathrm{trim}}_i(\mathbf{U}_i,\mathbf{x}), \tag{5}$$

where $\mathbf{R}_i(\mathbf{U}_i, \mathbf{x})$ are the flow equations at each design point, and $\boldsymbol{\lambda}_i \in \mathbb{R}^{N_{f,i}}$ and $\boldsymbol{\mu}_i \in \mathbb{R}^{N_{t,i}}$ are the Lagrange multipliers associated with the flow equations and the trim constraints, respectively. The first-order optimality conditions are obtained by setting the partial derivatives of $\mathcal{L}$ to zero,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}} = \mathbf{0}, \tag{6}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_i} = \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{U}_i} + \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i} + \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{U}_i} = \mathbf{0}, \qquad i = 1, ..., N_m; \tag{7}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}_i} = \mathbf{R}_i(\mathbf{U}_i, \mathbf{x}) = \mathbf{0}, \qquad i = 1, ..., N_m; \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_i} = \mathbf{R}_i^{\text{trim}}(\mathbf{U}_i, \mathbf{x}) = \mathbf{0}, \qquad i = 1, ..., N_m. \tag{9}$$

As we solve the flow equations for a given design $\mathbf{x}$ each time at every design point, Eqn. 8 is always satisfied during the optimization. The trim constraints can be handled by either the flow solver or the optimizer; the former is used currently. A set of design variables is dedicated to satisfying the trim constraints, denoted as trim variables $\mathbf{x}_t$, $\dim(\mathbf{x}_t) = \dim(\mathbf{J}^{\text{trim}}) = N_t$. Hence Eqn. 9 is satisfied by the variation of the trim variables, so that Eqn. 6 breaks down into,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_s} = \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_s} = \mathbf{0}, \tag{10}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_t} = \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_t} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_t} = \mathbf{0}, \tag{11}$$

where $\mathbf{x}_s$ is the set of active design parameters in the optimization. We can now choose $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ at each point such that Eqn. 7 and Eqn. 11 are enforced after each flow solve,

$$\begin{aligned} \boldsymbol{\lambda}_i^T &= -\left(\omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{U}_i} + \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{U}_i}\right)\left(\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right)^{-1} = (\omega_i \boldsymbol{\Psi}_i^{\text{adapt}} + \boldsymbol{\Psi}_i^{\text{trim}} \boldsymbol{\mu}_i)^T, \\ \boldsymbol{\mu}^T &= [\boldsymbol{\mu}_1^T, ..., \boldsymbol{\mu}_{N_m}^T] = -\left(\sum_{i=1}^{N_m} \omega_i \frac{dJ_i^{\text{adapt}}}{d\mathbf{x}_t}\right)\left(\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t}\right)^{-1}. \end{aligned} \tag{12}$$

There may exist a set of trim variables which can eliminate the coupling of the trim constraints among different design points (i.e. make $d\mathbf{J}^{\text{trim}}/d\mathbf{x}_t$ diagonal), so that $\boldsymbol{\mu}_i$ only depends on the i^{th} design point. In lift-constrained optimization problems, angle of attack is such a choice. Eqn. 12 defines coupled adjoint variables $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ that incorporate the adjoints of both the objective and the trim outputs at each design point, $\boldsymbol{\Psi}_i^{\text{adapt}} \in \mathbb{R}^{N_{f_i} \times 1}$ and $\boldsymbol{\Psi}_i^{\text{trim}} \in \mathbb{R}^{N_{f,i} \times N_{t,i}}$, which satisfy

$$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right]^T \boldsymbol{\Psi}_i^{\text{adapt}} + \left[\frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{U}_i}\right]^T = \mathbf{0}, \qquad \left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right]^T \boldsymbol{\Psi}_i^{\text{trim}} + \left[\frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{U}_i}\right]^T = \mathbf{0}. \tag{13}$$

In Eqn. 12, $d(\cdot)/d\mathbf{x}_t$ terms are defined as,

$$\begin{aligned} \frac{dJ_i^{\text{adapt}}}{d\mathbf{x}_t} &= \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_t} + \left(\boldsymbol{\Psi}_i^{\text{adapt}}\right)^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t}, \\ \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} &= \frac{\partial \mathbf{J}^{\text{trim}}}{\partial \mathbf{x}_t} + \left(\boldsymbol{\Psi}_i^{\text{trim}}\right)^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t}. \end{aligned} \tag{14}$$

The total derivative symbols are used in a sense that the outputs only depend on the trim variables $\mathbf{x}_t$ when the active design parameters $\mathbf{x}_s$ are held constant; i.e., the total derivative is defined in a sub-problem varying $\mathbf{x}_t$ to satisfy the constraints given a fixed $\mathbf{x}_s$. For our optimization problem, $\mathbf{x}_t$ depends on $\mathbf{x}_s$, and the total derivatives should only be defined with respect to the active design parameters $\mathbf{x}_s$. Such derivatives are denoted by $D(\cdot)/D\mathbf{x}_s$ for later use.

With the specific choice of $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ in Eqn. 12, we can evaluate the objective gradients, i.e., the total derivatives in optimization, with respect to the active design variables $\mathbf{x}_s$ via the Lagrangian function, starting with Eqn. 10,

$$\begin{aligned}
\frac{DJ_m^{\text{adapt}}}{D\mathbf{x}_s} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_s} &= \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\lambda}_i^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_s} \\
&= \sum_{i=1}^{N_m} \omega_i \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \omega_i (\boldsymbol{\Psi}_i^{\text{adapt}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \left[\frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}_i^{\text{trim}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} \right] \\
&= \sum_{i=1}^{N_m} \omega_i \left[\frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}_i^{\text{adapt}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} \right] + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \left[\frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{x}_s} + (\boldsymbol{\Psi}_i^{\text{trim}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_s} \right] \\
&= \sum_{i=1}^{N_m} \omega_i \frac{dJ_i^{\text{adapt}}}{d\mathbf{x}_s} + \sum_{i=1}^{N_m} \boldsymbol{\mu}_i^T \frac{d\mathbf{J}_i^{\text{trim}}}{d\mathbf{x}_s} \\
&= \sum_{i=1}^{N_m} \left(\omega_i \frac{dJ_i^{\text{adapt}}}{d\mathbf{x}_s} + \boldsymbol{\mu}_i^T \frac{d\mathbf{J}_i^{\text{trim}}}{d\mathbf{x}_s} \right).
\end{aligned} \tag{15}$$

Similar to Eqn. 14, $d(\cdot)/d\mathbf{x}_s$ is the sensitivity measured with respect to the active design variables $\mathbf{x}_s$ while keeping the trim variables $\mathbf{x}_t$ fixed.

Now the optimization problem has been reduced to finding an optimal design $\mathbf{x}_s$ that drives to zero the gradients in Eqn. 15. However, in a practical calculation, on a finite-dimensional space, discretization errors appear in both the flow equations and the adjoint equations, so that infinite-dimensional optimality cannot be guaranteed even when the finite-dimensional optimality condition is satisfied. The present work focuses on controlling the error in the optimization problem via error estimation and mesh adaptation.

3. Discretization

Evaluation of the objective function at each optimization step relies on a flow simulation, in this work over an airfoil. The governing equations for the fluid flow are compressible Navier-Stokes,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}) - \nabla \cdot \vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}), \tag{16}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the conservative flow state vector of rank s, $\vec{\mathbf{F}}$ and $\vec{\mathbf{G}}$ denote the inviscid and viscous fluxes respectively, and $\mathbf{S}$ represents the source term required when modeling turbulence. The viscous flux is assumed to be linear on the state gradients, $\vec{\mathbf{G}}_i(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{K}_{i,j}(\mathbf{u})\partial_j \mathbf{u}$, where $\mathbf{K}_{i,j}$ denotes the diffusivity tensor. When running Reynolds-averaged turbulent cases, we use the Spalart-Allmaras one-equation model, with a negative turbulent-viscosity modification [31].

We discretize Eqn. 16 with the discontinuous Galerkin (DG) finite-element method, which is suitable for high-order accuracy and *hp*-refinement [32, 33, 34, 17, 35]. However, the framework proposed in this work can be applied to other discretizations supporting output-based error estimation and mesh adaptation. Consider a partition $\mathcal{T}_h$ of the computational domain Ω consisting of N_e non-overlapping elements Ω_e, $\mathcal{T}_h = \{\Omega_e : \cup \Omega_e = \Omega, \cap \Omega_e = \emptyset\}$. The state is approximated by piece-wise polynomials lying on the approximation space $\boldsymbol{\mathcal{V}}_h$, with no continuity constraints on the approximation between adjacent elements. Formally, the approximation space is defined as $\boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where $\mathcal{V}_h = \{v \in L^2(\Omega) : v|\Omega_e \in \mathcal{P}^p, \forall \Omega_e \in \mathcal{T}_h\}$, and $\mathcal{P}^p$ denotes polynomials of order p on the reference space of element Ω_e. The weak form of Eqn. 16 follows from multiplying the equation by test functions (taken from the same approximation space), integrating by parts, and coupling elements via unique inter-element fluxes,

$$\begin{aligned}\int_{\Omega_e} \mathbf{w}_h^T \frac{\partial \mathbf{u}}{\partial t} d\Omega - \int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \left[\vec{\mathbf{F}}(\mathbf{u}_h) - \vec{\mathbf{G}}(\mathbf{u}_h, \nabla \mathbf{u}_h)\right] d\Omega \\ + \int_{\partial\Omega_e} \mathbf{w}_h^T \left[\widehat{\mathbf{F}}(\mathbf{u}_h^+, \mathbf{u}_h^-) - \widehat{\mathbf{G}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla \mathbf{u}_h^+, \nabla \mathbf{u}_h^-)\right] \cdot \vec{n} dS \\ - \int_{\partial\Omega_e} (\mathbf{u}_h^+ - \{\mathbf{u}_h\})^T \vec{\mathbf{G}}(\mathbf{u}_h^+, \nabla \mathbf{w}_h^+) \cdot \vec{n} dS = \int_{\Omega_e} \mathbf{w}_h^T \mathbf{S}(\mathbf{u}_h, \nabla \mathbf{u}_h) d\Omega \qquad \forall \mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h.\end{aligned} \tag{17}$$

On the element boundary Ω_e, $(\cdot)^+, (\cdot)^-$ denote respectively the quantities taken from the element or its neighbor, $\{\cdot\}$ is the face/edge average or the boundary value, and $\hat{(\cdot)} \cdot \vec{n}$ represents the uniquely defined normal numerical flux on element interfaces. We use the Roe approximate Riemann solver [36] for the inviscid flux $\widehat{\mathbf{F}}$, while for the viscous flux $\widehat{\mathbf{G}}$ we use the second form of Bassi and Rebay [37]. The last term on the left-hand side (LHS) of Eqn. 17 symmetrizes the weak form and ensures adjoint consistency.

In this paper, we focus on steady state systems, so that $\frac{\partial \mathbf{u}}{\partial t}$ is omitted for later exposition. Choosing a basis for the test and trial spaces, the DG weak form in Eqn. 17 yields a system of discrete algebraic equations in the form of Eqn. 2,

$$\mathbf{R}_h(\mathbf{U}_h, \mathbf{x}) = \mathbf{0}, \tag{18}$$

where $\mathbf{U}_h \in \mathbb{R}^{N_h}$ is the discrete state vector of basis function coefficients of dimension N_h, and $\mathbf{R}_h$ is the discrete residual vector, a nonlinear function of the state vector $\mathbf{U}_h$ and the design variables $\mathbf{x}$. The subscript h refers to fidelity of the approximation/test space with respect to the approximation order and mesh refinement.

4. Output Error Estimation

4.1. Adjoint-based Error Estimation

In practice it is generally not possible to obtain the true numerical error for an output, whereas the difference between a coarse space and fine space solution often serves as an acceptable surrogate,

$$\text{output error: } \delta J \equiv J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h). \tag{19}$$

In this expression, J represents the output of interest, and the subscripts h and H denote the fine and coarse spaces, respectively. In the present work, the fine space is achieved by increasing the elements' approximation order p, to $p + 1$. We do not solve the nonlinear fine-space flow problem for the error prediction, but we instead use the linear fine-space adjoint solution, $\boldsymbol{\Psi}_h$, defined as the sensitivity of the output to the residual (see Eqn. 13). The adjoint weights the residual perturbation to produce an output perturbation [20],

$$\begin{aligned}\delta J &= J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) = J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \\ &\approx -\boldsymbol{\Psi}_h^T[\mathbf{R}_h(\mathbf{U}_h^H) - \mathbf{R}_h(\mathbf{U}_h)] = -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H),\end{aligned} \tag{20}$$

where $\mathbf{U}_h$ is the (hypothetical) exact solution on the fine space, $\mathbf{U}_h^H$ is the state injected into the fine space from the coarse one, which generally will not give a zero fine space residual, $\mathbf{R}_h(\mathbf{U}_h^H) \neq \mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}$. The derivation of Eqn. 20 originates from small perturbation assumption and is valid for outputs whose definition does not change between the coarse and fine spaces, $J_H(\mathbf{U}_H) = J_h(\mathbf{U}_h^H)$.

4.2. Output Error Estimation Under Trim Constraints

Normally, the error estimation is applied only to the output in which we are most interested, i.e., the objective. However, our optimization problem requires simultaneous solutions of flow equations and trim constraints. The numerical error of the trim outputs may indirectly affect the calculation of the objective. To take this effect into account, coupled adjoints should be used for the error estimates.

