Journal of Computational Physics (2019)

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Output-based mesh adaptation for multifidelity PDE-constrained optimization

Guodong Chen*, Krzysztof J. Fidkowski

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, United States

ARTICLEINFO

Article history:

Keywords: Multifidelity PDE-constrained optimization, Discretization error, Adjoint-based error estimation, Output-based mesh adaptation, Unstructured mesh optimization

ABSTRACT

In this paper, we present a method to control the discretization error in PDE-constrained optimization problems using meshes adapted via adjoint-based error estimates. The method is capable of dealing with optimization problems containing output constraints in which the mesh is adapted and itself optimized to predict both objective outputs and constraint outputs with appropriate accuracy. We use unstructured mesh optimization to maximize accuracy of the results for a given number of degrees of freedom. The error estimates drive a multifidelity optimization process, preventing over-optimization on a coarse mesh, or over-refinement on an undesired design. We demonstrate the accuracy and efficiency of our proposed method on several airfoil optimization problems governed by the compressible Navier-Stokes equations. We expect the framework to be even more important for optimization problems with complex systems, dramatic design changes, or high-accuracy requirements.

1. Introduction

PDE-constrained optimization refers to the optimization of a system described by partial differential equations (PDEs). It arises in many engineering and physics applications, as many complex systems are governed by PDEs, including the heat equation in thermodynamics, the Navier-Stokes equations in fluid mechanics, Maxwell's equations in electromagnetism, the elasticity equations in structural dynamics, etc. Analytic solutions to these equations are typically not accessible, so the design or optimization process in these systems highly depends on the numerical solution of the underlying PDEs.

The design process combines the PDE analysis with numerical optimization methods. The PDE analysis (*state problem*) is to solve the PDE for the *state variables*, given boundary and initial conditions. The optimization process explores the design space to minimize the *objective function* based on the *state variables*. For the optimization algorithms, both gradient-based and gradient-free methods can be used. Gradient-free methods, such as genetic algorithms

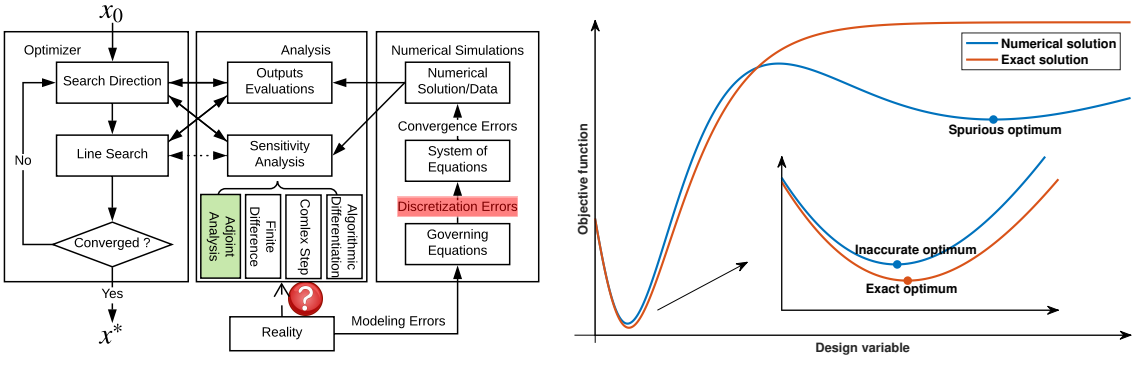
*Corresponding author.
E-mail address: cgderic@umich.edu (Guodong Chen), kfid@umich.edu (Krzysztof J. Fidkowski)

and neural networks 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 calculation [2].

With the increasing power of modern computers and highly-developed numerical methods, numerical simulation of large PDE systems is nowadays commonly carried out in design analysis. However, the number of functional evaluations needed to reach the optimum with gradient-free methods is still computationally taxing for high-dimensional problems, so that gradient-based methods are extensively used in engineering design. Gradients with respect to each design parameter are needed at every optimization iteration, which requires an accurate and efficient method for estimating the sensitivities. Methods such as finite differencing, complex-step derivative approximation [3], and algorithmic differentiation [4] depend on the number of design parameters, while the adjoint method [2, 5, 6, 7, 8] is largely independent of the number of design parameters.

A diagram of practical PDE-constrained optimization with gradient based methods is shown in Figure 1(a). Many error sources exist in the PDE analysis, which means that the optimizer is working on the information with numerical errors. There are modeling errors when the real-world system is modelled with simplification assumptions, numerical errors during the discretization of the PDE on the computational domain, and convergence errors when solving the discretized system of equations. Modeling errors inherent to the PDEs can be reduced by model validation or choosing more complex models, but the estimation of these errors is not addressed by our present work. Instead, we aim to efficiently and robustly solve optimization problems in which the chosen model is assumed to be exact. In addition, we do not address convergence errors, which can typically be controlled by appropriate solver settings. Therefore, the numerical errors in the PDE-constrained optimization in this paper refer to the errors caused by the discretization of the continuous equations on a finite-dimensional space.

Discretization errors strongly affect the reliability of the optimization objective and sensitivity calculations, and hence the quality of the optimization results. Numerical errors appear in every single design analysis during the optimization, and the effect becomes even more severe when the optimizer sequentially uses inaccurate data from the numerical simulations. Figure 1(b) shows a simple optimization problem constrained by one dimensional advection-diffusion PDEs. The numerical solution is from solves on a coarse uniformly distributed discretization. There is only one local minimum for the original continuous problem, while for the discrete numerical solution, we find a spurious local minimum caused purely by numerical errors induced by discretization. Depending on the starting point, the optimizer may (a) get stuck to a spurious optimum created by numerical errors, or (b) work on the numerical errors rather than on physics and converge to an incorrect optimum. In order to avoid these undesired behaviors, the discretization errors must be carefully controlled in the optimization.



(a) PDE-constrained optimization flow chart

(b) Example of PDE-constrained optimization with numerical error

Fig. 1. PDE-constrained optimization with numerical error

For a single fixed design, the simulation accuracy and efficiency can be dramatically improved by adaptive methods in which discretization is iteratively improved through local mesh refinement and/or approximation order increment. Residual-based and feature-based error estimates and mesh adaptation can be robust for elliptic PDEs like those of structural elasticity [9, 10, 11, 12], and several contributions have been made to integrate these approaches to structure optimization problems [13, 14, 15]. However for hyperbolic systems like those governing fluid dynamics,

errors can propagate by the convection-dominant nature of the system, making the prediction of regions requiring high resolution non-intuitive. An alternative and more robust way to identify the important areas for the output of interest is through adjoint solutions, which provide the sensitivity of the output to residual perturbations. The idea to combine output error estimation and gradient-based optimization is natural, as both methods require output adjoint solutions. Even though adjoint-based error estimation and mesh adaption have been successfully demonstrated in many engineering problems [16, 17, 18, 19, 20, 21, 22, 23, 24], their potential application to optimization problems has not received much attention. Lu [25] incorporated adjoint-based error estimation and mesh adaptation into gradient-based optimization. The constraints are realized as simple quadratic penalty functions added to the objective. Progressive optimization is used with mesh adaptation based on the error of the penalized objective. Nemec and Aftosmis [26] modified the penalty terms to avoid vanishing of constraints error when they are satisfied. Li and Hartmann [27] eliminated the constraint by trimming with an individual design variable, and introduced a multi-target adaptation algorithm in which mesh adaptation targets the objective and constraint outputs equally on a fixed fidelity. Hicken and Alonso [28] used the gradient norm error as the refinement indicator, actively control the first order optimality condition, while approximating higher-order derivatives. Chen and Fidkowski [29] adopted an efficient error estimate for the optimization problems accounting the effects of general output constraints, and used it to drive the progressive optimization.

In most previous works on optimization combined with error estimation, mesh adaptation has only been used to add refinement. However, in order to control the numerical error at each fidelity (error level), the mesh may get adapted at many areas that are important for the intermediate designs but not necessary for the final optimal design, which may quantitatively decrease the efficiency of the high-fidelity optimization level. Building on our previous work [29], we employ a more sophisticated spatial adaptation method, unstructured mesh optimization through error sampling and synthesis (MOESS) proposed by Yano and Darmofal [30]. A cost-based multi-level optimization framework is developed in this work, at each optimization level the computational cost is restricted to a certain degrees of freedom. Within the same optimization fidelity, the mesh is optimized for each design to predict both the objective and constraint outputs accurately. The goal is to take the best use of a given degrees of freedom, so that the overall computational cost of the high-fidelity optimization can be reduced.

The remainder of this paper proceeds as follows. We describe general PDE-constrained optimization problems 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 respectively. 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. Optimization formulation

2.1. Continuous and discrete optimization

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

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

where $\mathcal{J} : \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ represents a scalar objective function, $\boldsymbol{\mathcal{R}}^{\text{e}} : \mathcal{U} \times \mathcal{X} \to \mathbb{R}^{n_{\text{e}}}$ and $\boldsymbol{\mathcal{R}}^{\text{ie}} : \mathcal{U} \times \mathcal{X} \to \mathbb{R}^{n_{\text{ie}}}$ denote n_e equality and n_{ie} inequality constraints, respectively. The objective and constraints are always defined by the outputs (responses) of the PDEs, which consequently depend on the state variables $\mathbf{u}$. The state $\mathbf{u}$ is the solution of the governing equations, lying in the solution space $\mathcal{U}$, which can be an infinite-dimensional space. In a variational setting, the governing PDEs can be represented by a semi-linear from,

$$\mathcal{R}(\mathbf{u}, \mathbf{v}; \mathbf{x}) = 0, \quad \forall\, \mathbf{v} \in \mathcal{V} \tag{2}$$

where $\mathcal{V}$ denotes an appropriately-defined test space for the given PDEs, and the semi-linear residual map $\mathcal{R} : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ corresponds to the weak formulation of the PDEs. The state $\mathbf{u} \in \mathcal{U}$ is solved within the design space $\mathcal{X}$ to satisfy the governing PDEs, which implicitly defines $\mathbf{u}$ as a function of $\mathbf{x}$: $\mathbf{u} = \mathbf{u}(\mathbf{x})$. Moreover, the optimal deign $\mathbf{x}$ has to be

in the feasible space $\mathcal{F}(\mathcal{X}) = \{\mathbf{x} \in \mathcal{X} : \mathcal{R}^{\text{e}}(\mathbf{u}, \mathbf{x}) = \mathbf{0}, \mathcal{R}^{\text{ie}}(\mathbf{u}, \mathbf{x}) \geq \mathbf{0}\}$ that satisfies the constraints. Depending on the optimization algorithm, intermediate designs in an iterative process may not be in the feasible set.