Consider a given design $\mathbf{x}_s$. The error in the objective comes from both the inexact solution $\mathbf{U}_H$ and the inexact trim constraints satisfaction (inexact trim variables $\mathbf{x}_{t,H}$). We can estimate the error in the composite objective using the linearization given by Eqn. 7 and Eqn. 11,

$$\begin{aligned}
\delta J_m^{\text{adapt}} &= \sum_{i=1}^{N_m} \omega_i \left[J_{H,i}^{\text{adapt}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H}) - J_{h,i}^{\text{adapt}}(\mathbf{U}_{h,i}, \mathbf{x}_{t,h}) \right] \\
&= \sum_{i=1}^{N_m} \omega_i \left[J_{h,i}^{\text{adapt}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - J_{h,i}^{\text{adapt}}(\mathbf{U}_{h,i}, \mathbf{x}_{t,h}) \right] \\
&= \sum_{i=1}^{N_m} \omega_i \left[\frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{U}_i} \delta \mathbf{U}_i + \frac{\partial J_i^{\text{adapt}}}{\partial \mathbf{x}_t} \delta \mathbf{x}_t \right] \\
&= -\sum_{i=1}^{N_m} \left[\boldsymbol{\lambda}_{h,i}^T \delta \mathbf{R}_{h,i} + \boldsymbol{\mu}_{h,i}^T \delta \mathbf{R}_{h,i}^{\text{trim}} \right] \\
&= -\sum_{i=1}^{N_m} \left[\boldsymbol{\lambda}_{h,i}^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) + \boldsymbol{\mu}_{h,i}^T \mathbf{R}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \right].
\end{aligned} \tag{21}$$

For the second term in Eqn. 21, we can expand the trim residual as

$$\begin{aligned}
\mathbf{R}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) &= \mathbf{J}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - \bar{\mathbf{J}}_i^{\text{trim}} \\
&= [\mathbf{J}_{H,i}^{\text{trim}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H}) - \bar{\mathbf{J}}_i^{\text{trim}}] + [\mathbf{J}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - \mathbf{J}_{H,i}^{\text{trim}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H})].
\end{aligned} \tag{22}$$

The first term above is automatically driven to zero because of the trimming on the coarse space. For the second term, again, if the definition of the trim outputs does not depend on the approximation space, then $\mathbf{J}_{h,i}^{\text{trim}}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) = \mathbf{J}_{H,i}^{\text{trim}}(\mathbf{U}_{H,i}, \mathbf{x}_{t,H})$. Hence, the second term in Eqn. 21 is often negligible, resulting a simpler form of the error estimate for the composite objective,

$$\begin{aligned}
\delta J_m^{\text{adapt}} &= -\sum_{i=1}^{N_m} \boldsymbol{\lambda}_{h,i}^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \\
&= -\sum_{i=1}^{N_m} (\omega_i \boldsymbol{\Psi}_{h,i}^{\text{adapt}} + \boldsymbol{\Psi}_{h,i}^{\text{trim}} \boldsymbol{\mu}_{h,i})^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \\
&= \sum_{i=1}^{N_m} \left[-\omega_i (\boldsymbol{\Psi}_{h,i}^{\text{adapt}})^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) - \boldsymbol{\mu}_{h,i}^T (\boldsymbol{\Psi}_{h,i}^{\text{trim}})^T \mathbf{R}_{h,i}(\mathbf{U}_{h,i}^H, \mathbf{x}_{t,H}) \right] \\
&= \sum_{i=1}^{N_m} (\omega_i \delta J_i^{\text{adapt}} + \boldsymbol{\mu}_{h,i}^T \delta \mathbf{J}_i^{\text{trim}}),
\end{aligned} \tag{23}$$

where $\delta J_i^{\text{adapt}}$ is the objective error and $\delta \mathbf{J}_i^{\text{trim}}$ stands for the trim output error, using standard adjoint-based error estimates (Eqn. 20). The weighting by $\boldsymbol{\mu}_{h,i}$ accounts for the effects of trim output error on the objective calculations.

5. Mesh Adaptation

If we would like to use the same mesh for all of the design points, then Eqn. 23 can be directly used to localize the error to each element, which then serves as the indicator for mesh adaptation. However, this can be inefficient when the flow features change significantly at different operating conditions, e.g., from subsonic to supersonic regimes. To achieve certain accuracy in such cases, the mesh should be adapted in the areas important for all of the design points, and hence unnecessary computational effort is added to each flow solve if using a single mesh.

In the present work, we allow different meshes for different design points. The objective error in Eqn. 23 is first localized to each design point as $\delta J_{m,i}^{\text{adapt}} = \omega_i \delta J_i^{\text{adapt}} + \boldsymbol{\mu}_{h,i}^T \delta \mathbf{J}_i^{\text{trim}}$. Then a common approach for obtaining an error indicator is to take the absolute value of the elemental error contribution. When trim outputs are involved, we do not allow cancellation between objective and trim output error indicators, so that the final error indicator on element e at flight condition i, $\epsilon_{i,e}$, is given by

$$\epsilon_{i,e} = \omega_i |\delta J_{i,e}^{\text{adapt}}| + |\boldsymbol{\mu}_i^T||\delta \mathbf{J}_{i,e}^{\text{trim}}| = \omega_i \epsilon_{i,e}^{\text{adapt}} + |\boldsymbol{\mu}_i^T| \boldsymbol{\epsilon}_{i,e}^{\text{trim}}, \tag{24}$$

where $\epsilon_{i,e}^{\text{adapt}}$ and $\boldsymbol{\epsilon}_{i,e}^{\text{trim}}$ are the non-negative error indicators for the objective output and the trim outputs respectively.

At each operating condition, the mesh adaptation can be performed as a refinement process to achieve certain accuracy, or as a modification or optimization process at a given cost to improve the accuracy. We will denote the former approach as error-based adaptation while the latter as cost-based adaptation. Although each adaptation strategy has been well-studied for fixed configurations, adaptation involving multiple operating conditions has seldom been investigated. In our implementation, mesh adaptation for multipoint flow simulations is a two-stage process: the desired error/cost is first allocated to each design point; common adaptation techniques are then applied to each design point. The sections below describe the error/cost allocation strategies and the adaptation methods adopted at individual design points.

5.1. Error/Cost Allocation for Multipoint Mesh Adaptation

In error-based mesh adaptation, a target error tolerance is specified to drive mesh refinement. Thus as a first step, we want to determine the error tolerance at each design point. This high-level error split relies on an error convergence model that dictates how the output errors behave with respect to changes in cost, usually measured by the system degrees of freedom, DOF. Here, an *a priori* error-cost model is assumed,

$$|\delta J_{m,i}^{\text{adapt}}| \propto C_i^{-r_i/d}, \tag{25}$$

where C_i is the cost at the i^{th} flight condition, r_i is the corresponding output convergence rate, and d is the spatial dimension. The convergence rate at each design point depends on the approximation order and the smoothness of the problem. We use the ideal super-convergent rate of outputs in an adjoint-consistent setting to prevent too aggressive refinement or coarsening (cost redistribution), though lower convergence rates should be expected for under-resolved meshes.

We follow the idea of equally distributing the error-to-cost ratios [38, 21], *i.e.*, the marginal error reduction per cost increase. This is considered optimal as we can otherwise further reduce the error without adding cost by just reallocating degrees of freedom among different design points. For one adaptive iteration, the change in the error due to cost redistribution is

$$\left|\delta J_{m,i}^{\text{adapt}}\right| = \left|\delta J_{m,i}^{\text{adapt},0}\right| \left(\frac{C_i}{C_i^0}\right)^{-r_i/d}, \tag{26}$$

where the superscript 0 indicates values in the unadapted meshes. The marginal error-to-cost ratio at each design point can be obtained by

$$\begin{aligned}\frac{\partial |\delta J_{m,i}^{\text{adapt}}|}{\partial C_i} &= -\frac{r_i}{d}\left|\delta J_{m,i}^{\text{adapt},0}\right| \left(\frac{C_i}{C_i^0}\right)^{-r_i/d-1} \frac{1}{C_i^0} \\ &= -\frac{r_i}{d}\left|\delta J_{m,i}^{\text{adapt},0}\right| \left(\frac{C_i}{C_i^0}\right)^{-r_i/d} \frac{1}{C_i} \\ &= -\frac{r_i}{d}\frac{|\delta J_{m,i}^{\text{adapt}}|}{C_i}, \qquad i = 1, ..., N_m.\end{aligned} \tag{27}$$

Further, we define the desired error ratios f_i^δ and cost ratios f_i^C as

$$f_i^\delta = \frac{\delta J_{m,i}^{\text{adapt}}}{\delta J_{m,1}^{\text{adapt}}}, \qquad f_i^C = \frac{C_i}{C_1}, \qquad i = 1, ..., N_m. \tag{28}$$

By equidistributing the error-to-cost ratios among different design points, the desired error ratios f_i^δ and cost ratios f_i^C can be determined using Eqn. 27. In this work, we further assume identical convergence rates ($r_i = r, i = 1, 2, ..., N_m$) if the same approximation order is used and the meshes are well-adapted. Eqn. 27 then implies that the desired error ratios and cost ratios are equal, and can be solved as

$$f_i^\delta = f_i^C = \left|\frac{\delta J_{m,i}^{\text{adapt},0}}{\delta J_{m,1}^{\text{adapt},0}}\right|^{\frac{1}{r/d+1}} \left[\frac{C_i^0}{C_1^0}\right]^{\frac{r/d}{r/d+1}}, \qquad i = 1, 2, ..., N_m. \tag{29}$$

The desired ratios in Eqn. 29 can then guide the error redistribution in error-based multipoint mesh adaptation, or cost redistribution in cost-based adaptation. Ideally, specifying desired cost ratios and error ratios are equivalent if the *a priori* error-cost model in Eqn. 25 is perfect, which is not the case in general. Furthermore, an error-based mesh adaptation procedure in practice does not focus on matching exactly the target error, since the *a posteriori* output error may deviate from the *a priori* estimation. Instead, an error tolerance τ_i is specified in the error-based adaptation, and the adaptation (refinement) reduces the *a posteriori* error until it is below the error tolerance, $|\delta J_{m,i}^{\text{adapt}}| \le \tau_i$. On the other hand, the desired cost can be specified in cost-based adaptation such that the adaptation redistributes the degrees of freedom on the computational mesh to reduce the error while keeping the cost fixed. Therefore, cost-based and error-based adaptation with equal desired ratios result in different meshes even if the same adaptation method is used.

- For cost-based multipoint mesh adaptation, given a fixed total cost C, the desired cost at each design point is then redistributed as,

$$C_i = \frac{f_i^C}{\sum_{j=1}^{N_m} f_j^C} C. \tag{30}$$

  At each design point, the mesh adaptation then refines areas more important for output prediction and coarsens elsewhere to keep the cost fixed (within some tolerance).

- In error-based multipoint mesh adaptation, given a total error tolerance τ of the composite objective J_m^{adapt}, we first distribute the tolerance to each design point according to the desired error ratios,

$$\tau_i = \frac{f_i^\delta}{\sum_{j=1}^{N_m} f_j^\delta} \tau. \tag{31}$$

  At each design point, the mesh is refined until the objective error component is below the error tolerance, $|\delta J_{m,i}^{\text{adapt}}| \le \tau_i$. Thus the cost at each design point C_i increases as the mesh is adapted, although the local adaptation may support coarsening.

5.2. Mesh Adaptation at Individual Design Points

At each design point, the mesh is adapted with either cost-based or error-based strategies. During the mesh adaptation, the error indicators in Eqn. 24 provide information of local refinement or coarsening. Meanwhile, the mesh elements should be properly stretched to effectively capture strong directional features. This mesh anisotropy information can come from heuristics, or from a sampling approach. The corresponding adaptation methods considered here are: (a) mesh adaptation with Hessian-based anisotropy detection, and (b) mesh optimization via error sampling and synthesis (MOESS). Both methods rely on metric-based global remeshing, in which the mesh information, including the desired element sizes and stretching directions, is encoded in a continuous Riemannian metric field. The desired metric field is determined by the elemental error indicator given in Eqn. 24, together with the local anisotropy information, obtained using a solution

Hessian in Hessian-based adaptation or through a sampling process in MOESS. Thus, the mesh adaptation at individual design points can be: error-based or cost-based with Hessian-based anisotropy or MOESS sampling-inferred anisotropy. However, MOESS often involves an optimization iterative process with fixed cost, making it inefficient in an error-based setting, thus error-based MOESS is not considered here. The following sections describe the adaptation algorithms in detail.

5.2.1. Metric-based Remeshing

A Riemannian metric field, $\boldsymbol{\mathcal{M}}(\vec{x}) \in \mathbb{R}^{d\times d}$, is a smooth field of symmetric positive definite (SPD) tensors that can be used to encode element size and stretching information of a mesh. The metric tensor, $\boldsymbol{\mathcal{M}}(\vec{x})$, provides a measure of distance between points; for any two spatial points a and b, the length of vector $\overrightarrow{ab}$ under the metric, $l_{\boldsymbol{\mathcal{M}}}(\overrightarrow{ab})$, is defined as

$$l_{\boldsymbol{\mathcal{M}}}(\overrightarrow{ab}) = \int_0^1 \sqrt{\overrightarrow{ab}^T \boldsymbol{\mathcal{M}}(\vec{x}_a + \overrightarrow{ab}s)\overrightarrow{ab}}\, ds, \quad \forall\ \vec{x}_a, \vec{x}_b \in \mathbb{R}^d. \tag{32}$$

A mesh that conforms to a metric field is one in which all of the edges have the same unit length under the metric, to some tolerance. The mesh adaptation takes the current mesh-implied metric field and outputs the desired metric field, followed by a remeshing process which generates a mesh that conforms to the desired metric field. The metric-conforming mesh generator used in this work is the Bi-dimensional Anisotropic Mesh Generator (BAMG) [39]. The mesh-implied elemental metric field is obtained by solving a linear system at each simplex element enforcing the unit edge length under the metric. Then the elemental metric field can be transferred to nodes using an affine-invariant averaging algorithm [40]. With modifications to the mesh-implied metric field, the desired metric field is specified at vertices, and this conforms to the input required by the mesh generator.

5.2.2. Hessian-based Mesh Adaptation

One dominant approach for detecting the anisotropy is to estimate the directional interpolation error of the solution [41, 42, 43], and we describe here an extension of such an approach that incorporates output error adaptive indicators [18, 44]. For linear approximation, *i.e.*, $p = 1$, the interpolation error of a scalar solution u over an edge E in the mesh, with unit tangent vector $\vec{s}$ and length h, can be approximated by

$$\delta_{u,E} \propto |\vec{s}^T \mathbf{H}\vec{s}|h^2, \tag{33}$$

where $\mathbf{H}$ is the solution Hessian matrix,

$$H_{i,j} = \frac{\partial^2 u}{\partial x_i \partial x_j}, \quad i, j \in \{1, ..., d\}. \tag{34}$$

The second derivatives can be estimated by a quadratic reconstruction of the linear solution. The scalar u used in this work is the Mach number as it has been found to be generally effective, although more sophisticated quantities can also be used [41]. Suppose an edge conform to a metric $\boldsymbol{\mathcal{M}}$, assumed constant along the edge. Then the edge is of unit length under the metric measure,

$$l_{\boldsymbol{\mathcal{M}}} = \sqrt{\vec{s}^T \boldsymbol{\mathcal{M}}\vec{s}}\, h = 1. \tag{35}$$

Thus the interpolation error and the metric are related by Eqn. 33 and Eqn. 35, and with the requirements of error equidistribution along edges, we have

$$\frac{\vec{s}^T \boldsymbol{\mathcal{M}}\vec{s}}{|\vec{s}^T \mathbf{H}\vec{s}|} = \kappa, \tag{36}$$

where κ is a constant defined by the desired interpolation error. In order for Eqn. 36 to hold in any principal direction, $\boldsymbol{\mathcal{M}}$ can be chosen as

$$\boldsymbol{\mathcal{M}}_{\mathbf{H}} = \kappa \mathbf{Q}|\boldsymbol{\Lambda}_{\mathbf{H}}|\mathbf{Q}^T = \mathbf{Q}\boldsymbol{\Lambda}_{\boldsymbol{\mathcal{M}}}\mathbf{Q}^T. \tag{37}$$

Here, $\mathbf{Q}$ denotes the orthonormal matrix containing the eigenvectors (principal directions) of $\mathbf{H}$, and $\mathbf{\Lambda_H}$ and $\mathbf{\Lambda_{\mathcal{M}}}$ are the diagonal matrices containing the eigenvalues of the Hessian matrix $\mathbf{H}$ and the metric tensor $\boldsymbol{\mathcal{M}}$, respectively. Eqn. 37 implies the relative shape of the desired metric field and the mesh conforms to the metric,

$$\frac{\lambda_{\mathcal{M},i}}{\lambda_{\mathcal{M},j}} = \left|\frac{\lambda_{\mathbf{H},i}}{\lambda_{\mathbf{H},j}}\right| \quad \Longrightarrow \quad \frac{h_i}{h_j} = \sqrt{\frac{\lambda_{\mathcal{M},j}}{\lambda_{\mathcal{M},i}}} = \sqrt{\left|\frac{\lambda_{\mathbf{H},j}}{\lambda_{\mathbf{H},i}}\right|}, \tag{38}$$

where $\lambda_\mathbf{H}$ and $\lambda_\mathcal{M}$ represent the eigenvalues of $\mathbf{H}$ and $\boldsymbol{\mathcal{M}}$, respectively, i and j index the principal directions, and h_i and h_j are the desired element sizes in the corresponding directions. For higher-order approximations, the interpolation error is characterized by the $p + 1^{\text{st}}$ derivatives, while the first d largest directional derivatives are used to determine the principle directions $\mathbf{Q}$, and the corresponding stretching matrices $\mathbf{\Lambda_H}$ and $\mathbf{\Lambda_{\mathcal{M}}}$ [44]. The key idea of Hessian-based mesh adaptation with output error estimation is to use Eqn. 38 to control the mesh shape (anisotropy) while using the output error indicator to determine the absolute element sizes.

In order to perform mesh adaptation, we need to predict the desired element sizes, or the number of the elements N^f in the adapted (fine) mesh. Let n_e, not necessary an integer, be the number of adapted mesh elements contained in element e for the original mesh. Denoting the current element sizes by h_i^c and the requested element sizes by h_i^f, where i again indexes the principal directions; n_e can be approximated as

$$n_e = \prod_{i=1}^{d} (h_i^c / h_i^f). \tag{39}$$

We assume that on the adapted mesh, the output error is equally distributed in the mesh elements as ϵ^f. We can then relate the growth in the number of elements to an error reduction factor through an *a priori* estimate,

$$n_e \epsilon^f = \epsilon_e^c \left(\frac{h_{\text{ref}}^f}{h_{\text{ref}}^c}\right)^{\bar{p}_e+1} = \epsilon_e^c \left(\prod_{i=1}^{d} \frac{h_i^f}{h_i^c}\right)^{(\bar{p}_e+1)/d}, \tag{40}$$

where ϵ_e^c denotes the current error indicator in element e, $\bar{p}_e = \min(p, \gamma_e)$, and γ_e is the lowest order of any singularity within element e. h_{ref} is the reference element size that characterizes the convergence, defined in this work as the geometric mean of the edge lengths at principle directions. Eqn. 39 and Eqn. 40 relate n_e to the desired equidistributed elemental error,

$$n_e \epsilon^f = \epsilon_e^c \left(\prod_{i=1}^{d} \frac{h_i^f}{h_i^c}\right)^{(\bar{p}_e+1)/d} = \epsilon_e^c n_e^{-(\bar{p}_e+1)/d} \quad \Longrightarrow \quad n_e^{1+(\bar{p}_e+1)/d} = \frac{\epsilon_e^c}{\epsilon^f}. \tag{41}$$

Substituting Eqn. 41 into $N^f = \sum_e n_e$, we can solve for both ϵ and n_e if N^f is given and $\bar{p}_e$ is assumed equal everywhere in the mesh,

$$\epsilon^f = \left(\frac{\sum_e (\epsilon_e^c)^{\frac{1}{1+(\bar{p}_e+1)/d}}}{N^f}\right)^{1+(\bar{p}_e+1)/d}, \qquad n_e = \left(\frac{\epsilon_e^c}{\epsilon^f}\right)^{\frac{1}{1+(\bar{p}_e+1)/d}}. \tag{42}$$

After obtaining n_e, the desired element size h_i^f can be calculated using Eqn. 39 with current element size information stored in the current mesh-implied metric. In this work, Hessian-based adaptation is made error-based or cost-based depending on how the total number of elements N^f is specified:

- Cost-based Hessian adaptation: a fixed total cost, *i.e.*, number of elements N^f is given. The mesh element sizes and stretching are updated iteratively to match the desired metric field.

- Error-based Hessian adaptation: an error tolerance on the output is given as τ. At each adaptive iteration, the current number of elements N^c is increased by a fixed-growth factor, *i.e.*, $N^f = f^{\text{growth}} N^c$, until the output error estimation is below the tolerance, $\epsilon \leq \tau$.

5.2.3. Mesh Optimization via Error Sampling and Synthesis

Mesh adaptation with Hessian-based anisotropy relies on a scalar solution u, which should be carefully chosen to correlate to the chosen output of interest. An inflection in u may lead to inappropriate mesh stretching, and inadequate resolution may occur where the magnitude of the Hessian is close to zero. Here we consider a more sophisticated mesh adaptation method: mesh optimization via error sampling and synthesis (MOESS). In MOESS, the mesh adaptation is formulated as an optimization problem in which the optimal change of the metric field is iteratively determined based on a prescribed metric-cost model and a sampling-inferred metric-error relationship. We briefly review this method and discuss its modification in this section.

5.2.3.1. Error Convergence Model

MOESS requires a model for how the error changes as the metric changes. At element Ω_e, the changes of the metric are described by a symmetric local step matrix $\boldsymbol{\mathcal{S}}_e \in \mathbb{R}^{d\times d}$,

$$\boldsymbol{\mathcal{M}}_e = \boldsymbol{\mathcal{M}}_{e,0}^{\frac{1}{2}} \exp(\boldsymbol{\mathcal{S}}_e) \boldsymbol{\mathcal{M}}_{e,0}^{\frac{1}{2}}, \tag{43}$$

where $\boldsymbol{\mathcal{M}}_e$ and $\boldsymbol{\mathcal{M}}_{e,0}$ denote the modified metric and the current one. The corresponding error change is given by a generalized error convergence model [38],

$$\epsilon_e = \epsilon_{e,0} \exp[\mathrm{tr}(\boldsymbol{\mathcal{R}}_e \boldsymbol{\mathcal{S}}_e)], \tag{44}$$

where $\boldsymbol{\mathcal{R}}_e$ is a symmetric error convergence rate tensor containing elemental directional convergence information. At each adaptation iteration, $\boldsymbol{\mathcal{R}}_e$ is determined separately for each element through a local sampling procedure in which the element is refined with different configurations, *i.e.*, imposing different step matrices, and the resulting changes to the output error are estimated.

For a triangle element, four refinement options are considered, as shown in Figure 1. For each refinement option i, the corresponding error $\epsilon_{e,i}$ is estimated to determine $\boldsymbol{\mathcal{R}}_e$ using Eqn. 44. The error estimates require the fine space adjoint solution for each proposed refinement, which can be expensive to solve even for a local patch of elements. Instead, we use an element-local projection method [45] to approximate the fine-space adjoint in semi-refined spaces associated with each refinement option.

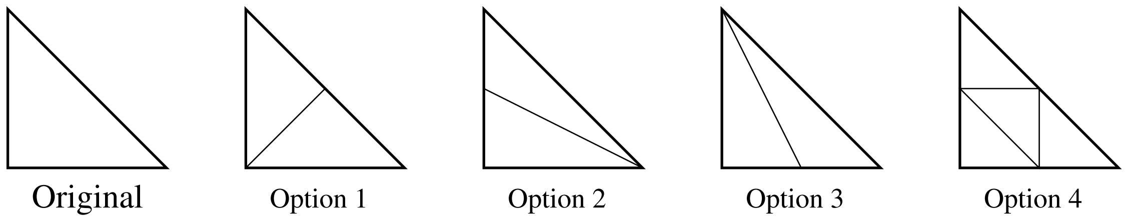


Figure 1: Four refinement options for a triangle. Each one is implicitly considered during the error sampling, though the elements are never actually refined.

5.2.3.2. Cost Model

We use degrees of freedom to measure the cost of refinements, which on each element just depends on the approximation order p. The cost of a local refinement is thus directly proportional to the number of new elements occupying the original area of element Ω_e, n_e, which is given by Eqn. 39. Assuming that the current mesh and the refined mesh are both metric-conforming, n_e can be related to the change in the metric,

$$C_e = C_{e,0} n_e = C_{e,0} \prod_{i=1}^{d} (h_i^c / h_i^f) = C_{e,0} \sqrt{\frac{\det(\boldsymbol{\mathcal{M}}_e)}{\det(\boldsymbol{\mathcal{M}}_{e,0})}} = C_{e,0} \exp\left[\frac{1}{2}\mathrm{tr}(\boldsymbol{\mathcal{S}}_e)\right], \tag{45}$$

where C_e is the expected cost over the original element with current cost $C_{e,0}$, after the refinement with step matrix $\boldsymbol{\mathcal{S}}_e$. The total cost is the sum of the elemental costs over the mesh, $C = \sum_{e=1}^{N_e} C_e$.

5.2.3.3. Mesh Optimization Algorithm

The goal of the optimization is to determine the optimal step tensor field, approximated by values at mesh vertices, $\boldsymbol{\mathcal{S}}_v(\vec{x})$, to minimize the total error indicator while keep the cost fixed,

$$\begin{aligned} \min_{\boldsymbol{\mathcal{S}}_v} \quad & \epsilon = \sum_{i=1}^{N_e} \epsilon_e(\boldsymbol{\mathcal{S}}_v) \\ \text{s.t.} \quad & C = \sum_{i=1}^{N_e} C_e(\boldsymbol{\mathcal{S}}_v) = \text{const.} \end{aligned} \tag{46}$$

The optimality condition requires

$$\frac{\partial \epsilon}{\partial \boldsymbol{\mathcal{S}}_v} + \lambda_s \frac{\partial C}{\partial \boldsymbol{\mathcal{S}}_v} = \mathbf{0}, \tag{47}$$

where λ_s is the Lagrange multiplier. Instead of solving the high-dimensional optimization problem, we follow an iterative process to equally distribute a locally defined Lagrange multiplier [38],

$$\lambda_{s,e} = \frac{\partial \epsilon_e / \partial \boldsymbol{\mathcal{S}}_v}{\partial C_e / \partial \boldsymbol{\mathcal{S}}_v}. \tag{48}$$

$\lambda_{s,e}$ can be interpreted as the local marginal error-to-cost ratio, which when equidistributed yields a solution of the original optimization in Eqn. 46.

MOESS can also be made error-based or cost-based similarly as described in Section 5.2.2. However, the optimal step tensor field is determined iteratively with a fixed cost in MOESS as it is formulated, making the error-based approach expensive and inefficient as the cost changes during the adaptation. Therefore, only the cost-based adaptation strategy is considered for MOESS in this work. The implementation of mesh adaptation in a multipoint optimization problem is given in Section 6.3.

6. Optimization Approach

6.1. Objective and Constraints

For demonstration, two-dimensional airfoil shape optimizations are considered in this work. In particular, the problem considered here is to search for an optimal design (including the airfoil shape and the angles of attack) to minimize the overall drag under a range of flight conditions, subject to fixed lift coefficients and a minimum airfoil volume. The optimization objective is the weighted sum of the drag coefficients at different design points, and the corresponding constraints are

$$\begin{aligned} & R_i^{\text{e}}(\mathbf{U}_i, \mathbf{x}) = c_{\ell,i} - \bar{c}_{\ell,i} = 0, \qquad i = 1, ..., N_m; \\ & R^{\text{ie}}(\mathbf{x}) = A(\mathbf{x}) - A_{\min} \geq 0. \end{aligned} \tag{49}$$

A and $A_{\min}$ represent the current and minimum volumes of the airfoil, and $c_{\ell,i}$ and $\bar{c}_{\ell,i}$ denote the current and the target lift coefficients at each design point.