Consider an infinitesimal state perturbation $\delta\mathbf{u}$, added to the weak statement in Eqn. (2). An adjoint solution $\boldsymbol{\psi} \in \mathcal{V}$ can be defined as the sensitivity of the output of interest to the residual perturbation by the following relationship,

$$\delta\mathcal{J} = \mathcal{J}(\mathbf{u} + \delta\mathbf{u}) - \mathcal{J}(\mathbf{u}) = \delta\mathcal{R} = -\mathcal{R}(\mathbf{u} + \delta\mathbf{u}, \boldsymbol{\psi}) \tag{3}$$

The infinitesimal state and residual perturbations can be related via residual linearization,

$$\mathcal{R}(\mathbf{u} + \delta\mathbf{u}, \mathbf{v}) = \mathcal{R}'[\mathbf{u}](\delta\mathbf{u}, \mathbf{v}), \quad \forall\ \mathbf{v} \in \mathcal{V} \tag{4}$$

where the prime denotes the Fréchét derivative with respect to the arguments in the square brackets. Also linearizing the output, we have

$$\delta\mathcal{J} = \mathcal{J}'[\mathbf{u}](\delta\mathbf{u}) = -\mathcal{R}(\mathbf{u} + \delta\mathbf{u}, \boldsymbol{\psi}) = -\mathcal{R}'[\mathbf{u}](\delta\mathbf{u}, \boldsymbol{\psi}) \tag{5}$$

where Eqn. (3) and Eqn. (4) are used in the second and third equalities, respectively. For these linearizations to exist, both the semilinear residual form and the output are assumed to be differentiable. In order for Eqn. (5) to be true for any perturbations, the adjoint $\boldsymbol{\psi} \in \mathcal{V}$ must be the solution of the continuous adjoint equation,

$$\mathcal{J}'[\mathbf{u}](\mathbf{w}) + \mathcal{R}'[\mathbf{u}](\mathbf{w}, \boldsymbol{\psi}) = 0, \quad \forall\ \mathbf{w} \in \mathcal{U} \tag{6}$$

However, we cannot solve the PDEs in Eqn. (2) analytically most of the time, so we discretize the PDEs on the computational domain, then the original PDEs and adjoint equation can be reformulated as: determine $\mathbf{u}_h \in \mathcal{U}_h$ and $\boldsymbol{\psi}_h \in \mathcal{V}_h$ such that

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h) = 0, \quad \forall\ \mathbf{v}_h \in \mathcal{V}_h \tag{7}$$

$$\mathcal{J}_h'[\mathbf{u}_h](\mathbf{w}_h) + \mathcal{R}_h'[\mathbf{u}_h](\mathbf{w}_h, \boldsymbol{\psi}_h) = 0, \quad \forall\ \mathbf{w}_h \in \mathcal{U}_h \tag{8}$$

where $\mathcal{U}_h$ and $\mathcal{V}_h$ are finite dimensional functional spaces. The subscript h indicates a discretization of the PDEs, including the approximation order and the spatial discretization. To simplify the analysis, we transfer the variational statements into vector form, assuming the trial and test spaces are the same. Consider a complete basis set on the state space and adjoint space, $\mathcal{U} = \mathcal{V} = \text{span}\{\phi_i, i = 1, ..., N\}$, where N denotes the dimension of the functional space. Then the state and adjoint solutions can be rewritten as linear combinations spanning over their approximation space, $\mathbf{u}_h = \sum_{i=1}^{N} U_{h,i}\, \phi_i$, $\boldsymbol{\psi}_h = \sum_{i=1}^{N} \Psi_{h,i}\, \phi_i$, so that $\mathbf{u}_h$ and $\boldsymbol{\psi}_h$ can be uniquely defined by their associated coefficient vectors $\mathbf{U}_h = [U_{h,i}]_{i=1}^{N}$ and $\boldsymbol{\Psi}_h = [\Psi_{h,i}]_{i=1}^{N}$. We can also define the residual vector as $\mathbf{R}_h = [\mathcal{R}_h(\mathbf{u_h}, \phi_i)]_{i=1}^{N}$ and the output as $J_h(\mathbf{U_h}) = \mathcal{J}_h(\mathbf{u}_h)$, now the state and adjoint equations can be written as

$$\mathbf{R}_h(\mathbf{U}_h) = \mathbf{0} \tag{9}$$

$$\left(\frac{\partial J_h}{\partial \mathbf{U}_h}\right)^T + \left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right)^T \boldsymbol{\Psi}_h = \mathbf{0} \tag{10}$$

Consequently, a fully discretized form of the original optimization problem, which augments the constraints with the state equations, can be stated as

$$\begin{aligned} \min_{\mathbf{x}} \quad & J_h(\mathbf{U}_h, \mathbf{x}), \quad \mathbf{U}_h \in \mathbb{R}^N,\ \mathbf{x} \in \mathcal{X} \\ \text{s.t.} \quad & \mathbf{R}_h(\mathbf{U}_h, \mathbf{x}) = \mathbf{0} \\ & \mathbf{R}_h^{\text{e}}(\mathbf{U}_h, \mathbf{x}) = \mathbf{0} \\ & \mathbf{R}_h^{\text{ie}}(\mathbf{U}_h, \mathbf{x}) \geq \mathbf{0} \end{aligned} \tag{11}$$

The equality and inequality constraints are also vectorized in the equation above. In this paper, continuous and discrete optimization refer to the optimization governed by PDEs in continuous and discretized form, while in other contexts these terms may refer to the optimizations with continuous and discrete design spaces.

2.2. Optimization via the adjoint

Inactive inequality constraints $\mathbf{R}^{\text{ie}}_{\text{ia}}$, do not affect the optimization explicitly, while the active ones $\mathbf{R}^{\text{ie}}_{\text{a}}$ behave like equality constraints. We omit the subscript h here for simpler exposition. In general, the inequality constraints can also be transformed into equality constraints with non-negative slack variables [31]. For simplicity, we only consider the active inequality constraints and the equality constraints, put together into one vector of trim constraints, $\mathbf{R}^{\text{trim}} = [\mathbf{R}^{\text{e}} \;\; \mathbf{R}^{\text{ie}}_{\text{a}}]^T$,

$$\mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}) = \mathbf{J}^{\text{trim}}(\mathbf{U}, \mathbf{x}) - \bar{\mathbf{J}}^{\text{trim}} = \mathbf{0} \tag{12}$$

where $\bar{\mathbf{J}}^{\text{trim}}$ is a set of target trim outputs, for example, the target load in a structure optimization problem or the target lift in an airfoil drag minimization problem. In order to distinguish the trim outputs from the objective output, we denote the latter by J^{adapt}, as the objective output is directly targeted for adaptation.

The adjoint-based optimization is equivalent to searching for the stationary point of the Lagrangian function that augments the governing equations with trim constraints,

$$\mathcal{L}(\mathbf{U}, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J^{\text{adapt}}(\mathbf{U}, \mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{R}(\mathbf{U}, \mathbf{x}) + \boldsymbol{\mu}^T \mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}) \tag{13}$$

where $\boldsymbol{\lambda}, \boldsymbol{\mu}$ are the Lagrange multipliers associated with the state equations and the trim constraints, respectively.

By requiring stationarity with respect to the design variables $\mathbf{x}$, the states $\mathbf{U}$, and the Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, we arrive at the *first order necessary condition for optimality*, or the *Karush-Kuhn-Tucker condition*,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}} = \mathbf{0} \tag{14a}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{U}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{U}} = \mathbf{0} \tag{14b}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = \mathbf{R}(\mathbf{U}, \mathbf{x}) = \mathbf{0} \tag{14c}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}} = \mathbf{R}^{\text{trim}}(\mathbf{U}, \mathbf{x}) = \mathbf{0} \tag{14d}$$

The above optimality condition can be a large, coupled, nonlinear system, at least larger and more coupled compared to the state and adjoint equations, especially for high-dimensional optimization problems. Solving this system in the full space is intractable for complex systems, when the governing PDE solve is already challenging. One popular alternative is to reduce the system size by eliminating the states $\mathbf{U}$ and the Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, correspondingly, state and adjoint equations. The resulting system thus becomes a manageable one only with respect to the design variables $\mathbf{x}$. These kind of methods are termed as *reduced space* methods in contrast to the *full space* methods, where the whole system is solved simultaneously. In this paper, we used the former approach for several reasons. First of all, the PDE solver for large-scale state and adjoint equations have been developed for decades and are now very efficient and robust. Another reason is that the whole system is often very ill-conditioned, whereas the four subsystems are typically better conditioned individually [32]. Additionally, most of the standard optimization packages fail with very high dimensional systems.

In the *reduced space* method, we solve the state equations for a given deign each time, in other words, Eqn. (14c) is always satisfied during the optimization. Then we can choose $\boldsymbol{\lambda}$ such that Eqn. (14b) is enforced after each states solve,

$$\boldsymbol{\lambda}^T = -\left(\frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{U}}\right) \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^{-1} = (\boldsymbol{\Psi}^{\text{adapt}} + \boldsymbol{\Psi}^{\text{trim}} \boldsymbol{\mu})^T \tag{15}$$

Eqn. (15) gives a coupled adjoint variable $\boldsymbol{\lambda}$ that incorporates the adjoints of both the objective and the trim outputs, $\boldsymbol{\Psi}^{\text{adapt}}$ and $\boldsymbol{\Psi}^{\text{trim}}$, which satisfy the discretized adjoint equation in Eqn. (10)

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \boldsymbol{\Psi}^{\text{adapt}} + \frac{\partial J^{\text{adapt}}}{\partial \mathbf{U}}^T = \mathbf{0}, \qquad \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \boldsymbol{\Psi}^{\text{trim}} + \frac{\partial \mathbf{J}^{\text{trim}}}{\partial \mathbf{U}}^T = \mathbf{0} \tag{16}$$

With this specific choice of $\boldsymbol{\lambda}$, we can evaluate the gradient of the Lagrangian function with respect to the design

variables, starting with Eqn. (14a),

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{x}} &= \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \boldsymbol{\mu}^T \frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}} \\ &= \frac{\partial J^{\text{adapt}}}{\partial \mathbf{x}} + (\boldsymbol{\Psi}^{\text{adapt}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \boldsymbol{\mu}^T \left[\frac{\partial \mathbf{R}^{\text{trim}}}{\partial \mathbf{x}} + (\boldsymbol{\Psi}^{\text{trim}})^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \right] \\ &= \frac{dJ^{\text{adapt}}}{d\mathbf{x}} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}}\end{aligned} \tag{17}$$

The last equality is obtained via adjoint-based sensitivity analysis, where $d(\cdot)/d\mathbf{x}$ denotes the total derivative with respect to design variables by considering the states as an implicit function of $\mathbf{x}$, $\mathbf{U}(\mathbf{x})$.