The lift constraints are treated as the trimming constraints, and the angle of attack at each design point α_i is chosen as the trim variable, $\mathbf{x}_t = [\alpha_1, \alpha_2, ..., \alpha_{N_m}]$, to decouple the trim conditions at various points. During each flow solve, the trim constraints are enforced by a trimming process, which involves a Newton-Raphson iteration of the angles of attack and is presented with details in Appendix A. The inequality volume constraint, independent of the flow states, is assumed to be measured exactly and handled by the optimizer.

6.2. Airfoil Parameterization and Mesh Deformation

The airfoil shape is parameterized using the Hicks-Henne basis functions [46], taking a baseline airfoil and creating a new airfoil shape by adding a linear combination of "bump" functions to its upper and lower surfaces,

$$z(x) = z_{\text{base}}(x) + \sum_{i=1}^{N_s} a_i \phi_i(x). \tag{50}$$

x denotes in this case the position along the airfoil chord, and z is the vertical coordinates of the upper or lower airfoil surfaces, considered separately. ϕ_i are the basis functions taken from an optimized basis set [47], whose coefficients a_i are the active design variables for the optimization problem, $\mathbf{x}_s = [a_1, a_2, ..., a_{N_s}] \in \mathbb{R}^{N_s}$. Table 1 summarizes the multipoint aerodynamic optimization problem considered in this work.

Table 1: Multipoint aerodynamic shape optimization problem

	Function/Variable	Description	Quantity
Minimize	$\sum_{i=1}^{N_m} \omega_i c_{d,i}$	Weighted drag coefficients sum	1
With respect to	$\mathbf{x}_s$	Hicks-Henne basis function coefficients	N_s
	$\mathbf{x}_t$	Angles of attack	N_m
Subject to	$c_{\ell,i} - \bar{c}_{\ell,i} = 0$	Lift constraints	N_m
	$A - A_{\min} \geq 0$	Volume constraint	1

At each optimization iteration, the objective function needs to be re-evaluated, which requires a flow solve on the updated geometry, and hence a new mesh must be obtained each time. Rather than regenerating meshes every time, the airfoil boundary deformation is propagated to the interior mesh with an explicit inverse-distance interpolation algorithm [48].

6.3. Optimization Algorithm

Sequential Least Squares Programming (SLSQP) [49] with quasi-Newton type Hessian approximation is used in this work. The gradient of the objective function is calculated by the adjoint method, per Eqn. 15, and the objective and constraints are evaluated with the numerical solution of Eqn. 16 based on the discretization given in Section 3. Although the optimization problem is formulated in an augmented Lagrangian form in Section 2, any gradient-based constrained optimization algorithm can be used since the Lagrange multipliers associated with the trimming constraints are obtained during the trimming process for mesh adaptation. If the trimming constraints are handled by the optimizer, augmented Lagrangian methods have to be used in order to provide the corresponding Lagrange multipliers for adaptation purposes [15].

Instead of optimizing on a mesh with fixed resolution, which would always require the highest fidelity for accurate calculations, the mesh is progressively refined as the optimization proceeds, resulting in a variable-fidelity optimization. Rather than performing optimization and mesh adaptation sequentially, one after another, an interactive framework is introduced. Two possible ways to incorporate the mesh adaptation and design optimization are considered here: optimization-driven adaptation and adaptation-driven optimization. In the former approach, the optimization tolerance at each fidelity is prescribed by the user. Starting with a loose optimization tolerance, the total maximum allowable error, set to be equal to the optimization tolerance, is first divided into each design point by Eqn. 31. The objective function at each design point is then evaluated on the same coarse initial mesh, and the error estimation and mesh adaptation are performed individually to control the numerical error to be below the error tolerance at the current fidelity. The allowable numerical error tolerance decreases as the optimization fidelity increases. For the latter approach, several mesh levels (degrees of freedom) are defined before the optimization. Starting with a low total cost, the degrees of freedom are redistributed according to Eqn. 30. Then the mesh is modified in Hessian-based adaptation or optimized in MOESS at each design point to improve the accuracy. Once the objective improvement is smaller than the objective error estimate, the optimization terminates at the current cost level and the fidelity increases through mesh adaptation with a higher cost. We refer to these methods as error-based optimization or cost-based optimization, depending on the information specified. The error-based optimization needs an error-based mesh adaptation method: here we use error-based Hessian adaptation; while the cost-based one requires cost-based adaptation mechanics, which can be either cost-based Hessian adaptation or MOESS.

Compared to a more traditional optimization methodology with an *a priori* mesh, unnecessarily fine meshes at the early stages of shape optimization are avoided in the proposed variable-fidelity framework. The problem setup time is significantly reduced with easier mesh generation. Moreover, the elements that introduce most of the error may differ significantly for different shape configurations during the optimization.

Both approaches reduce the chance of over-refining elements that are not relatively important for the final design, which is necessary if the adaptation mechanics do not allow for coarsening. On the other hand, the coupling between error tolerance and optimization tolerance at each fidelity actively controls the optimization at each step to avoid unnecessary convergence at low fidelity. Finally, the new framework can effectively prevent over-refining on an unintended shape, or over-optimizing on a coarse mesh.

The proposed optimization frameworks with error estimation and mesh adaptation are summarized in Algorithm 1 and Algorithm 2, using error-based and cost-based approaches, respectively. Optimization tolerance levels and cost levels are specified by the user, driving the mesh adaptation to actively control the numerical errors. In this paper, we assume that the error estimation is sufficiently accurate to represent the "true" numerical error, which may be inappropriate when the adjoint is not well-resolved or when the problem is highly nonlinear. In practice, a safety factor η can be used to ensure the numerical error to always be below the optimization tolerance; $\eta = 1$ is adopted in this paper.

Algorithm 1: Optimization with error estimation and mesh adaptation (error-based)

```
input : initial design $\mathbf{x}_0$, initial coarse mesh $\mathcal{T}_h$, optimization tolerance levels $\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_n$, safety
        factor $\eta \leq 1$
output: adapted meshes at each design point $\mathcal{T}_{h,i}$
        optimized design $\mathbf{x}^*$ with controlled objective error $\delta J_{m,h}^{\text{adapt}} \leq \mathcal{O}_n$
1  for $l = 1, 2, ..., n$ do
2      set the total error tolerance as $\tau_l = \eta \mathcal{O}_l$
3      while not converged do                                          ▷ optimization algorithm
4          distribute the total error tolerance $\tau_l$ at each design point as $\tau_{l,i}$, using Eqn. 31
5          for $i = 1, ..., N_m$ do
6              while $\delta J_{m,i}^{adapt} > \tau_{l,i}$ do
7                  adapt the mesh $\mathcal{T}_{h,i}$ with refinements                 ▷ Hessian adaptation
8                  update $\mathbf{x}_{t,l}$ to meet trim constraints                 ▷ trimming process
9                  compute the objective component $J_{m,i}^{\text{adapt}}$ and its error estimate $\delta J_{m,i}^{\text{adapt}}$
10             end
11         end
12         update the composite objective $J_m^{\text{adapt}} = \sum_{i=1}^{N_m} J_{m,i}^{\text{adapt}}$
13         calculate the composite objective gradient $dJ_m^{\text{adapt}}/d\mathbf{x}_{s,l}$, per Eqn. 15
14         update the active design $\mathbf{x}_{s,l}$ with meshes $\mathcal{T}_{h,i}$ fixed                ▷ line search
15     end
16     finish optimization at level $l$, $\mathbf{x}_{l+1} = \mathbf{x}_l$
17 end
```

Algorithm 2: Optimization with error estimation and mesh adaptation (cost-based)

```
input  : initial design $\mathbf{x}_0$, initial coarse mesh $\mathcal{T}_h$, cost levels $C_1, C_2, ..., C_n$, safety factor $\eta \geq 1$
output: optimized mesh at each design point $\mathcal{T}_{h,i}$ with total cost $C_n$
        optimized design $\mathbf{x}^*$ with optimized accuracy at given total cost $C_n$
1  for $l = 1, 2, ..., n$ do
2      distribute the total cost $C_l$ among various design points as $C_{l,i}$, using Eqn. 30
3      while not converged do                                                  ▷ optimization algorithm
4          for $i = 1, ..., N_m$ do
5              for $j = 1, ..., N_{adapt}$ do
6                  adapt the mesh $\mathcal{T}_{h,i}$ with DOF redistribution          ▷ Hessian adaptation/MOESS
7                  update $\mathbf{x}_{t,l}$ to meet the trim constraints                    ▷ trimming process
8                  compute the objective component $J_{m,i}^{\text{adapt}}$ and its error estimate $\delta J_{m,i}^{\text{adapt}}$
9              end
10         end
11         update the composite objective $J_m^{\text{adapt}} = \sum_{i=1}^{N_m} J_{m,i}^{\text{adapt}}$
12         calculate the composite objective gradient $dJ_m^{\text{adapt}}/d\mathbf{x}_{s,l}$, per Eqn. 15
13         set the optimization tolerance $\mathcal{O}_l = \eta \sum_{i=1}^{N_m} \delta J_{m,i}^{\text{adapt}}$
14         update the active design $\mathbf{x}_{s,l}$ with meshes $\mathcal{T}_{h,i}$ fixed              ▷ line search
15     end
16     finish optimization at level $l$, $\mathbf{x}_{l+1} = \mathbf{x}_l$
17 end
```

7. Results

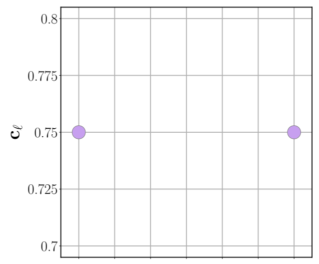
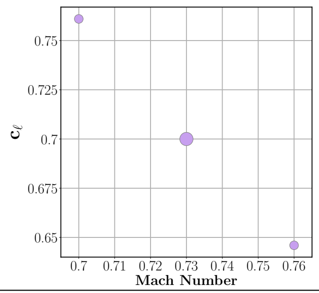
As a simple demonstration of the proposed optimization frameworks, we consider two-dimensional airfoil optimization problems in transonic flow regimes, over a range of flight conditions. The goal of the optimization is to search for an optimal airfoil shape and angles of attack to minimize the drag coefficients, subject to fixed lift trim conditions and a minimum volume constraint. We only consider the discretization errors in drag and lift calculations, and the airfoil volume measurements are assumed to be exact. Furthermore, the trim constraint tolerances are always set to be sufficiently small to make sure the sensitivity calculation in Eqn. 15 is accurate. The airfoil shape is parameterized with 16 Hicks-Henne basis functions, and the design parameter vector includes both the shape parameters and the angle of attack at each design point. Unstructured triangular meshes and DG $p = 2$ approximation are used for the discretization. The airfoil boundary is represented by cubic curved mesh elements. We first test our proposed methods on a two-point, inviscid airfoil optimization problem, following which a more practical turbulent case including three flight conditions is considered. A detailed description of the two cases are given in Table 2.

7.1. Multipoint Inviscid Transonic Airfoil Optimization

In this test case, the two-point optimization starts with a Royal Aircraft Establishment (RAE) 2822 airfoil and seeks an optimal shape and angles of attack to minimize the weighted drag coefficient, subject to fixed lift constraints and nondecreasing airfoil volume. The two operating conditions including the corresponding lift trim constraints are listed in Table 2.

Under the high lift trim condition, flow around the original RAE 2822 airfoil features a strong shock on the upper surface, the location and strength of which vary depending on the operating conditions, *i.e.*, Mach number in this case. Without any priori knowledge about the flow fields around the airfoil at each design point, a fairly fine mesh with specific refinement around the airfoil is generally used in optimization. Effort can be put into generating meshes suitable for capturing the shocks effectively, either based on experience or output-based error estimates. However, this only helps the analysis on the original shape. If the shock moves or its strength reduces as the optimization proceeds, the specific resolution for the initial design is

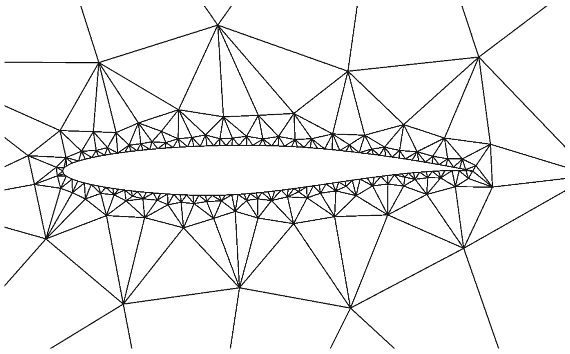
Table 2: Operating conditions for multipoint optimization problems

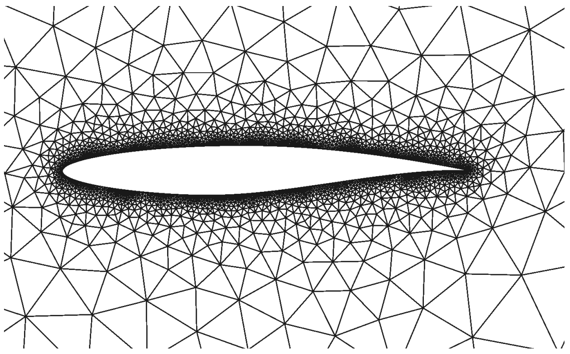
Case	Point	Weights w_i	Mach	c_ℓ	Reynolds number	$M - c_\ell$ plot
7.1	1	0.50	0.70	0.750	–	
	2	0.50	0.76	0.750	–	
7.2	1	0.25	0.70	0.761	4.79×10^6	
	2	0.50	0.73	0.700	5.00×10^6	
	3	0.25	0.76	0.646	5.21×10^6	

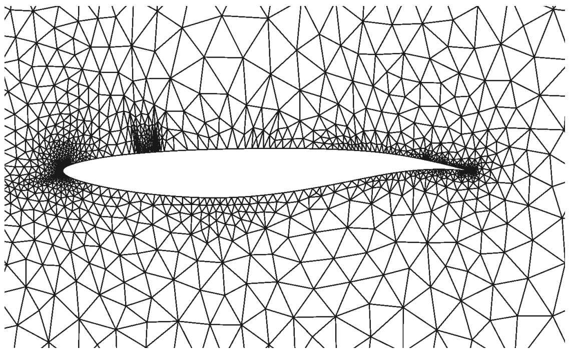
wasted. Particularly, we expect in this case for the shape to be modified such that the shock strength is significantly weakened. Any substantial refinement on the initial shock location will thus not effectively increase the accuracy but instead add considerable computational cost to the optimization.