Now the optimization problem has been reduced to finding an optimal design $\mathbf{x}$ and the corresponding Lagrange multipliers $\boldsymbol{\mu}$ satisfying,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{x}} &= \frac{dJ^{\text{adapt}}}{d\mathbf{x}} + \boldsymbol{\mu}^T \frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}} = \mathbf{0} \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}} &= \mathbf{R}^{\text{trim}} = \mathbf{0}\end{aligned} \tag{18}$$

However, since in a practical calculation, on a finite-dimensional space, the discretization error appears in both the flow equations and the adjoint equations, optimality cannot be guaranteed even when Eqn. (18) is satisfied. The present work focuses on controlling the error in the optimization problem via error estimation and mesh adaptation.

3. Governing equations and discretization

Evaluation of the objective and the constraints at each optimization step relies on the numerical simulation of the system. The governing PDE considered here is transport in conservation form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0} \tag{19}$$

which encompasses scalar advection-diffusion and the compressible Navier-Stokes equations. For the latter, $\mathbf{u}$ is the conservative state vector composed by the flow variables, $\vec{\mathbf{F}}$ denotes the total invicid and viscous flux vectors, and $\mathbf{S}$ represents the source term required when modeling turbulence. When running Reynolds-averaged turbulent cases, we use the Spalart-Allmaras one-equation model, with a negative turbulent-viscosity modification [33].

We discretize Eqn. (19) with the discontinuous Galerkin (DG) finite-element method, which is suitable for high-order accuracy and hp-refinement [34, 35, 36]. The computational domain Ω is divided into a shape-regular mesh $\mathcal{T}_h$ consisting of N_e non-overlapping elements Ω_e, $\mathcal{T}_h = \{\Omega_e : \bigcup_{e=1}^{Ne} \Omega_e = \Omega, \bigcap_{e=1}^{Ne} \Omega_e = \emptyset\}$. In DG, the state is approximated by piece-wise polynomials lying on the approximation space $\mathcal{V}_h^p$, with no continuity constraints imposed on the approximations between adjacent elements. The approximation space consists of element-wise polynomials and is defined as $\mathcal{V}_h^p = \{v_h \in L^2(\Omega) : v_h|_{\Omega_e} \in \mathcal{P}^{p_e}, \forall \Omega_e \in \mathcal{T}_h\}$, where $\mathcal{P}^{p_e}$ denotes polynomials of order p_e on element Ω_e, a distribution that is not necessary uniform throughout the mesh. The weak form of Eqn. (19) follows from multiplying the equation by test functions (taken from the approximation space), integrating by parts, and coupling elements via unique inter-element fluxes. We use the Roe approximate Riemann solver [37] for the invicid flux, and the second form of Bassi and Rebay (BR2) for viscous flux [38]. Choosing a basis for the test and trial spaces yields a system of nonlinear, algebraic equations in the form of Eqn. (9)

$$\mathbf{R}_H(\mathbf{U}_H, \mathbf{x}) = \mathbf{0} \tag{20}$$

Here, $\mathbf{R}_H$ is the residual vector, a nonlinear function of the discrete state vector $\mathbf{U}_H$ and the design variables $\mathbf{x}$. For the steady state problems considered in this work, $\mathbf{R}_H$ is the discrete spatial residual vector. The subscript H refers to discretization fidelity of the approximation/test space with respect to the approximation order and mesh refinement.

4. Objective error estimation

4.1. Adjoint-based error estimation

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

$$\text{output error: } \delta J \equiv J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \tag{21}$$

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_e, to $p_e + 1$. We do not solve the nonlinear fine-space flow problem for the error prediction, and instead we use the linear fine-space adjoint solution, $\mathbf{\Psi}_h$. The adjoint weights the residual perturbation to produce an output perturbation,

$$\begin{aligned}
\delta J &= J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \\
&= J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) = \frac{\partial J_h}{\partial U_h}\delta\mathbf{U} \\
&= -\mathbf{\Psi}_h^T \delta\mathbf{R}_h = -\mathbf{\Psi}_h^T[\mathbf{R}_h(\mathbf{U}_h^H) - \mathbf{R}_h(\mathbf{U}_h)] \\
&= -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)
\end{aligned} \tag{22}$$

where $\mathbf{U}_h$ is the (hypothetical) exact solution on the fine space, and $\mathbf{U}_h^H$ 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. (22) originates from the small perturbation assumptions, 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. Error estimation for optimization problems

Normally, error estimation is applied only to the output in which we are most interested, i.e. the objective. However, our optimization problem requires the simultaneous solution of flow equations and constraints, i.e. trim outputs. The numerical error of the trim outputs may indirectly affect the calculation of the objective [39]. To take this effect into account, the coupled adjoint should be used for the error estimation.

Consider a given design $\mathbf{x}$, and suppose that the error of the objective only comes from the inexact solution $\mathbf{U}_h^H$. We can estimate the error in the objective with the linearization given by Eqn. (14b),

$$\begin{aligned}
\delta J^{\text{adapt}} &= -\boldsymbol{\lambda}_h^T \delta\mathbf{R}_h - \boldsymbol{\mu}_h^T \delta\mathbf{R}_h^{\text{trim}} = -\boldsymbol{\lambda}_h^T \mathbf{R}_h(\mathbf{U}_h^H) - \boldsymbol{\mu}_h^T \delta\mathbf{R}_h^{\text{trim}} \\
&= -(\mathbf{\Psi}_h^{\text{adapt}} + \mathbf{\Psi}_h^{\text{trim}}\boldsymbol{\mu}_h)^T \mathbf{R}_h(\mathbf{U}_h^H) - \boldsymbol{\mu}_h^T(\mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H) - \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h)) \\
&= \delta J^{\text{adapt}} - \boldsymbol{\mu}_h^T(\mathbf{\Psi}_h^{\text{trim}})^T \mathbf{R}_h(\mathbf{U}_h^H) - \boldsymbol{\mu}_h^T \delta\mathbf{J}^{\text{trim}} \\
&= \delta J^{\text{adapt}}
\end{aligned} \tag{23}$$

This is consistent with the previous analysis without the trim conditions, since we keep the design fixed between the coarse and fine spaces, and because we assume that the error only comes from the inexact state solution $\mathbf{U}_h^H$. In general, however, we need to deal with both the objective error and the constraints error. The problem becomes worse if we have high accuracy in the objective while little confidence in the constraint outputs, or vice versa. If we run the optimization on the fine space and the coarse space, even with the same target constraint outputs, we will generally obtain different designs. This difference may come from the deviation of both the design parameters and the flow states, and separate error estimation and mesh adaptation for the objective and trim outputs can be inefficient.

If we consider the optimal design on the coarse space $(\mathbf{U}_H, \mathbf{x}_H)$ and the fine space $(\mathbf{U}_h, \mathbf{x}_h)$, since the optimality conditions Eqn. (14a) and Eqn. (14b) both hold now, the error comes from the inexact solution $\mathbf{U}_h^H$ as well as the deficient design $\mathbf{x}_H$,

$$\begin{aligned}
\delta J_{\text{opt}}^{\text{adapt}} &= -\boldsymbol{\lambda}_h^T \delta\mathbf{R}_h - \boldsymbol{\mu}_h^T \delta\mathbf{R}_h^{\text{trim}} = -\boldsymbol{\lambda}_h^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) - \boldsymbol{\mu}_h^T \mathbf{R}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H) \\
&= -(\mathbf{\Psi}_h^{\text{adapt}} + \mathbf{\Psi}_h^{\text{trim}}\boldsymbol{\mu}_h)^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) - \boldsymbol{\mu}_h^T(\mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H) - \bar{\mathbf{J}}^{\text{trim}}) \\
&= \delta J^{\text{adapt}}(\mathbf{x}_H) + \boldsymbol{\mu}_h^T \delta\mathbf{J}^{\text{trim}}(\mathbf{x}_H) - \boldsymbol{\mu}_h^T(\mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H) - \bar{\mathbf{J}}^{\text{trim}})
\end{aligned} \tag{24}$$

Since the definition of the outputs is often the same on the coarse and fine spaces, we have that

$$\mathbf{J}_h^{\text{trim}}(\mathbf{U}_h^H, \mathbf{x}_H) = \mathbf{J}_H^{\text{trim}}(\mathbf{U}_H, \mathbf{x}_H) = \bar{\mathbf{J}}^{\text{trim}} = \mathbf{J}_h^{\text{trim}}(\mathbf{U}_h, \mathbf{x}_h) \tag{25}$$

Hence, the last term in Eqn. (24) is often negligible for the optimal design, resulting a simpler form for the error of the optimal objective,

$$\delta J_{\text{opt}}^{\text{adapt}} = -(\boldsymbol{\Psi}_h^{\text{adapt}} + \boldsymbol{\Psi}_h^{\text{trim}}\boldsymbol{\mu}_h)^T \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_H) = \delta J^{\text{adapt}}(\mathbf{x}_H) + \boldsymbol{\mu}_h^T \delta \mathbf{J}^{\text{trim}}(\mathbf{x}_H) \tag{26}$$

The error without the subscript opt is the output error without the trim constraints. Eqn. (26) gives a prediction of optimal objective error on the coarse space due to the spatial discretization. With the advantages of adjoint-based error estimation, we avoid the expensive solves of both the the optimal design $\mathbf{x}_h$ and flow states $\mathbf{U}_h$, i.e. the whole optimization process on the fine space. However, the estimation requires the fine-space adjoints $\boldsymbol{\Psi}_h$ as well as the fine-space Lagrange multipliers $\boldsymbol{\mu}_h$. In our implementation, the fine-space adjoints $\boldsymbol{\Psi}_h$ are approximated by reconstructing the coarse-space adjoints $\boldsymbol{\Psi}_H$ [23], while the Lagrange multipliers are extracted from the optimizer on the coarse space.