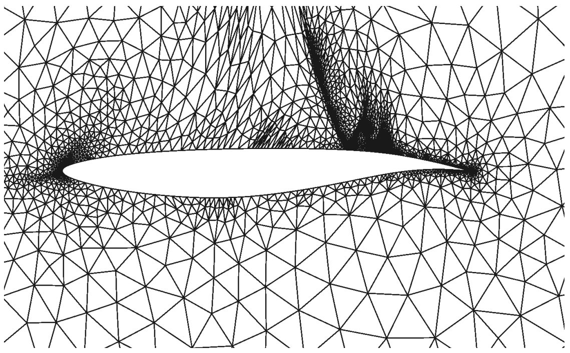
We test both the error-based and cost-based optimization frameworks as described in Algorithm 1 and Algorithm 2, with various adaptation methods. For error-based optimization, we use error-based Hessian adaptation; while for the cost-based optimization, both MOESS and cost-based Hessian adaptation are used. All these three optimizations start with the same initial mesh consisting of 393 triangular elements, as shown in Figure 2(a). In the error-based optimization, a set of optimization tolerance levels is specified with an ultimate tolerance of 0.02 drag counts, *i.e.*, 2×10^{-6}. On the other hand, the cost-based optimization starts with a fairly low cost level, and degrees of freedom are added once the optimization converges at current cost level, until the final optimization tolerance is below 0.02 drag counts. To compare with traditional methods, we also run fixed-fidelity optimization on two fixed meshes. The coarse one has comparable DOF as the finest mesh used in the variable-fidelity optimization while the fine one has double the cost. The optimization tolerances are also set to be 0.02 drag counts. The meshes used in these different optimizations are summarized in Figure 2(b)—2(h). Only the coarse mesh used in the fixed-fidelity optimization is shown for conciseness, as the finer one has more elements but similar uniform refinement around the airfoil boundary.

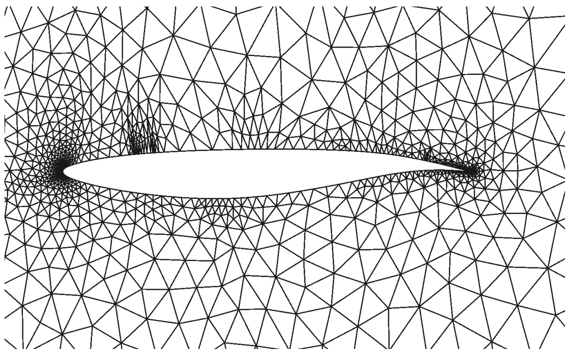
The objective convergence and mesh evolution are shown in Figure 3. In Figure 3(a), we plot the objectives verses the aggregated total degrees of freedom, which are only accumulated at each optimization major iteration, *i.e.*, not including the line search. We see in the plot that the estimated discretization error of the objective is always above the optimization tolerance in the fixed-fidelity (fixed-mesh) optimization. On the coarse fixed mesh, the discretization error is large and sometimes even comparable to the objective values. Although the objective error decreases as the finer fixed mesh is used, it is still fairly large compared to the optimization tolerance. In these scenarios, the optimizer may work on the numerical error instead of the physics to minimize the drag, leading to inaccurate designs. On the other hand, discretization error is always controlled to be below the optimization tolerance, or the optimization tolerance is adjusted to be equal to the discretization error in the proposed methods. Furthermore, the variable-fidelity optimizations with different adaptation methods all converge faster at the highest fidelity by virtue of a better starting point obtained from the lower fidelity. Significant computational resources can be saved with fast, low-fidelity

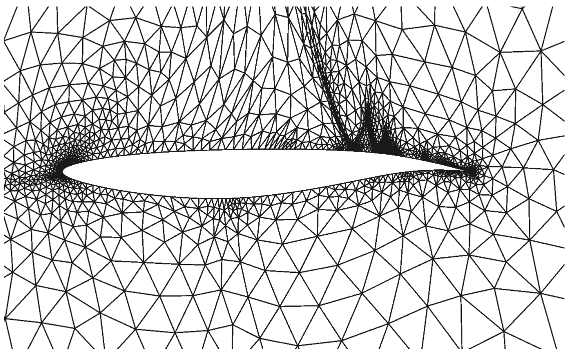

(a) Initial mesh for variable-fidelity optimization

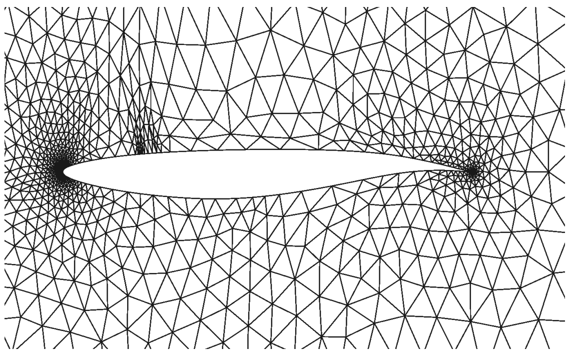

(b) The coarse mesh for fixed-fidelity optimization


(c) Final mesh at $M = 0.70$ (error-based Hessian adaptation)


(d) Final mesh at $M = 0.76$ (error-based Hessian adaptation)


(e) Final mesh at $M = 0.70$ (cost-based Hessian adaptation)


(f) Final mesh at $M = 0.76$ (cost-based Hessian adaptation)


(g) Final mesh at $M = 0.70$ (MOESS)

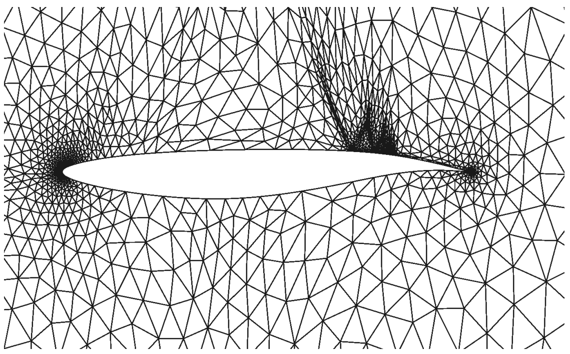
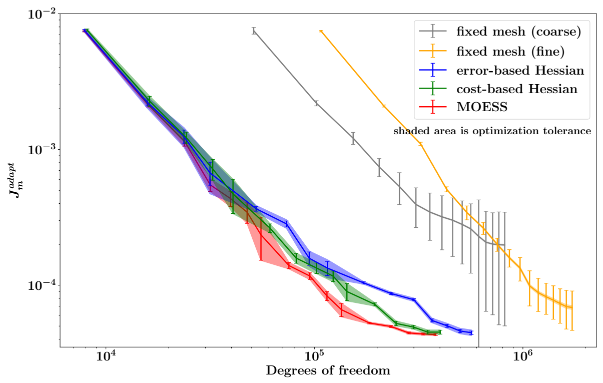
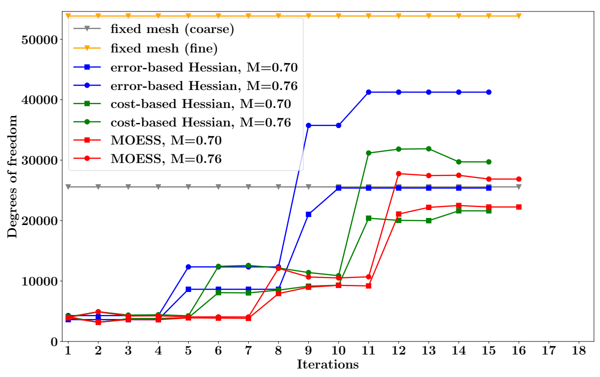

(h) Final mesh at $M = 0.76$ (MOESS)

Figure 2: Meshes for variable-fidelity and fixed-fidelity optimization (inviscid, transonic)

optimizations.



(a) Objective convergence history



(b) Mesh size evolution

Figure 3: Convergence history and mesh size evolution for different methods (inviscid, transonic)

We can also observe from the mesh evolution plot in Figure 3(b) that the mesh sizes, if adapted, required to achieve similar accuracy on different operating conditions are different. For all the mesh adaptation methods considered, the final mesh size for Mach number of 0.70 is smaller than the one required for Mach number 0.76. Mesh adaptation prevents unnecessarily fine meshes from being used for relatively simple operating conditions. The distinction among these methods comes from the difference between error-based and cost-based approaches, and the difference between Hessian adaptation and MOESS. The error-based approach refines the mesh every time the error is above the optimization tolerance, keeping it fixed otherwise, and hence it tends to over-refine areas that are important for some intermediate designs but not necessary for the final design. On the other hand, the cost-based approach always adapts the mesh while keeps the cost fixed at the same fidelity, so that the redistribution of the degrees of freedom avoids over-refinement and improves the accuracy. If we look at the error-based Hessian adaptation and the cost-based Hessian adaptation (blue and green lines) in Figure 3, they have very similar convergence and costs at low fidelities. However at the highest fidelity, the error-based approach has more refinements, which are added for some intermediate designs. Those extra refinements do not affect the final accuracy much as we can see both methods have similar final accuracy. This suggests using the cost-based optimization framework with cost-based adaptation methods. Although it requires several adaptive iterations with fixed DOF at each major optimization iteration, it prevents extremely fine meshes from being used at the highest fidelity, which is always the main overhead in the optimization. In terms of adaptation methods, both using cost-based approach, MOESS benefits from more appropriate anisotropy detection, resulting in lower cost and better accuracy. As we can see in Figure 2(f) and Figure 2(h), MOESS meshes tend to have more anisotropic elements in the post shock locations, though they have similar refinement at the shocks. If we look at the upstream region of these two meshes, as shown in Figure 4, the difference is more evident: Hessian adaptation only has isotropic refinement along the stagnation streamline since it is important for the output prediction but the Mach number field is isotropic across it; However, MOESS is able to detect the anisotropy through sampling and puts anisotropic resolution along the stagnation streamline. This refinement also indicates the trim output effects on the adaptation, as this anisotropy is more important for the lift calculation, *i.e.*, the trim output, while not very relevant to the prediction of the objective, *i.e.*, the drag. Therefore, we can see in Figure 3 that at the low fidelities, with similar cost, MOESS achieves lower objective error, hence better convergence and better design at the low fidelities. With a better starting design, the optimization at the highest fidelity has smoother convergence, as the sharp objective change at the highest fidelity that occurs in both error-based and cost-based Hessian adaptation is not observed in MOESS. Since the flow solve in the optimization always restarts from the solution on the previous design, MOESS converges faster and consumes less CPU time compared to cost-based Hessian adaptation due to smaller design changes, even though the aggregated total degrees of freedom are close. The computational cost saving is reflected in Table 3. Computational time results are obtained using parallel runs with 8 processors on the same machine (Intel Core i7-3770 3.40 GHz CPU with 16GB total RAM). The proposed variable-fidelity optimization frameworks with adapted meshes achieve substantial time savings compared to fixed-fidelity optimization with fixed meshes. As mentioned, the cost-based algorithm out-performs the error-based one with considerable time savings on the highest fidelity; cost-based optimization with MOESS achieves the most time savings, around 3 times and 8 times speedup compared to the optimizations on the coarse and fine fixed meshes respectively.

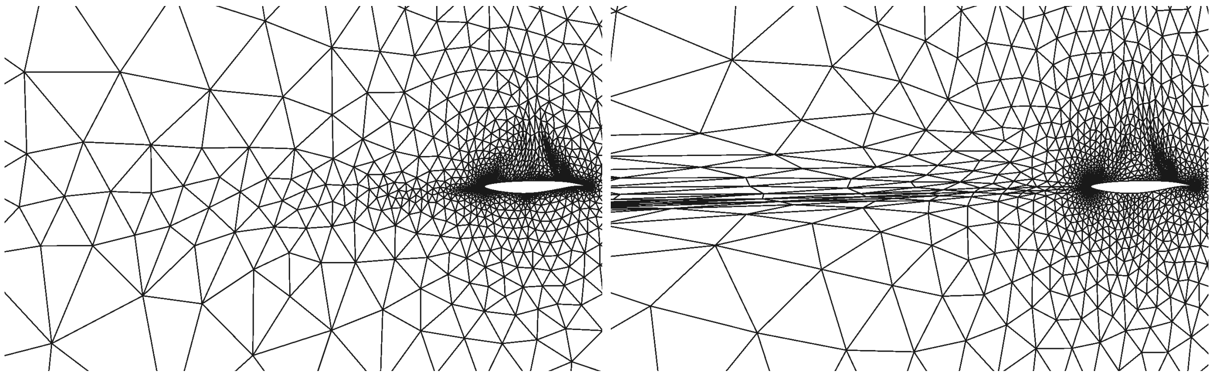


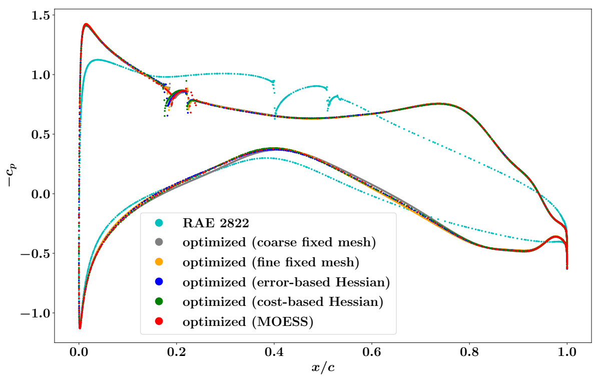
Figure 4: Meshes around the stagnation streamline, the left mesh is from cost-based Hessian adaptation, the right one is the MOESS adapted mesh.