Eqn. (26) provides the objective error estimate of the optimal design, which does not always hold during the optimization process. Thus, it is neither the error of the objective nor the error of the constraints when the design is away from optimal. However, it couples the objective error and constraints error, giving a better error level for the whole optimization problem, so it is expected to serve as a better adaptation indicator for the optimization or constraint problems. Moreover, in the multifidelity optimization framework, when most of the mesh adaptation happens after a successful optimization on the current fidelity, Eqn. (26) works better than the objective error without constraints. In this paper, we allow violation of the constraints during the optimization process, the constraints are only enforced for the optimal design, while for the methods requiring the exploration path to be always feasible, Eqn. (25) holds for every iteration, thereby the objective error estimate in Eqn. (26) is valid for all the intermediate designs.

5. Mesh refinement and optimization

5.1. Adaptation indicator

For a single output of interest, the error estimate can be localized in each element and serves as an indicator for mesh adaptation. A common approach is to keep track of the elemental error contribution, taking its absolute value as the indicator.

$$\mathcal{E} = J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) = -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) = -\sum_{e=1}^{N_e} \boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H), \qquad \epsilon_e \equiv \left|\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)\right| \tag{27}$$

where $\mathcal{E}$ denotes the total output error estimate, $\epsilon_e \geq 0$ is the error indicator for element Ω_e.

For the optimization problem, the error estimate in Eqn. (26) can also be localized in each element and guides the mesh adaptation.

$$\epsilon_e = \left|-(\boldsymbol{\Psi}_{h,e}^{\text{adapt}} + \boldsymbol{\Psi}_{h,e}^{\text{trim}}\boldsymbol{\mu}_h)^T \mathbf{R}_{h,e}(\mathbf{U}_h^H, \mathbf{x}_H)\right| = \left|\delta J_{h,e}^{\text{adapt}} + \boldsymbol{\mu}_h^T \delta \mathbf{J}_{h,e}^{\text{trim}}\right| \leq \epsilon_e^{\text{adapt}} + \left|\boldsymbol{\mu}_h^T\right| \boldsymbol{\epsilon}_e^{\text{trim}} = \epsilon_{e,\text{con}} \tag{28}$$

where ϵ_e allows cancellations between objective and constraints error indicators, while $\epsilon_{e,\text{con}}$ provides a more conservative error indicator.

A naive adaptation strategy is to adapt a fixed fraction of the mesh at each adaptation cycle, in which the elements with highest errors are flagged for refinement. However, such a strategy cannot detect strong directional features, such as shocks or boundary layers in flow problems. In order to provide anisotropic resolution, we have to include the solution anisotropy with the error estimate in the adaptation techniques. We consider two anisotropic adaptation strategies here: solution Hessian-based anisotropic mesh adaptation, mesh optimization via error sampling and error synthesis (MOESS).

5.2. Metric-based remeshing

A Riemannian metric field, $\mathcal{M}(\vec{x})$, is a smoothly varying field of symmetric positive definite (SPD) tensors that can be used to encode anisotropic information of the computational mesh, including desired mesh sizes and stretching directions. At any point in the physical space, $\vec{x}$, the metric tensor $\mathcal{M}(\vec{x})$ provides a "yardstick" for measuring the distance from $\vec{x}$ to another point infinitesimally far away, $\vec{x} + \delta\vec{x}$. The distance under the Riemannian metric is given by

$$\delta l = \sqrt{\delta\vec{x}^T \mathcal{M}(\vec{x})\delta\vec{x}} \tag{29}$$

After choosing a Cartesian coordinate system and basis for d-dimensional physical space, $\mathcal{M}$ can be represented as a $d \times d$ SPD matrix. The set of points at unit metric distance is an ellipse in 2D or an ellipsoid in 3D: eigenvectors of $\mathcal{M}$ gives its principal axes, while the length of each axis (stretching) is the inverse square root of the corresponding eigenvalue.

A mesh that conforms to a metric field is one in which all the edges have unit length under the metric, to some tolerance. The metric-conforming mesh is not unique; however, a family of *metric-conforming* meshes have similar approximation properties [40]. The *metric-conforming* mesh generator used in this work is the Bi-dimensional Anisotropic Mesh Generator (BAMG) [41]. BAMG requires a metric field which is specified at vertices of a background mesh to generate a new mesh described by the continuous metric field. For a desired new d-dimensional simplex mesh, the *mesh-implied* metric can be obtained by solving a linear system for the $d(d+1)/2$ independent entries of $\mathcal{M}_e$ at each element. The equations in this system enforce that each of the $d(d+1)/2$ edges has unit metric length. Then the discontinuous elemental metric field is averaged to the surrounding vertices using an affine-invariant algorithm [42]. The metric-conforming mesh and mesh-implied metric give a way of converting between an anisotropic mesh and a Riemannian metric field, a so called mesh-metric duality.

5.3. Hessian-based anisotropic mesh adaptation

Given a localized error estimate, an appropriate adaptation strategy can be determined by decreasing and equally distributing the error [11]. In order to make the metric-based mesh adaptation efficient, stretched elements have to be generated in areas where the solution exhibits high anisotropy. One dominant method for detecting the anisotropy is to estimate directional interpolation error of a scalar solution [11, 43, 44]. For approximation order $p = 1$, the error of a scalar solution u over an edge E in the mesh, with unit tangent vector $\mathbf{s}$ and length h, is given by

$$\epsilon_E \propto \left|\mathbf{s}^T \mathbf{H}\mathbf{s}\right| h^2 \tag{30}$$

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

$$H_{i,j} = \frac{\partial^2 u}{\partial x_i \partial x_j}, \quad i, j \in [1, ..., d] \tag{31}$$

Suppose the edge conforms to a metric $\mathcal{M}$, assumed constant along the edge. Since the edge is of unit length under the metric measure,

$$l_{\mathcal{M}} = \sqrt{\mathbf{s}^T \mathcal{M}\mathbf{s}h^2} = 1 \tag{32}$$

Then the interpolation error and metric are related by Eqn. (30) and Eqn. (32), and with the requirements of error equidistribution, we have

$$\frac{\left|\mathbf{s}^T \mathbf{H}\mathbf{s}\right|}{\mathbf{s}^T \mathcal{M}\mathbf{s}} = C \tag{33}$$

where C is a constant defined by the desired error distribution. In order for Eqn. (33) to be valid for edges in any principal direction, $\mathcal{M}$ can be chosen as

$$\mathcal{M}_{\mathbf{H}} = \frac{1}{C}\mathbf{Q}|\mathbf{\Lambda}|\mathbf{Q}^T = \frac{1}{h_{\text{ref}}^2}\mathbf{Q}|\mathbf{\Lambda}|\mathbf{Q}^T \tag{34}$$

Here, $\mathbf{Q}$ denotes the orthonormal matrix containing the eigenvectors of $\mathbf{H}$, and $\mathbf{\Lambda}$ as the corresponding diagonal matrix containing its eigenvalues. h_{ref} controls the absolute mesh size which can be determined by the error estimates and desired error distribution [44]. Anisotropy detection based on the standard Hessian matrix is not suited for higher order interpolation, due to the linear interpolation assumption used in the derivation of the Hessian-based method.

Fidkowski and Darmofal [45] extended the Hessian-based anisotropy detection to general approximation order p by estimating the $p+1^{st}$ derivatives.

In order to equally distribute the error, we also need to predict the element size, or the number of the elements N^f, in the adapted (fine) mesh. Let n_k, not necessarily an integer, be the number of the fine-mesh elements contained in element k at the original mesh. Denoting the current element size by h^c_{ref} and the requested element size as h^f_{ref}, n_k can be approximated as

$$n_k = \left(\frac{h^c_{\text{ref}}}{h^f_{\text{ref}}} \right)^d \tag{35}$$

Given an error tolerence e_0, to satisfy the error equidistribution, each fine-mesh element is allowed an error of e_0/N^f, which means that each element k is allowed an error of $n_k e_0/N^f$. We relate the growth in elements to an error reduction factor through an a priori estimate

$$\underbrace{n_k \frac{e_0}{N^f}}_{\text{allowable error}} = \underbrace{\epsilon_k \left(\frac{h^f_{\text{ref}}}{h^c_{\text{ref}}} \right)^{\bar{p}_k+1}}_{\text{a priori estimate}} \tag{36}$$

where ϵ_k is the current error indicator, $\bar{p}_k = \min(p_k, \gamma_k)$, and γ_k is the lowest order of any singularity within element k. Substituting Eqn. (35) into Eqn. (36) yields a relation between n_k and N^f.

$$n_k \frac{e_0}{N^f} = \epsilon_k n_k^{-(\bar{p}_k+1)/d} \Rightarrow n_k^{1+(\bar{p}_k+1)/d} = \frac{\epsilon_k}{e_0/N^f} \tag{37}$$

Substituting Eqn. (37) into $N^f = \sum_k n_k$, we can solve for N^f. In practice, we use a fixed-growth refinement strategy instead of relying on the a priori error estimate, *i.e.*, assume $N^f = f^{\text{growth}} N^c$ at each adaptation iteration, n_k and h^f_{ref} are determined by Eqn. (37) and Eqn. (35) respectively. Adaptation stops when we meet the error tolerence, $\epsilon \equiv \sum_k \epsilon_k \leq e_0$. For high-order solutions, the first d principal axes of the $p+1$ directional derivatives are used to characterize the element size and a priori error estimate.

5.4. Mesh optimization through error sampling and synthesis (MOESS)

Hessian-based anisotropic mesh adaptation has been shown to successfully detect solution anisotropy in many applications. However, it relies on a scalar solution u, which should be carefully chosen to correlate to the chosen output of interest. Also, 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. In addition, the fixed-fraction adaptation strategy and stop criterion used in Hessian-based adaptation can cause over-refinement for a design evaluation with certain error requirements. Due to these deficiencies, extra degrees of freedom may need to be added when the design changes. This kind of cost allocation may accumulate during the optimization, resulting in inefficiencies in practical applications.

In order to maximize the approximation potential of a mesh with a given number of degrees of freedom, we consider a more sophisticated spatial adaptation method: unstructured mesh optimization through 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 modifications in this section.

5.4.1. Error convergence model

The mesh optimization algorithm requires a model for how the error changes as the metric changes. Suppose that a metric step matrix $\boldsymbol{\mathcal{S}}_e$ is imposed on an element Ω_e with current error indicator of ϵ_{e0}. Instead of using a typical priori model in Eqn. (36), a generalized error model taking account of the directional convergence property is given in [30],

$$\epsilon_e = \epsilon_{e0} \exp[\text{tr}(\boldsymbol{\mathcal{R}}_e \boldsymbol{\mathcal{S}}_e)] \quad \Rightarrow \quad \frac{\partial \epsilon_e}{\partial \boldsymbol{\mathcal{S}}_e} = \epsilon_e \boldsymbol{\mathcal{R}}_e \tag{38}$$

where $\boldsymbol{\mathcal{R}}_e$ is a symmetric error convergence rate tensor containing the directional convergence information, while the step matrix $\boldsymbol{\mathcal{S}}_e$ encodes both the size change and the stretching of the new element. The total error over the mesh is the sum of the elemental errors, $\epsilon = \sum_{e=1}^{N_e} \epsilon_e$. During the optimization we will want to keep ϵ small, and we will want to determine the optimal step matrices at each element, and hence the metric changes at mesh vertices, $\boldsymbol{\mathcal{S}}_v$. The rate

tensor, $\boldsymbol{\mathcal{R}}_e$, is determined separately for each element through a local error sampling procedure in which the element is refined in different configurations and the resulting changes to the error are estimated.

For a triangular element, we consider four refinement options, indexed by i, as shown in Figure 2. We would like to know how much the error would decrease under each refinement option. One expensive option is to refine the element with the proposed cut, re-solve the primal and fine-space adjoint problems globally, and re-compute the error estimate. Though accurate, this would be impractically expensive. Another option is to only solve the primal/adjoint problems on a subset of the original mesh: the current element and its neighbors. This approach taken in [30] is less accurate but still performs very well as globally-exact primal/adjoint states are not necessary to estimate the error rate tensor. In this work we further simplify the estimation by not solving additional problems, even on a local patch of elements. Instead, we use an element-local projection method [46] to approximate the fine-space adjoint in semi-refined spaces associated with each refinement option.

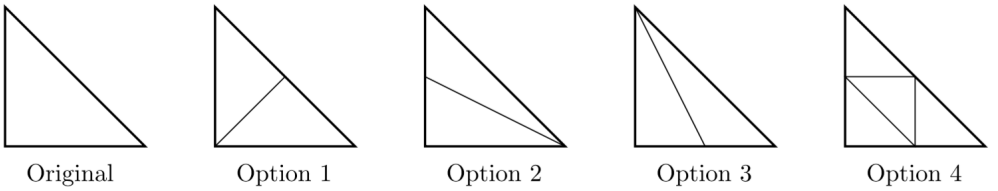


Fig. 2. Four refinement options for a triangle. Each one is considered implicitly during error sampling, though the elements are never actually refined.

5.4.2. Cost model

To measure the cost of refinement, we use degrees of freedom, dof, which on each element just depends on the approximation order p_e, assumed constant and equal to p over the elements. Again, we consider one element, Ω_e, with current cost C_{e0}, and a proposed metric step matrix $\boldsymbol{\mathcal{S}}_e$. The cost allocation for the new configuration is inversely proportional to the element area, which can be inferred from the metric,

$$C_e = C_{e0}\sqrt{\frac{\det(\boldsymbol{\mathcal{M}}_0^{1/2}\exp(\boldsymbol{\mathcal{S}}_e)\boldsymbol{\mathcal{M}}_0^{1/2})}{\det(\boldsymbol{\mathcal{M}}_0)}} = C_{e0}\exp\left[\frac{1}{2}\mathrm{tr}(\boldsymbol{\mathcal{S}}_e)\right] \quad \Rightarrow \quad \frac{\partial C_e}{\partial \boldsymbol{\mathcal{S}}_e} = C_e\frac{1}{2}\boldsymbol{\mathcal{I}} \tag{39}$$

where C_e is the expected cost over the original element area after applying $\boldsymbol{\mathcal{S}}_e$ to the original metric. The total cost over the mesh is the sum of the elemental costs, $C = \sum_{e=1}^{N_e} C_e$. During the optimization, we will want to determine the optimal step matrices given fixed cost to minimize the total error indicator ϵ.

5.4.3. Mesh optimization algorithm

Given a current mesh with its mesh-implied metric ($\boldsymbol{\mathcal{M}}_0(\vec{x})$), elemental error indicator ϵ_{e0}, and the elemental rate tensor $\boldsymbol{\mathcal{R}}_e$, the mesh optimization problem can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\mathcal{S}}_v} \quad & \epsilon(\boldsymbol{\mathcal{S}}_v) \\ \text{s.t.} \quad & C(\boldsymbol{\mathcal{S}}_v) = \text{const} \end{aligned} \tag{40}$$

where $\boldsymbol{\mathcal{S}}_v$ is the step matrix at each vertex, and the elemental step matrix $\boldsymbol{\mathcal{S}}_e$ in the error and cost models takes the algorithmic mean of its vertices' step matrices. Again, using the Lagrangian function, we have the first order optimality condition as

$$\frac{\partial \epsilon}{\partial \boldsymbol{\mathcal{S}}_v} + \lambda_s\frac{\partial C}{\partial \boldsymbol{\mathcal{S}}_v} = \mathbf{0} \tag{41}$$

where λ_s is the global Lagrange multiplier, taking the same value in every element. We do not solve this problem exactly, since this would be a very high-dimensional problem which may require extremely high computational effort, especially in an optimization problem where the optimal mesh changes as the design varies. Furthermore, the error model based on the empirical local sampling may not represent the error exactly. Therefore, solving the mesh

optimization problem exactly is inefficient and unnecessary. Instead, we follow Yano's optimization approach, by defining a "local" Lagrange multiplier,

$$\lambda_v = \frac{\partial \epsilon / \partial \boldsymbol{\mathcal{S}}_v}{\partial C / \partial \boldsymbol{\mathcal{S}}_v} \tag{42}$$

λ_v can be interpreted as the marginal improvement in the local error for a given investment in the local cost. The optimization problem is to equally distribute λ_v over the mesh vertices, which eventually converge to the same solution as Eqn. (41). In practice, the mesh optimization and state/adjoint solution are performed several times at a given target cost, until the error stops changing. Then the target cost is increased to reduce the error further if desired.

6. Optimization approach

6.1. Optimization algorithm

To be consistent with the analysis for the optimization formulation in Section 2, the optimization algorithm should involve the Lagrange multipliers. Sequential Least Square Programming (SLSQP) [47] with Broyden-Fletcher-Goldfarb-Shanno (BFGS) type Hessian approximation [48] is used in this work. The weak Wolfe condition is used to terminate the backtracking line search, ensuring a sufficient decrease at each optimization step. The gradients of the objective function and the constraints are calculated by the adjoint method, per Eqn. (18), and the objective and constraints are evaluated with the numerical solution of the state problem, Eqn. (19). The Lagrange multipliers associated with the trim constraints are extracted after each optimization step as a surrogate for the fine space multipliers, used in the error estimation for the objective in Eqn. (26).

6.2. Mesh adaptation in a multifidelity setting

With the requirements for high accuracy of the design, a fine discretization has to be chosen, so that the discrete optimization problem approximates the original continuous problem with required accuracy. 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 multifidelity 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. The objective function is first evaluated on a relatively coarse mesh, and then the error estimation and mesh adaptation are performed to control the numerical error to be below the optimization tolerance at current fidelity, the allowable numerical error decreases as the optimization fidelity increases. The mesh adaptation techniques used in this method is the Hessian-based anisotropic mesh adaptation. For the latter approach, several mesh levels (degrees of freedom) are defined before the optimization. Again, we start with a fairly coarse mesh, and then the mesh is optimized for each design to achieve the best accuracy. Once the objective change or the gradient norm 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. This approach is designed for MOESS to fulfill the potential of the cost at each fidelity. According to the prescribed information for each method, we refer them as error-based and cost-based methods respectively.

Compared with the fixed-fidelity optimization, unnecessarily fine meshes at the early stages of shape optimization are avoided in the two proposed multifidelity frameworks. Moreover, the areas that introduce most of the error may differ a lot for different designs during the optimization. Both approaches reduce the chance of over-refining areas that are not relatively important for the final design, which is important if the adaptation mechanics do not allow for coarsening. Compared with the multifidelity optimization without error estimation, the optimization tolerance and the error estimate are tightly coupled to actively control the optimization at each step and avoid the waste of low-fidelity convergence. Therefore, we expect the two methods can effectively prevent over-optimizing on a coarse mesh, or over-refining on an unintended shape.

6.3. Consistent objective-sensitivity analysis

One should note that even at the same fidelity, the mesh is not necessarily fixed as in the traditional design method. Rather, the mesh is also adapted if needed, e.g. refined in the error-based method or optimized in the cost-based method to control the numerical error. Recall the discretized optimization problem in Eqn. (11), omitting the

additional trim constraints for simplicity,

$$\begin{aligned} \min_{\mathbf{x}} \quad & J_h(\mathbf{U}_h(\mathbf{x}), \mathbf{x}) \\ \text{s.t.} \quad & \mathbf{R}_h(\mathbf{U}_h(\mathbf{x}), \mathbf{x}) = \mathbf{0} \end{aligned} \tag{43}$$

The above problem formulation infers a dependence on the discretization h, *i.e.* the computational mesh. One could prove that by refining the discretization, the discretized optimization problem converges to the continuous one. However, it should be mentioned that, the discretized optimization is an independent problem, which is characterized by the behavior of both the continuous problem and the discretization induced error.

$$J_h(\mathbf{u}_h, \mathbf{x}) = \mathcal{J}(\mathbf{u}, \mathbf{x}) + \mathcal{E}_h(\mathbf{u} - \mathbf{u}_h, \mathbf{x}) \quad \Rightarrow \quad \frac{dJ_h}{d\mathbf{x}} = \frac{d\mathcal{J}}{d\mathbf{x}} + \frac{d\mathcal{E}_h}{d\mathbf{x}} \tag{44}$$

In general, the discretized objective is incorrect compared to the exact continuous objective due to the numerical error. Consequently, the discretized objective gradient also involves the gradient of the error (assuming the error is a continuous function with respect to the design parameters), which is unique to each discretization. If the mesh is not adapted, then the numerical error may lead to a substantial deviation of the objective. However, the numerical error does not affect the convergence of the discretized optimization problem, since the discretized gradient analysis is consistent with the discretized objective function. In fact, if exact differentiation is used in Eqn. (44), one determines the exact gradient of the discretized objective functional. Due to the consideration above, controlling the error of the sensitivity seems to be of limited importance for the optimization process. However, objective sensitivity is commonly used in the stopping criterion for the optimization problem, controlling the sensitivity error can ensure the optimality condition [28].