Table 3: Computational cost comparison (inviscid, transonic). In cost-based optimization, the optimization tolerance is dynamically adjusted to be equal to the objective error estimate; the approximate values of the optimization tolerance in this table are from the last iteration on each fidelity.

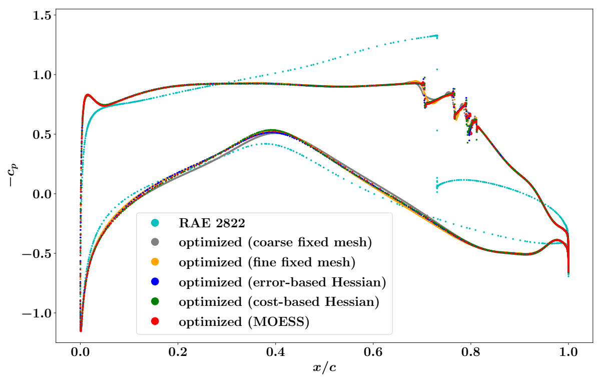
	Optimization level	Optimization tol (Drag count)	CPU time (s)
Fixed-fidelity (coarse fixed mesh)	L3	0.020	6.861×10^4
Fixed-fidelity (fine fixed mesh)	L3	0.020	1.924×10^5
Variable-fidelity (error-based Hessian)	L1	2.000	7.880×10^2
	L2	0.200	7.153×10^3
	L3	0.020	3.668×10^4
Variable-fidelity (cost-based Hessian)	L1	≈ 1.329	2.149×10^3
	L2	$\delta J_m^{\text{adapt}} \approx 0.130$	8.241×10^3
	L3	≈ 0.015	2.113×10^4
Variable-fidelity (MOESS)	L1	≈ 0.822	2.334×10^3
	L2	$\delta J_m^{\text{adapt}} \approx 0.074$	5.550×10^3
	L3	≈ 0.007	1.630×10^4

The initial and optimized airfoils are compared in Figure 5, while the final objective values are collected in Table 4, in which the corresponding "true" objective values are also obtained via adapted meshes on the final designs, with discretization error controlled to be one order of magnitude smaller than the final optimization tolerance. All of the optimizations flatten the upper surface near the forward section, while curving and increasing the thickness at the lower surface. The curvature reduction on the top surface smooths the flow acceleration region to weaken the shock, while the thickened lower surface and curved aft section are required to maintain the lift and area constraints. Therefore, the strong shocks are significantly reduced at both operating conditions, as shown in the pressure distributions in Figure 5(a)–5(b). In the optimization runs with adapted meshes, areas around the airfoil leading and trailing edges are significantly refined, and many elements are dedicated to the shocks and the stagnation streamline. However, in the optimization with fixed meshes, elements are not efficiently distributed, and areas that are important for output predictions are not well-resolved, which as a result causes high objective error as seen in Figure 3(a).

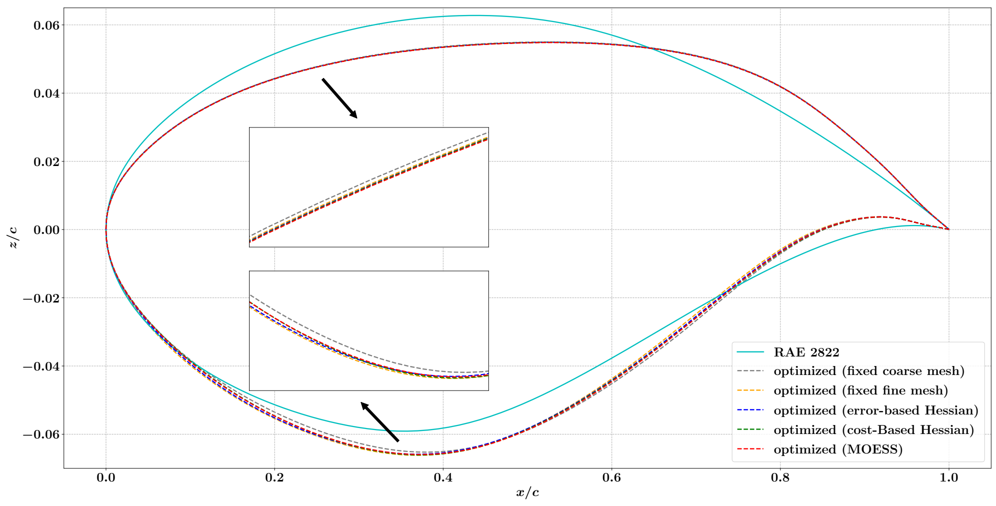
When the numerical error is too high, for example on the coarse fixed mesh, the optimization converges to a noticeably different design compared to the designs obtained from other optimizations, as shown in Figure 5(c). Thus the "true" objective value for the optimized design on the coarse fixed mesh is much higher compared to designs obtained on the other meshes. In the optimization with the fixed fine mesh, the discretization error is still high, but the optimization is able to converge to a similar design compared to designs produced by optimizations with discretization error control, as shown in Figure 5(c). Although the "true" objective value is also close to (still slightly higher) the objective values of our proposed methods with mesh adaptation (the difference among these methods is below or comparable to the optimization tolerance, which means that the optimization on these meshes converges correctly), the final objective value reported on the fixed mesh is far from accurate for practical design and the cost is extremely high compared to our proposed methods, which is observed in both Table 3 and Table 4. The proposed methods with mesh adaptation are able to obtain a reasonable design, and the associated error estimation is also accurate enough to provide confidence in the final design and computed output quantities.



(a) Pressure distribution at $M = 0.70$



(b) Pressure distribution at $M = 0.76$



(c) Initial and optimized shapes

Figure 5: Pressure distribution for the initial and optimized designs (inviscid, transonic)

7.2. Multipoint Turbulent Transonic Airfoil Optimization

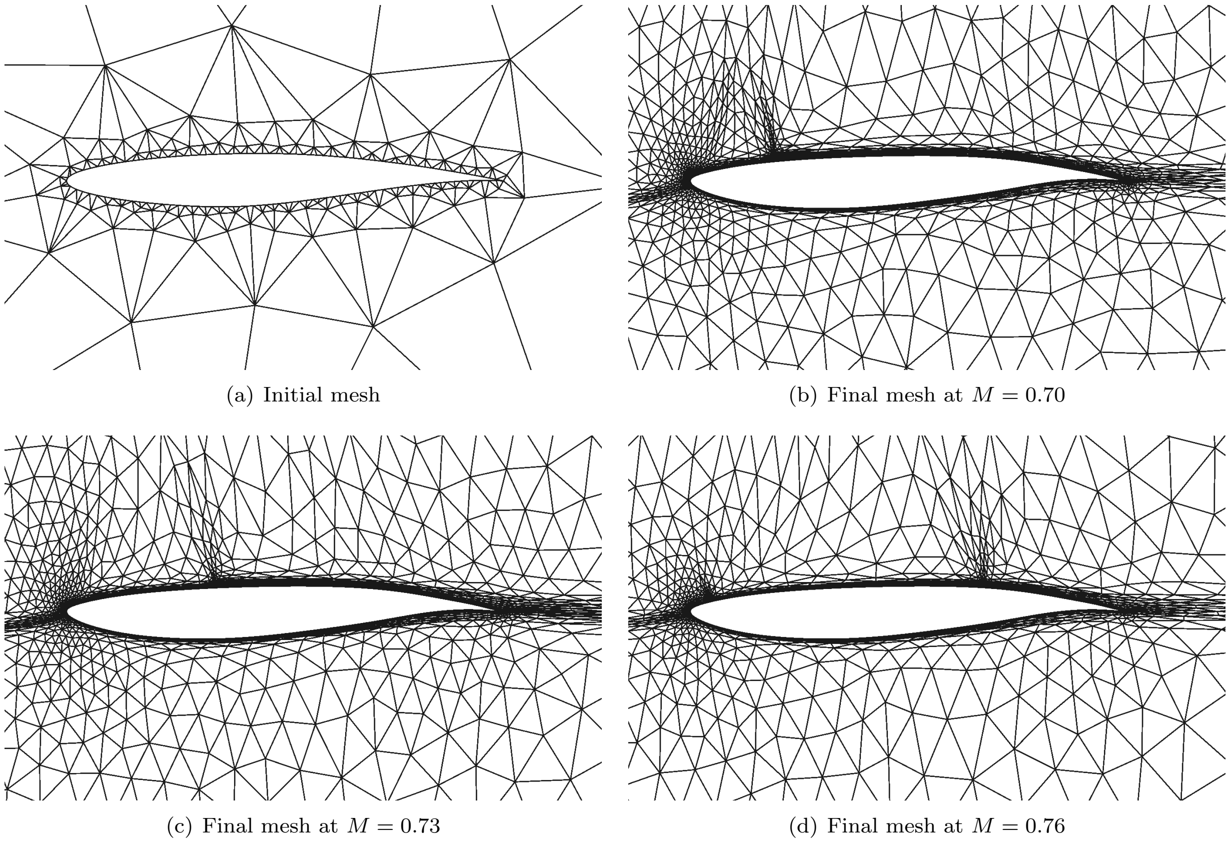
Another problem considered in this paper is a more sophisticated fully-turbulent case. We set up a three-point optimization problem, analogous to minimizing the integrated drag coefficients over a range of

Table 4: Optimization results on different meshes (inviscid, transonic)

	Final mesh DOF	J_m^{adapt}	J_m^{adapt} ("true")
Fixed mesh (coarse)	51144	$1.978 \times 10^{-4} \pm 1.477 \times 10^{-4}$	4.932×10^{-5}
Fixed mesh (fine)	107688	$6.854 \times 10^{-5} \pm 2.208 \times 10^{-5}$	4.552×10^{-5}
Error-based Hessian	66630	$4.466 \times 10^{-5} \pm 1.311 \times 10^{-6}$	4.338×10^{-5}
Cost-based Hessian	51324	$4.513 \times 10^{-5} \pm 1.504 \times 10^{-6}$	4.383×10^{-5}
MOESS	49116	$4.333 \times 10^{-5} \pm 7.202 \times 10^{-7}$	4.278×10^{-5}

Mach numbers at a fixed aircraft weight and altitude. The optimization again starts with the RAE 2822 airfoil, and seeks an optimal shape and angles of attack to minimize the composite drag coefficients with fixed lift constraints and non-decreasing airfoil volume; the details of the case setup are given in Table 2.

For turbulent flow simulations at high Reynolds number, one of the key flow features is the thin boundary layer. Due to the linear velocity profile in the viscous sub-layer, Hessian-based mesh adaptation is usually inefficient since the Mach number Hessian is close to zero within this region. Therefore, only MOESS with cost-based variable-fidelity optimization is used in this case. The starting and final mesh at each flight condition are compared in Figure 6. At all the flight conditions considered, MOESS is able to effectively detect strong directional flow features and put highly-anisotropic elements around the airfoil boundary, at shock locations on the top surface, along the stagnation streamline and also near the wake region.


(a) Initial mesh

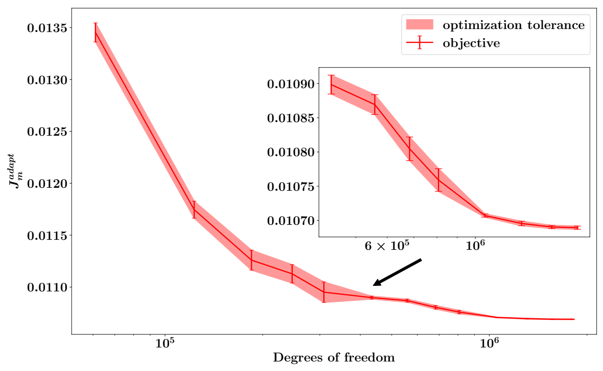
(b) Final mesh at $M = 0.70$

(c) Final mesh at $M = 0.73$

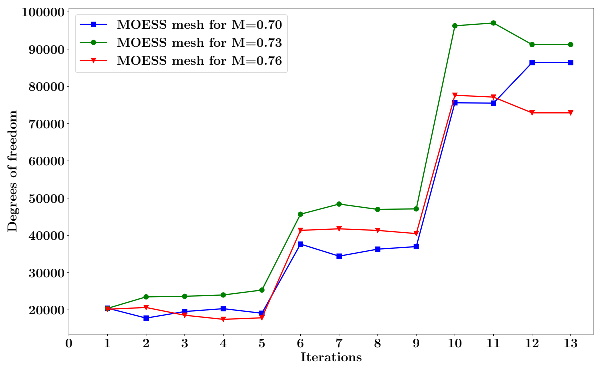
(d) Final mesh at $M = 0.76$

Figure 6: Initial mesh and final meshes during the optimization (turbulent, transonic)

The objective and mesh size are collected at each optimization step as shown in Figure 7. In the convergence plot, we can see that the composite objective errors are close even for different designs at the same optimization fidelity, as the total degrees of freedom are optimally distributed among different flight conditions, and the meshes are optimized individually at each of them. The degrees of freedom assigned to the design point at Mach number of 0.73 are consistently higher than at the other two points, mainly due to the higher weight used during the optimization. On the other hand, the cost distribution between Mach number of 0.70 and 0.76 changes as the design varies. As the optimization progresses, the meshes get refined and optimized, and more detailed design improvement is then made with smaller objective error and tighter optimization tolerance, *i.e.*, both the design and meshes converge to the optimum.