In the proposed frameworks, inconsistent objective-sensitivity analysis may occur since we are changing the discretized optimization problem every time we adapt the mesh. Although all of the problems are approximations to the continuous optimization problem, each of them has their own behavior because of the embedded numerical error. There may not exist a feasible gradient-based update path between two design and discretization pairs, since the gradient depends on the discretization and hence cannot guide the update between different discretizations. For the sake of consistent objective-gradient analysis, we update the design on the same mesh, and then perform mesh adaptation if needed, yielding a sequence of designs and discretizations, which converges to the optimal design of the continuous optimization problem,

$$\{J_h\} = \{J_{h_0}(\mathbf{x}_0), J_{h_0}(\mathbf{x}_1), J_{h_1}(\mathbf{x}_1), ..., J_{h_{N-1}}(\mathbf{x}_N), J_{h_N}(\mathbf{x}_N)\} \qquad \lim_{h_N \to 0} |J_{h_N}(\mathbf{x}_N) - \mathcal{J}(\mathbf{x}^*)| = 0 \tag{45}$$

6.4. Algorithm overview

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 or cost levels are specified by the user, driving the mesh adaptation to actively control the numerical errors. In this paper, we assume the error estimation is accurate enough 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 be always below the optimization tolerance; $\eta = 1$ is adopted in this paper.

7. Results

In this section, we first demonstrate the importance of controlling the numerical error in optimization with a problem constrained by one-dimensional advection diffusion equations. Then two-dimensional airfoil shape optimization problems constrained by compressible Navier-Stokes equations are considered. The two methods proposed in Section Section 6 are applied in the airfoil optimization to compare their effectiveness and efficiency.

7.1. Scalar advection-diffusion

We now take a closer look at the example shown in Figure 1(b), the system is governed by the one-dimensional scalar advection-diffusion equation,

$$\begin{aligned} a\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x} = 0 \qquad & x \in [0, L] \\ u(0) = 0 \qquad u(L) = 1 \end{aligned} \tag{46}$$

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

input : initial deign $\mathbf{x}_0$, initial coarse mesh $\mathcal{T}_h$, optimization tolerance levels $\tau_0, \tau_1, ..., \tau_n$, safety factor $\eta \leq 1$
output: optimal design $\mathbf{x}^*$ with controlled objective error $\mathcal{E}(J_h) \leq \tau_n$

```
for l = 0, 1, ..., n do
    set error tolerance as E_l ← η τ_l
    while not converge do                                   ▷ optimization algorithm
        compute objective function J_h(x_l) and its error estimate δJ_h(x_l)
        while δJ_h > E_l do
            adapt mesh T_h
        end
        calculate objective gradient dJ_h/dx and update design x_l   ▷ line search
    end
    finish optimization at level l, x_{l+1} = x_l
end
```

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

input : initial deign $\mathbf{x}_0$, initial coarse mesh $\mathcal{T}_h$, cost levels $C_0, C_1, ..., C_n$, safety factor $\eta \geq 1$
output: optimal design $\mathbf{x}^*$ with optimal accuracy at given cost C_n

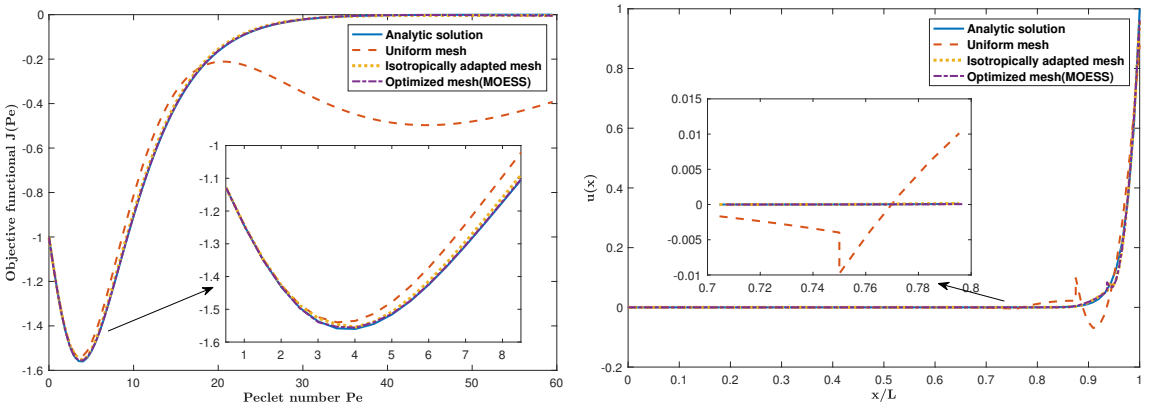
```
for l = 0, 1, ..., n do
    while not converge do                                   ▷ optimization algorithm
        optimize mesh T_h to minimize numerical error at fixed cost C_l
        calculate objective J_h(x_l) and its error estimate δJ_h(x_l)
        set optimization tolerance τ_l = η δJ_h
        calculate objective gradient dJ_h/dx and update design x_l   ▷ line search
    end
    finish optimization at level l, x_{l+1} = x_l
end
```

The optimization problem is formulated as seeking an optimal Peclet number defined as $Pe = aL/\nu$ to minimize the scalar gradient at a specified location,

$$\min_{\mathbf{x}=Pe} \quad \left.\frac{\partial u}{\partial x}\right|_{x=0.76L} \tag{47}$$

The objective functional $J = \frac{\partial u}{\partial x}|_{x=0.76L}$ is an implicit function of the design variable $\mathbf{x}$ (Peclet number) defined by the underlying state equation. We discretize the continuous optimization problem using DG with approximation order $p = 2$. Three computational meshes with the same degrees of freedom are tested: a uniformly-distributed mesh, an isotropically-refined mesh, and an optimized mesh with MOESS, both starting from a coarser mesh. The discretized objective functional on different meshes are shown in Figure 3(a). We see that the naive uniformly distributed mesh produces a spurious optimum besides the expected one due to the discretization error, while the adapted meshes are able to predict a reasonably accurate objective functional over the entire design space. Therefore, the optimization performed on the uniform mesh can heavily depend the starting point, especially for gradient-based methods. The optimizer may converge to the spurious local optimum if the descent direction is pointing to it. With the same degrees of freedom, the adapted meshes based on objective error are more robust to the starting point, since the objective shape over the design space is preserved by reducing the numerical error. However, the numerical error can hardly be eliminated: we can only converge to the optimum by some tolerance comparable to the objective error, which is always the case in most engineering applications. By zooming into the region near the exact optimum, we can see that the uniform mesh has the highest error leading to a most inaccurate optimum. The optimized mesh tends to give a better accuracy compared to the isotropically adapted mesh. In order to have a closer look at the spurious optimum, we plot the state solution $u(x)$ over the computational domain on different meshes in Figure 3(b). We can observe a severe numerical oscillation near the location of interest for the solution on the uniform mesh, while the adapted meshes are made to predict an accurate objective, which reduces the possibility of the occurrence of spurious optima.

The numerical oscillation shown in Figure 3(b) is only one of the possible sources that may cause spurious optima. Many physical features that are sensitive to numerical errors, such as boundary layers or shocks in flow problems, can also potentially create spurious optima.



(a) Discretized objective functional on different meshes

(b) State solutions u at the spurious optima

Fig. 3. Optimization constrained by 1D scalar advection-diffusion PDE on different meshes

A comparison of the optimization on these meshes starting from different Peclet numbers is listed in Table 1. We do not consider the error estimation and optimization coupling here since the error in the objective is fairly large on these coarse meshes, which can cause early convergence for the optimization. The optimization tolerance is set to be small enough compare to the numerical error. The optimization on the fixed uniform mesh converges to the spurious optimum when the initial point is close to it, whereas the adapted meshes produce more accurate results that are more robust to the starting point as expected. Furthermore, the optimization with optimized meshes produced by MOESS exhibits better accuracy since the optimized meshes tend to obtain a better approximation of the objective over the design space.

Table 1. Optimization results (advection-diffusion PDE) on different meshes (DG, $p = 2$, $N_e = 8$, *optimization tol* = 1e−10)

Initial design	Mesh	$\mathbf{x}_h^*$	$J_h(\mathbf{x}_h^*)$	$\|\mathbf{x}_h - \mathbf{x}^*\|$	$\|J_h(\mathbf{x}_h^*) - \mathcal{J}(\mathbf{x}^*)\|$
$\mathbf{x}_0 = 40$	Uniform mesh	44.6029	-0.4970	40.7969	1.0645
	Iso-adapted mesh	3.7215	-1.5472	0.0846	0.0142
	Optimized mesh	3.8100	-1.5622	0.0040	0.0007
$\mathbf{x}_0 = 20$	Uniform mesh	3.6458	-1.5404	0.1602	0.0211
	Iso-adapted mesh	3.8440	-1.5562	0.0380	0.0052
	Optimized mesh	3.8100	-1.5622	0.0040	0.0007

7.2. Airfoil shape optimization

The simple example of the one-dimensional advection-diffusion problem shows the importance of controlling the objective error in PDE-constrained optimization. However, it is more common in engineering applications to optimize the objective subject to some constraints, where the numerical error can affect both the objective and constraint outputs. The error in the constraints can often indirectly affect the calculation of the objective, hence the mesh should be adapted to predict both the objective and constraint outputs with appropriate accuracy. As a simple demonstration of the constrained optimization approach, we consider two dimensional airfoil shape optimization problems in different flow regimes, which impose different state PDE constraints to the optimization problem. The goal of the optimization is to seek an optimal airfoil shape and incidence angle to minimize the drag coefficient subject to fixed lift trim condition and the minimum volume constraint. We only consider the numerical errors in the drag and lift calculations, and the airfoil volume measurements are assumed to be exact.

The airfoil is parametrized with 10 Hicks-Henne basis functions [49], and cubic curved mesh elements are used to represent the boundary. At each line search iteration in the optimization, the objective function needs to be re-evaluated, which requires a flow solution on the updated geometry, and hence a new mesh must be obtained every time. Regeneration of a mesh, especially for a complex geometry or with high resolution, could be time-consuming and non-trivial. And also for the purpose of objective-gradient consistency, mesh regeneration which may change the mesh topology is not preferred here. Thus, an efficient way to update the computational mesh is needed, and in this work, we use radial basis function (RBF) interpolation [50, 51] to deform the mesh for small shape changes and the linear elasticity equations [52] for large deformations. We first show the effective way of controlling the error of the optimization problem with a simple smooth laminar flow problem. Then the two different proposed adaptation strategies are compared in a more challenging inviscid transonic flow problem. Finally, a more practical turbulent transonic case is investigated.

7.2.1. Laminar, subsonic airfoil

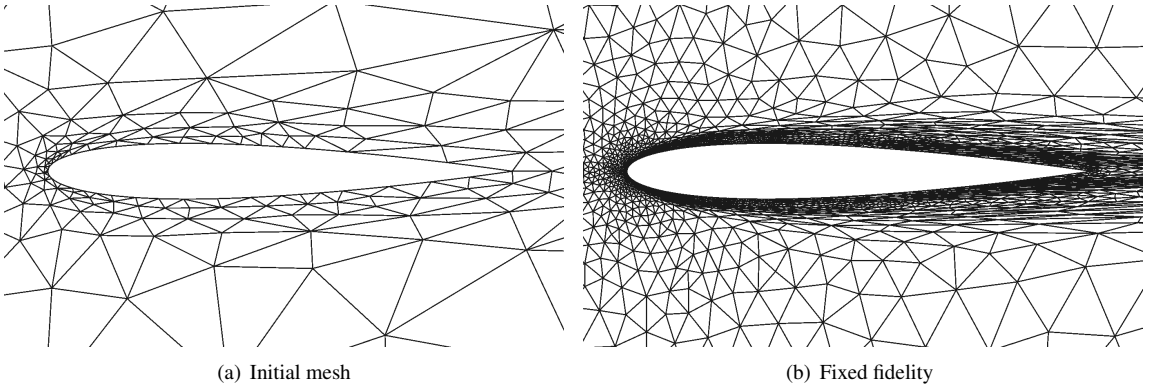
A subsonic laminar flow case is first studied. The initial design is a NACA 0012 airfoil at zero angle of attack with a free-stream Mach number $M_\infty = 0.5$ and Reynolds number $Re = 5000$. Three optimization and error controlling strategies are investigated: fixed (highest) fidelity optimization without error estimation and mesh adaptation, multifidelity optimization with error estimation and mesh adaptation only on the objective (only the error in the drag coefficient, c_d, is controlled during the optimization), and the multifidelity optimization with meshes adapted on both the objective and constraints as proposed in Section 4 (total error in the objective is restricted). We only consider the effects of different error control strategies, thus only the Hessian-based adaptation mechanics is used here. Different adaptation mechanics are compared in Section 7.2.2.

All of the optimization runs start from the same initial mesh consisting of 533 triangular elements with a DG $p = 1$ discretization. For the fixed-fidelity optimization, the mesh is first adapted to meet the objective tolerance, and then no more mesh adaptation occurs during the optimization. In contrast, in the other two methods, the mesh is adapted during the optimization, taking into account a changing error estimate. The initial symmetric airfoil should produce zero lift at zero incidence, and thus the initial condition is further from feasible if a higher target lift is specified, which means higher error of the outputs may appear during the optimization.

The target lift coefficient is set to be $c_l^* = 0.1$ here with a minimum volume as 95% of the initial NACA 0012 airfoil, and the final optimization tolerance is 1×10^{-4} in the drag coefficient. We should expect the final meshes to be comparable in size for all of the methods because of the same ultimate tolerance for the optimization. The meshes during the optimization, however, may be quite different. The fixed fidelity mesh and the meshes at the same intermediate optimization fidelity for the multifidelity methods are shown in Figure 4. These meshes show that multifidelity optimization significantly reduces the mesh size and computational resources during the early optimization iterations. The drag adjoint and lift adjoint (density component) are also shown with the meshes for the two multifidelity optimizations. Though the shape is not the same, the flow features and mesh sizes are similar. Most of the mesh adaptation happens at the leading edge since the drag adjoint is the largest (and least resolved) near the airfoil nose, while the proposed adaptation based on combined error estimation, Figure 4(d), adapts more on the trailing edge, which is also important for accurate lift prediction.

The objective and constraint convergence history are shown in Figure 5(a) and Figure 5(b). From the convergence plots, we see that the discretization error of the objective increases as the shape changes for the fixed-fidelity optimization without mesh adaptation. The objective error may be above the optimization tolerance during the optimization even though we start with a fairly fine mesh, and it converges to a noticeably different design compared to the results of the other two methods with error control, as shown in Figure 5(c). On the other hand, the objective error is always controlled to be below the optimization tolerance via mesh adaptation in the other two methods. However, the mesh adapted only on the objective has similar but slightly lower constraint error compared to the fixed mesh. The proposed method achieves lowest constraint error by taking it into account in the error estimation and adaptation, and of course requires higher computational cost at the highest fidelity as shown in Figure 5(d). Furthermore, the fixed-fidelity optimization requires the most iterations on the highest fidelity (the number of objective evaluations would be even more because of the line search between each major iteration). Multifidelity frameworks benefit from fewer iterations on the finest mesh since the they have better starting designs obtained from the coarser meshes.

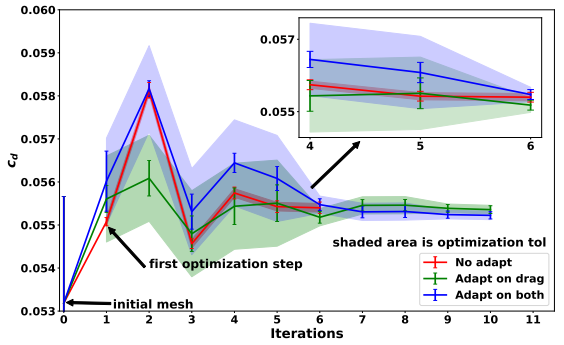
Local Mach number contours of the final designs on the corresponding meshes achieved by all these three methods are given in Figure 6. Again, we can see that the mesh adapts on both the objective and constraint has more refinement on the airfoil upper surface around the trailing edge, which is important for accurate lift calculation while it has little



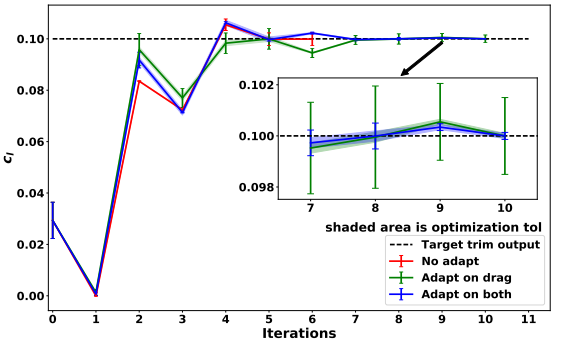
(a) Initial mesh (b) Fixed fidelity

(c) Adapt only on drag, 1371 elements (Drag adjoint, $-0.3 \sim 0.9$) (d) Adapt on both, 1302 elements (Lift adjoint, $-16 \sim 21$)

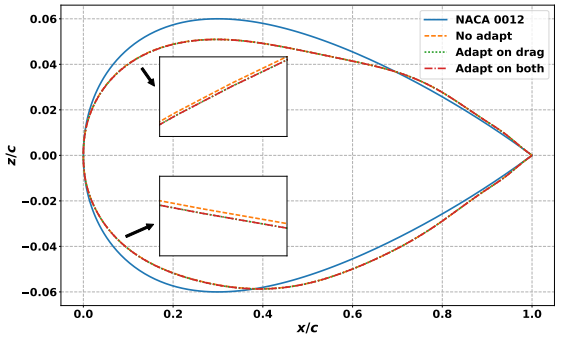
Fig. 4. Initial mesh and intermediate meshes for different error controlling strategies



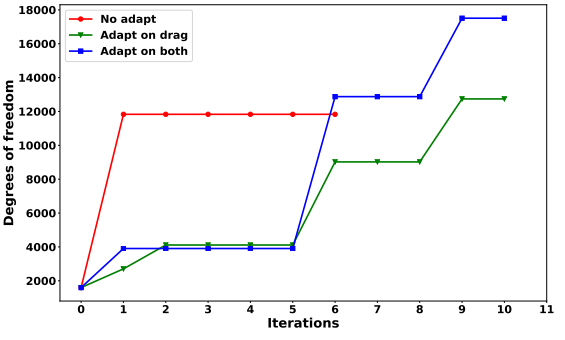
(a) Objective convergence history



(b) Constraint convergence history



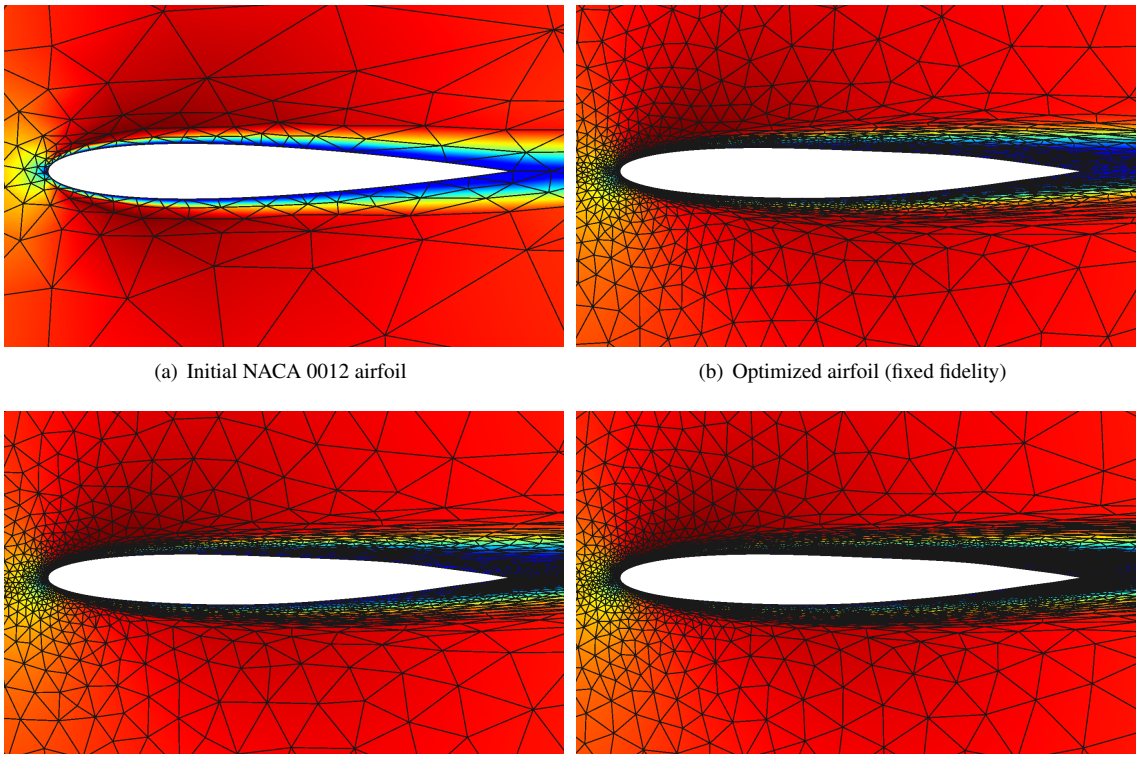
(c) Initial and optimized shapes



(d) Mesh size evolution

Fig. 5. Optimization history and final designs for different error controlling strategies (laminar, subsonic)