(a) Objective convergence history



(b) Mesh size evolution

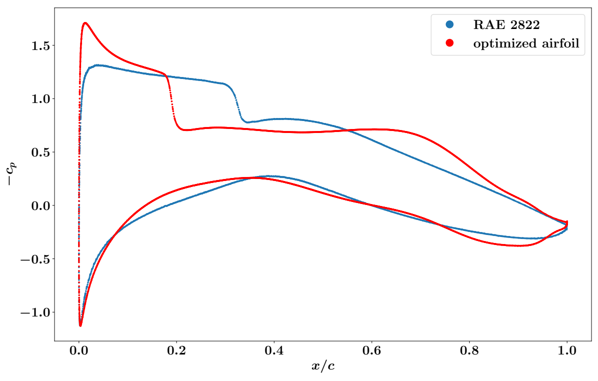
Figure 7: Convergence history and mesh size evolution for different methods (turbulent, transonic)

Figures 8(a)–8(c) shows the initial and final pressure distributions at each design point. The corresponding final airfoil shape is shown in Figure 8(e). As in the inviscid case, the upper surface of the airfoil is flattened, while more curvature is added to the lower surface aft section. As we can see in the pressure distribution plots, both the location and the strength of the original strong shock at each design point are modified. The optimizer weakens the shock at both Mach numbers of 0.73 and 0.76, while slightly strengthens the shock at the Mach number of 0.70, resulting in a significant reduction in the composite drag coefficient as shown in Figure 7(a). The drag divergence curves for the original RAE 2822 airfoil and the optimized design around the nominal flight conditions are depicted in Figure 8(d). Despite some sacrifice in the performance at low cruise speeds, the new design achieves significantly lower drag values for Mach numbers above 0.73 and is able to maintain good performance over a much wider range of Mach numbers compared to the original design.

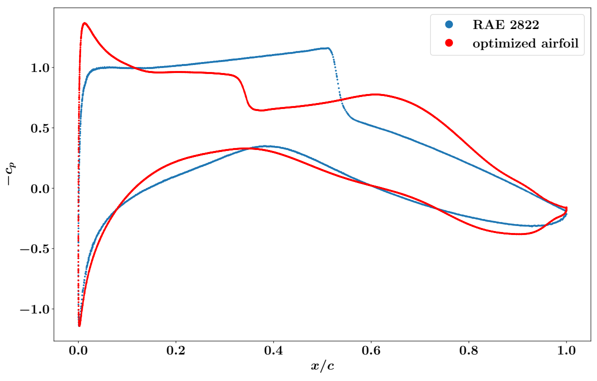
8. Conclusions

In practical aerodynamic design processes, the optimization problem has to be posed such that a range of operating conditions, including off-design points, are considered in the objective as well as the constraints. To ensure convergence to the "true" optimal design, the numerical error at each design point has to be carefully controlled. As the flow conditions involved can vary dramatically, *a priori* meshes appropriate for all the design points can be hard to generate and are generally not sufficient for the requirements of high-fidelity optimization.

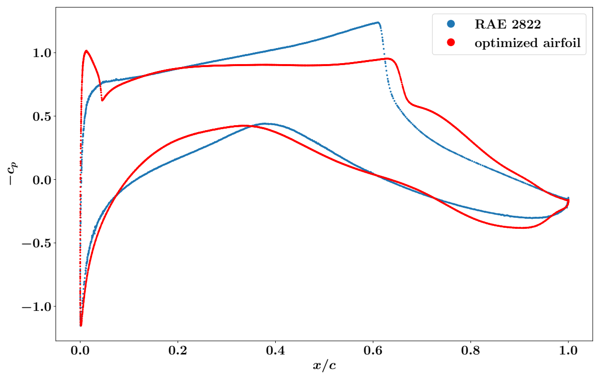
In this work, we presented a variable-fidelity framework that integrates output-based error estimation and mesh adaptation with a gradient-based algorithm for multipoint aerodynamic shape optimization problems. The proposed framework can considerably facilitate the optimization setup and accelerate the design process. The designer only needs to input an initial mesh, which can be fairly coarse and easy to generate. The mesh adaptation (fidelity increase) is then tightly coupled with the optimization algorithm either with an error-based or a cost-based strategy. The variable-fidelity optimization framework driven by mesh adaptation is capable of preventing over-optimizing and over-refining, as shown in the test cases. Design optimization with mesh optimization via error sampling and synthesis (MOESS) is shown to be more efficient and effective



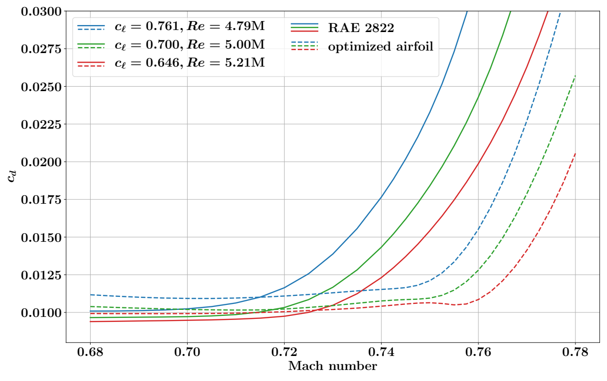
(a) Pressure distribution at $M = 0.70$



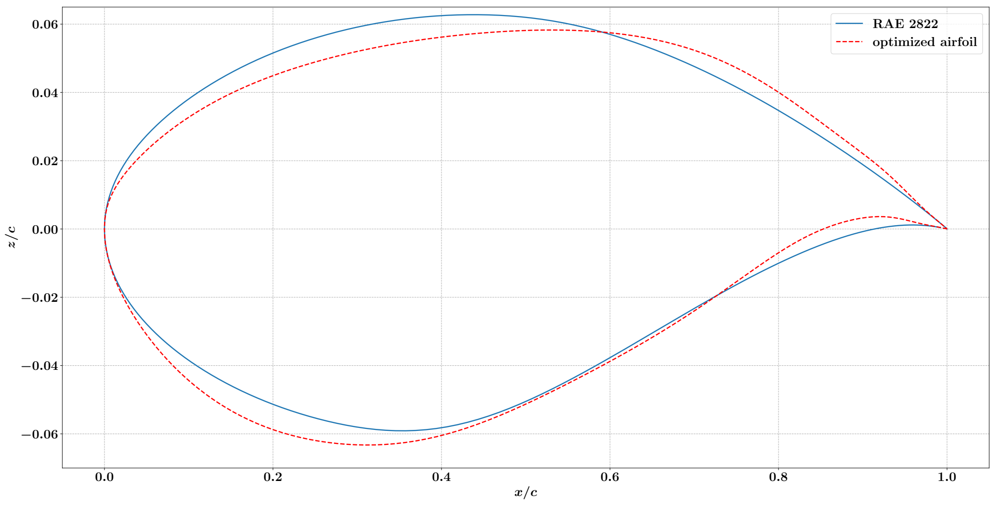
(b) Pressure distribution at $M = 0.73$



(c) Pressure distribution at $M = 0.76$



(d) Drag divergence curves around the nominal flight conditions



(e) Initial and optimized shapes

Figure 8: Pressure distribution for the initial and optimized designs (turbulent, transonic)

by optimized computational cost distribution among various flight conditions and optimized meshes at each point. This benefit can become more significant when higher fidelity is required, or when more highly anisotropic physics govern the system.

With more judicious considerations of the objective functions and constraints, and additional parameters, the new method can provide realistic configurations in practical design scenarios. Also, the computational cost allocation adopted in this work still partially relies on *a priori* error-cost relations, which can be inefficient when inappropriate convergence rates are predicted/used. More appropriate error and cost models, such as *a posteriori* ones constructed during the optimization, can be developed to guide the cost distribution among different design points. Furthermore, only mesh adaptation (h-adaptation) is considered here to control the discretization error. More efficient adaptation mechanics, such as approximation order increment (p-adaptation), and combinations (hp-adaptation), can also be applied to the proposed methods in the future.

Acknowledgments

The authors acknowledge the support of the Boeing Company, with technical monitor Dr. Mori Mani, and the Department of Energy under grant DE-FG02-13ER26146/DE-SC0010341.

Appendix A. Trimming Algorithm

In the trimming process, given an active design $\mathbf{x}_s$, we want to satisfy the flow equation and the trim equation simultaneously, at each design point,

$$\begin{aligned}\mathbf{R} &= \begin{bmatrix} \mathbf{R}_1(\mathbf{U}_1, \mathbf{x}_t) \\ \vdots \\ \mathbf{R}_{N_m}(\mathbf{U}_{N_m}, \mathbf{x}_t) \end{bmatrix} = \mathbf{0}, \\ \mathbf{R}^{\text{trim}} &= \begin{bmatrix} \mathbf{J}_1^{\text{trim}}(\mathbf{U}_i, \mathbf{x}_t) - \bar{\mathbf{J}}_1^{\text{trim}} \\ \vdots \\ \mathbf{J}_{N_m}^{\text{trim}}(\mathbf{U}_{N_m}, \mathbf{x}_t) - \bar{\mathbf{J}}_{N_m}^{\text{trim}} \end{bmatrix} = \mathbf{0}. \end{aligned} \tag{A.1}$$

The Newton update of the system can be written as,

$$\left[\begin{array}{cccc} \frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i} & & & \frac{\partial \mathbf{R}_1}{\partial \mathbf{x}_t} \\ & \ddots & & \vdots \\ & & \frac{\partial \mathbf{R}_{N_m}}{\partial \mathbf{U}_{N_m}} & \frac{\partial \mathbf{R}_{N_m}}{\partial \mathbf{x}_t} \\ \hline \frac{\partial \mathbf{R}_1^{\text{trim}}}{\partial \mathbf{U}_1} & & & \frac{\partial \mathbf{R}_1^{\text{trim}}}{\partial \mathbf{x}_t} \\ & \ddots & & \vdots \\ & & \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{U}_i} & \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_t} \end{array}\right] \left[\begin{array}{c} \Delta \mathbf{U}_1 \\ \vdots \\ \Delta \mathbf{U}_{N_m} \\ \hline \Delta \mathbf{x}_t \end{array}\right] + \left[\begin{array}{c} \mathbf{R}_1 \\ \vdots \\ \mathbf{R}_{N_m} \\ \hline \mathbf{R}^{\text{trim}} \end{array}\right] = \mathbf{0}. \tag{A.2}$$

We can first solve the upper block of the system to obtain $\Delta \mathbf{U}_i$, then substitute it back to the lower block to solve for $\Delta \mathbf{x}_t$, *i.e.*, solve the system via a Schur complement. The resulting equation for $\Delta \mathbf{x}_t$ is

$$\begin{aligned} &\left[\frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{x}_t} - \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{U}_i}\left(\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right)^{-1}\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t}\right]\Delta \mathbf{x}_t + \left[\mathbf{R}_i^{\text{trim}} - \frac{\partial \mathbf{R}_i^{\text{trim}}}{\partial \mathbf{U}_i}\left(\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right)^{-1}\mathbf{R}_i\right] = \mathbf{0}, \\ \Rightarrow &\left[\frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{x}_t} - \frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{U}_i}\left(\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right)^{-1}\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t}\right]\Delta \mathbf{x}_t + \left[\mathbf{J}_i^{\text{trim}} - \bar{\mathbf{J}}_i^{\text{trim}} - \frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{U}_i}\left(\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right)^{-1}\mathbf{R}_i\right] = \mathbf{0}, \\ &\Rightarrow \left[\frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{x}_t} + (\tilde{\boldsymbol{\Psi}}_i^{\text{trim}})^T\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t}\right]\Delta \mathbf{x}_t + \left[\mathbf{J}_i^{\text{trim}} - \bar{\mathbf{J}}_i^{\text{trim}} + (\tilde{\boldsymbol{\Psi}}_i^{\text{trim}})^T\mathbf{R}_i\right] = \mathbf{0}, \quad i = 1, ..., N_m; \end{aligned} \tag{A.3}$$

where $\tilde{\mathbf{\Psi}}_i^{\text{trim}}$ has the same definition as the trim adjoint $\mathbf{\Psi}_i^{\text{trim}}$, while the latter is evaluated at converged flow solutions, *i.e.*, when $\mathbf{R}_i = \mathbf{0}$. If $\Delta\mathbf{U}_i$ and $\Delta\mathbf{x}_t$ are updated simultaneously, the system requires concurrent adjoint solves for $\tilde{\mathbf{\Psi}}_i^{\text{trim}}$, through which the flow solver module may need to be re-designed. Instead in this paper, we fix $\mathbf{x}_t$ first and solve for $\mathbf{U}_i$ to enforce the flow equations $\mathbf{R}_i = \mathbf{0}$, then update the trim variables using Eqn. A.3 which then simplifies to

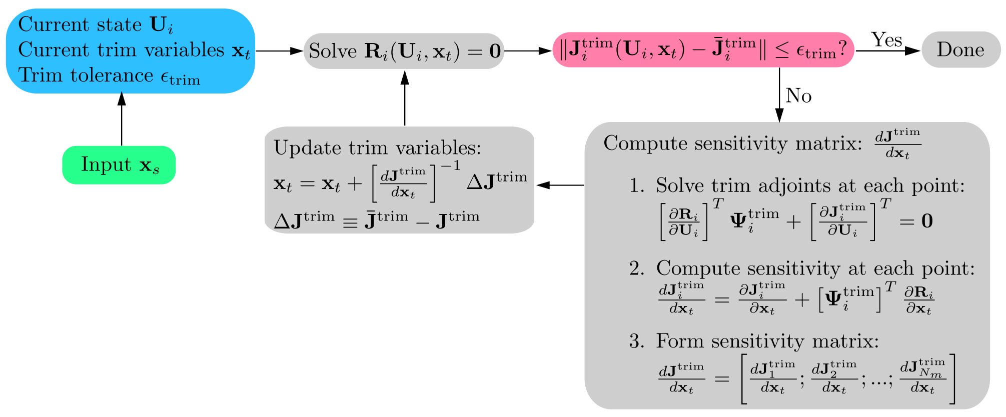
$$\begin{aligned}
\left[\frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{x}_t} + (\mathbf{\Psi}_i^{\text{trim}})^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t}\right] \Delta\mathbf{x}_t + \left[\mathbf{J}_i^{\text{trim}} - \bar{\mathbf{J}}_i^{\text{trim}}\right] &= \mathbf{0}, \\
\frac{d\mathbf{J}_i^{\text{trim}}}{d\mathbf{x}_t}\Delta\mathbf{x}_t + \left[\mathbf{J}_i^{\text{trim}} - \bar{\mathbf{J}}_i^{\text{trim}}\right] &= \mathbf{0}, \quad i = 1, ..., N_m; \\
\implies \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t}\Delta\mathbf{x}_t + \left[\mathbf{J}^{\text{trim}} - \bar{\mathbf{J}}^{\text{trim}}\right] &= \mathbf{0},
\end{aligned} \tag{A.4}$$

where

$$\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} = \begin{bmatrix} \text{---} & \frac{d\mathbf{J}_1^{\text{trim}}}{d\mathbf{x}_t} & \text{---} \\ \text{---} & \frac{d\mathbf{J}_2^{\text{trim}}}{d\mathbf{x}_t} & \text{---} \\ & \vdots & \\ \text{---} & \frac{d\mathbf{J}_{N_m}^{\text{trim}}}{d\mathbf{x}_t} & \text{---} \end{bmatrix}, \qquad \mathbf{J}^{\text{trim}} - \bar{\mathbf{J}}^{\text{trim}} = \begin{bmatrix} \mathbf{J}_1^{\text{trim}} - \bar{\mathbf{J}}_1^{\text{trim}} \\ \mathbf{J}_2^{\text{trim}} - \bar{\mathbf{J}}_2^{\text{trim}} \\ \vdots \\ \mathbf{J}_{N_m}^{\text{trim}} - \bar{\mathbf{J}}_{N_m}^{\text{trim}} \end{bmatrix}. \tag{A.5}$$

The process iterates until Eqn. A.1 is satisfied, as shown in Figure A.1. Although this approach is less efficient than solving $\mathbf{x}_t$ and $\mathbf{U}_i$ simultaneously, it requires minimal rewrites of the flow solver code. For the lift-constrained problem considered in this work, $\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t}$ is diagonal, *i.e.*, the trim constraints at each design point are decoupled and can be iterated independently.

The trimming process converges successfully most of the time when the sensitivity $\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t}$ is measured accurately, however, we do observe failures of the trimming process for some intermediate designs in the line search process, though these designs are never accepted as penalty is added if the lift constraint is not satisfied. More detailed investigations on these airfoil shapes show non-uniqueness of the solution at the same flow condition. The resulting bifurcation of the trim outputs hinders the convergence of the trimming process, which has also been observed by other researchers [50, 51, 52]. In this paper, we restrict the maximum number of trimming iterations to exit the loop in Figure A.1 if the trimming fails. Improved treatment of such situations will be considered in future work.



Current state $\mathbf{U}_i$
Current trim variables $\mathbf{x}_t$
Trim tolerance ϵ_{trim}
Input $\mathbf{x}_s$
Solve $\mathbf{R}_i(\mathbf{U}_i, \mathbf{x}_t) = \mathbf{0}$
$\|\mathbf{J}_i^{\text{trim}}(\mathbf{U}_i, \mathbf{x}_t) - \bar{\mathbf{J}}_i^{\text{trim}}\| \leq \epsilon_{\text{trim}}$?
Yes
Done
No
Update trim variables:
$\mathbf{x}_t = \mathbf{x}_t + \left[\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t}\right]^{-1} \Delta\mathbf{J}^{\text{trim}}$
$\Delta\mathbf{J}^{\text{trim}} \equiv \bar{\mathbf{J}}^{\text{trim}} - \mathbf{J}^{\text{trim}}$
Compute sensitivity matrix: $\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t}$
1. Solve trim adjoints at each point:
$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i}\right]^T \mathbf{\Psi}_i^{\text{trim}} + \left[\frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{U}_i}\right]^T = \mathbf{0}$
2. Compute sensitivity at each point:
$\frac{d\mathbf{J}_i^{\text{trim}}}{d\mathbf{x}_t} = \frac{\partial \mathbf{J}_i^{\text{trim}}}{\partial \mathbf{x}_t} + \left[\mathbf{\Psi}_i^{\text{trim}}\right]^T \frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_t}$
3. Form sensitivity matrix:
$\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}_t} = \left[\frac{d\mathbf{J}_1^{\text{trim}}}{d\mathbf{x}_t}; \frac{d\mathbf{J}_2^{\text{trim}}}{d\mathbf{x}_t}; ...; \frac{d\mathbf{J}_{N_m}^{\text{trim}}}{d\mathbf{x}_t}\right]$

Figure A.1: Feedback trimming process

References

[1] P. Hajela, Nongradient methods in multidisciplinary design optimization-status and potential, Journal of Aircraft 36 (1) (1999) 255–265. doi:10.2514/2.2432.

[2] A. Jameson, Aerodynamic design via control theory, Journal of Scientific Computing 3 (3) (1988) 233–260. doi:10.1007/bf01061285.

[3] W. Anderson, V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, Computers & Fluids 28 (4-5) (1999) 443–480. doi:10.1016/s0045-7930(98)00041-3.

[4] M. B. Giles, N. A. Pierce, An introduction to the adjoint approach to design, Flow, Turbulence and Combustion 65 (3) (2000) 393–415. doi:10.1023/a:1011430410075.

[5] M. B. Giles, M. C. Duta, J.-D. Müller, N. A. Pierce, Algorithm developments for discrete adjoint methods, AIAA Journal 41 (2) (2003) 198–205. doi:10.2514/2.1961.

[6] J. Brezillon, N. Gauger, 2D and 3D aerodynamic shape optimisation using the adjoint approach, Aerospace Science and Technology 8 (8) (2004) 715–727. doi:10.1016/j.ast.2004.07.006.

[7] R. Djeddi, K. Ekici, FDOT: A fast, memory-efficient and automated approach for discrete adjoint sensitivity analysis using the operator overloading technique, Aerospace Science and Technology 91 (2019) 159–174. doi:10.1016/j.ast.2019.05.004.

[8] G. K. W. Kenway, C. A. Mader, P. He, J. R. R. A. Martins, Effective adjoint approaches for computational fluid dynamics, Progress in Aerospace Sciences (In press). doi:10.1016/j.paerosci.2019.05.002.

[9] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, D. Saunders, Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1, Journal of Aircraft 36 (1) (1999) 51–60. doi:10.2514/2.2413.

[10] M. Nemec, D. W. Zingg, T. H. Pulliam, Multipoint and multi-objective aerodynamic shape optimization, AIAA Journal 42 (6) (2004) 1057–1065. doi:10.2514/1.10415.

[11] D. W. Zingg, S. Elias, Aerodynamic optimization under a range of operating conditions, AIAA Journal 44 (11) (2006) 2787–2792. doi:10.2514/1.23658.

[12] G. K. W. Kenway, J. R. R. A. Martins, Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration, Journal of Aircraft 51 (1) (2014) 144–160. doi:10.2514/1.c032150.

[13] G. K. W. Kenway, J. R. R. A. Martins, Multipoint aerodynamic shape optimization investigations of the common research model wing, AIAA Journal 54 (1) (2016) 113–128. doi:10.2514/1.j054154.

[14] J. E. Hicken, J. J. Alonso, PDE-constrained optimization with error estimation and control, Journal of Computational Physics 263 (2014) 136–150. doi:10.1016/j.jcp.2013.12.050.

[15] G. Chen, K. J. Fidkowski, Discretization error control for constrained aerodynamic shape optimization, Journal of Computational Physics 387 (1) (2019) 163–185. doi:10.1016/j.jcp.2019.02.038.

[16] R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, Acta Numerica 10 (2001) 1–102. doi:10.1017/s0962492901000010.

[17] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, Journal of Computational Physics 183 (2) (2002) 508–532. doi:10.1006/jcph.2002.7206.

[18] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows, Journal of Computational Physics 187 (1) (2003) 22–46. doi:10.1016/s0021-9991(03)00074-3.

[19] M. A. Park, Adjoint-based, three-dimensional error prediction and grid adaptation, AIAA Journal 42 (9) (2004) 1854–1862. doi:10.2514/1.10051.

[20] K. J. Fidkowski, D. L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, AIAA Journal 49 (4) (2011) 673–694. doi:10.2514/1.j050073.

[21] K. J. Fidkowski, Output-based space-time mesh optimization for unsteady flows using continuous-in-time adjoints, Journal of Computational Physics 341 (2017) 258–277. doi:10.1016/j.jcp.2017.04.005.

[22] M. Kouhi, E. Oñate, D. Mavriplis, Adjoint-based adaptive finite element method for the compressible Euler equations using finite calculus, Aerospace Science and Technology 46 (2015) 422–435. doi:10.1016/j.ast.2015.08.008.

[23] G. L. Halila, G. Chen, Y. Shi, K. J. Fidkowski, J. R. Martins, M. T. de Mendonça, High-reynolds number transitional flow simulation via parabolized stability equations with an adaptive RANS solver, Aerospace Science and Technology 91 (1) (2019) 321–336. doi:10.1016/j.ast.2019.05.018.

[24] J. Lu, An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, available: http://hdl.handle.net/1721.1/34134 (2005).

[25] M. Nemec, M. Aftosmis, Output error estimates and mesh refinement in aerodynamic shape optimization, in: 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, 2013, p. 0865. doi:10.2514/6.2013-865.

[26] D. Li, R. Hartmann, Adjoint-based airfoil optimization with discretization error control, International Journal for Numerical Methods in Fluids 77 (1) (2015) 1–17. doi:10.1002/fld.3971.

[27] G. Chen, K. J. Fidkowski, Airfoil shape optimization using output-based adapted meshes, in: 23rd AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 2017, p. 3102. doi:10.2514/6.2017-3102.

[28] 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.

[29] B. A. Rothacker, M. Ceze, K. Fidkowski, Adjoint-based error estimation and mesh adaptation for problems with output constraints, in: 32nd AIAA Applied Aerodynamics Conference, American Institute of Aeronautics and Astronautics, 2014, p. 2576. doi:10.2514/6.2014-2576.

[30] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004. doi:10.1017/cbo9780511804441.

[31] S. Allmaras, F. Johnson, P. Spalart, Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model, Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902 (2012).

[32] W. Reed, T. Hill, Triangular mesh methods for the neutron transport equation, Tech. rep., Los Alamos Scientific Lab, available: https://www.osti.gov/servlets/purl/4491151 (October 1973).

[33] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, Journal of Computational Physics 131 (2) (1997) 267–279. doi:10.1006/jcph.1996.5572.

[34] B. Cockburn, C.-W. Shu, Runge–kutta discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (3) (2001) 173–261. doi:10.1023/a:1012873910884.

[35] R. Hartmann, J. Held, T. Leicht, F. Prill, Discontinuous Galerkin methods for computational aerodynamics — 3D adaptive flow simulation with the DLR PADGE code, Aerospace Science and Technology 14 (7) (2010) 512–519. doi:10.1016/j.ast.2010.04.002.

[36] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357–372. doi:10.1016/0021-9991(81)90128-5.

[37] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations, in: Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2000, pp. 197–208. doi:10.1007/978-3-642-59721-3_14.

[38] M. Yano, An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, available: http://hdl.handle.net/1721.1/76090, (2012).

[39] F. Hecht, BAMG: Bidimensional anisotropic mesh generator, INRIA–Rocquencourt, France, available: https://www.ljll.math.upmc.fr/hecht/ftp/bamg/ (1998).

[40] X. Pennec, P. Fillard, N. Ayache, A Riemannian framework for tensor computing, International Journal of Computer Vision 66 (1) (2006) 41–66. doi:10.1007/s11263-005-3222-z.

[41] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, O. Pironneau, Anisotropic unstructured mesh adaption for flow simulations, International Journal for Numerical Methods in Fluids 25 (4) (1997) 475–491. doi:10.1002/(sici)1097-0363(19970830)25:4<475::aid-fld575>3.0.co;2-6.

[42] P. Frey, F. Alauzet, Anisotropic mesh adaptation for CFD computations, Computer Methods in Applied Mechanics and Engineering 194 (48-49) (2005) 5068–5082. doi:10.1016/j.cma.2004.11.025.

[43] A. Loseille, F. Alauzet, Continuous mesh framework part I: Well-posed continuous interpolation error, SIAM Journal on Numerical Analysis 49 (1) (2011) 38–60. doi:10.1137/090754078.

[44] K. J. Fidkowski, D. L. Darmofal, A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier–Stokes equations, Journal of Computational Physics 225 (2) (2007) 1653–1672. doi:10.1016/j.jcp.2007.02.007.

[45] K. Fidkowski, A local sampling approach to anisotropic metric-based mesh optimization, in: 54th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics, 2016, p. 0835. doi:10.2514/6.2016-0835.

[46] R. M. Hicks, P. A. Henne, Wing design by numerical optimization, Journal of Aircraft 15 (7) (1978) 407–412. doi:10.2514/3.58379.

[47] H.-Y. Wu, S. Yang, F. Liu, H.-M. Tsai, Comparisons of three geometric representations of airfoils for aerodynamic optimization, in: 16th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 2003, p. 4095. doi:10.2514/6.2003-4095.

[48] 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.

[49] D. Kraft, A software package for sequential quadratic programming, Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center–Institute for Flight Mechanics, Köln, Germany (1988).

[50] A. Jameson, Airfoils admitting non-unique solutions of the Euler equations, in: 22nd Fluid Dynamics, Plasma Dynamics and Lasers Conference, American Institute of Aeronautics and Astronautics, 1991. doi:10.2514/6.1991-1625.

[51] M. M. Hafez, W. H. Guo, Nonuniqueness of transonic flows, Acta Mechanica 138 (3-4) (1999) 177–184. doi:10.1007/bf01291843.

[52] A. Kuzmin, Non-unique transonic flows over airfoils, Computers & Fluids 63 (2012) 1–8. doi:10.1016/j.compfluid.2012.04.001.