effect on the drag prediction. We take the optimized shape as well as the final mesh produced by all theses methods, then increase the approximation order to p+1, and start a new optimization (fixed mesh) to get the "exact" optimization solutions on these meshes. Since the meshes are not fine enough, we do not expect the "exact" solutions to be the same. The main results for the optimization on different meshes are summarized in Table 2. Using Eqn. (26), we decompose the error of the optimal objective into two parts: pure objective error (third column) and the error due to inaccurate trim constraint (fourth column). If the mesh does not adapt to predict accurate constraints, the error in the constraints can indirectly affect the objective, which is comparable to the pure objective error as shown in the table. In the proposed method, both error sources are well controlled, as the optimal objective error estimate is below the optimization tolerance. In this problem, we know that some refinements (leading edge) for accurate drag prediction also improve the lift accuracy, however, the important areas for objective and constraints can be very different in some problems [39], and adapting only on the objective can lead to undesired designs due to inaccurate constraints. By comparing the difference between order p and $p+1$ solutions, we also find the error estimate given by Eqn. (26) is pretty close to the "exact" error in this problem. However, the adjoint-based error estimates can be inaccurate on very coarse meshes or when the adjoints are not well resolved. In addition, the Lagrange multipliers extracted from the optimizer can be inaccurate when the objective and constraint gradients are not accurate or when the problem is highly nonlinear. We would not expect the error estimate in Eqn. (26) to be very accurate in these problems or situations, but it can still give the user an estimate of the error level which can be a guideline for the optimization (a safety factor η can be used to ensure the error to be sufficiently small compared to the optimization tolerance). The total optimization costs are also compared in Table 2 by scaling with the wall time used by the fixed fidelity optimization. Though the iterations on the finest meshes are reduced, we increase the total computational cost with error control and mesh adaptation due to additional adjoint and flow solves. Those extra costs required for more accurate design can possibly be reduced by more efficient adaptation techniques.



(a) Initial NACA 0012 airfoil

(b) Optimized airfoil (fixed fidelity)

(c) Optimized airfoil (adapt on drag)

(d) Optimized airfoil (adapt on both)

Fig. 6. Local Mach number contour $(0 \sim 0.6)$ **for the initial and final designs (laminar, subsonic)**

Table 2. Laminar airfoil optimization results on different meshes (DG, $p = 1$, $final\ optimization\ tol = 1e-4$)

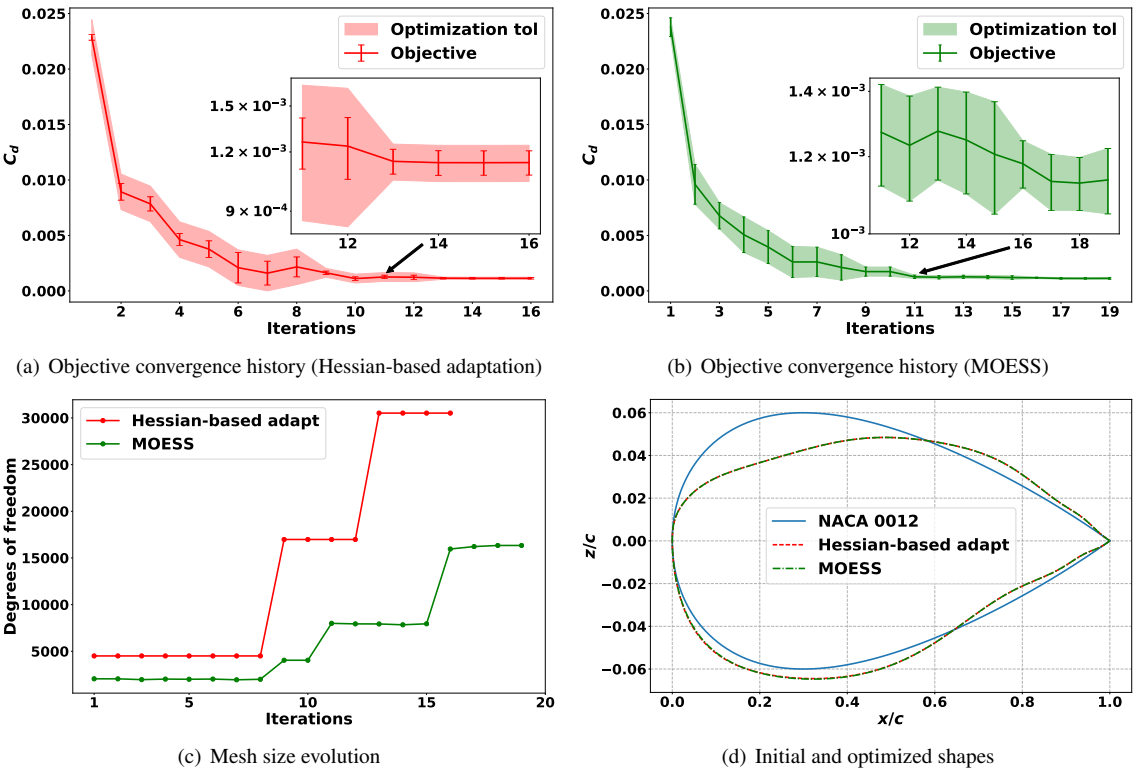
	$\left(J_{h,p}^{\text{adapt}}\right)^*$	δJ^{adapt}	$\boldsymbol{\mu}_h^T \delta \boldsymbol{J}^{\text{trim}}$	$\delta J_{\text{opt}}^{\text{adapt}}$	$\left(J_{h,p+1}^{\text{adapt}}\right)^*$	$\left\|\left(J_{h,p}^{\text{adapt}}\right)^* - \left(J_{h,p+1}^{\text{adapt}}\right)^*\right\|$	Cost
Fixed mesh	5.53993E-2	1.347E-4	1.456E-4	2.803E-4	5.51980E-2	2.013E-4	1.00
Adapt on drag	5.53572E-2	9.060E-5	8.490E-5	1.755E-4	5.51814E-2	1.758E-4	1.11
Adapt on both	5.52243E-2	8.402E-5	7.009E-6	9.103E-5	5.51327E-2	9.160E-5	2.56

7.2.2. Inviscid, transonic airfoil

We apply the new objective error estimation to an optimization problem based on an inviscid transonic flow at $M_\infty = 0.8$ around the NACA 0012 airfoil. The initial angle of attack is $\alpha = 1.25°$, and the goal is to minimize the drag with a target lift coefficient of $c_l^* = 0.4$. Again the minimum volume of the airfoil is set as 95% of the initial design. Although the transonic flow around the original NACA 0012 airfoil features a strong shock on the upper surface near the trailing edge, we expect that the shape would be modified during the optimization such that the shock strength is weakened or the shock is completely removed. The flow features and the outputs of interest are highly related to the location and strength of the shock, which may change a lot during the optimization. Thus, error estimation and mesh adaptation become more crucial in this case. In this problem, the mesh adaptation is based on the objective error estimate including the constraint error effects. Meanwhile, shocks are one of the very common anisotropic features in flow problems, since the flow field changes a lot in the direction normal to the shock while remains similar along the shock direction. Therefore, two different anisotropic adaptation mechanics are tested here, the Hessian-based mesh adaptation and mesh optimization via error sampling and synthesis (MOESS). Due to the different adaptation mechanisms, the couplings between optimization and mesh adaptation are also different. The former uses an error-based multifidelity optimization framework, while the latter adopts a cost-based one, as described in Algorithm 1 and Algorithm 2, respectively.

We start with the same initial mesh used in the laminar case, the objective convergence histories for these two adaptation mechanics are shown in Figure 7(a) and Figure 7(b). The objective on the initial mesh is not shown in the plots for simplicity. Since the error estimates have already included the effect of the constraints, we do not show the constraints convergence plots here. The optimization with Hessian-based mesh adaptation is performed with user-specified multiple error levels with an ultimate tolerance of 1×10^{-4} for the objective. On the other hand, the optimization using MOESS starts at a fairly low cost level, and degrees of freedom are added once the optimization converges at current cost level. The optimization stops when the total objective error (setting to be the optimization tolerance) is below 1×10^{-4}. As shown in the convergence plots, these two methods have similar convergence for this optimization problem. Most of the drag reduction happens at the lowest fidelity, where the flow solve is very cheap, though quite inaccurate. Both methods converge fast at the highest fidelity by virtue of better starting shapes obtained from the lower fidelity. We obtain fairly close optimal objective values as shown in Table 3. Compared with the initial objective value $c_{d,0}$ (the initial drag coefficient in the table is obtained by a very fine mesh on the original NACA 0012 airfoil, not the first point in the convergence plot), the total drag is therefore reduced by around 95% for both methods. The optimized shapes on these two meshes are also very similar as shown in Figure 7(d). The final designs approach a flattened upper surface and a higher aft camber, resembling a super-critical airfoil. The mesh size evolution is plotted in Figure 7(c). The mesh size (measured by dof) of the MOESS approach is much less than the Hessian-based mesh adaptation, which will be discussed in detail later.

Mach number contours for the initial and final designs on the corresponding meshes are shown in Figure 8. The initial strong shock is significantly weakened on the optimized airfoils. We can still see weak discontinuities in the flow field, though, and the drag could be further reduced if more basis functions were used to parametrize the airfoil shape and even higher fidelity flow calculations were performed. One can also notice that for the final designs in Figure 8, the Hessian-based adaptation requires a mesh which is much finer compared to the mesh optimized by MOESS. In fact, the optimized mesh only needs around half of the degrees of freedom required for the Hessian-based method, as shown in Figure 7(c). The total optimization cost with MOESS is thus also around half of the cost needed for Hessian-based adaptation as shown in Table 3. There are two main reasons making the optimization with MOESS more efficient, one is the adaptation mechanics and the other is the optimization algorithm. The Hessian-based mesh adaptation adapts where the scalar solution (Mach number here) anisotropy is large, which may not always be necessary for accurate objective and constraint prediction, while MOESS has a better anisotropy detection of the



(a) Objective convergence history (Hessian-based adaptation)

(b) Objective convergence history (MOESS)

(c) Mesh size evolution

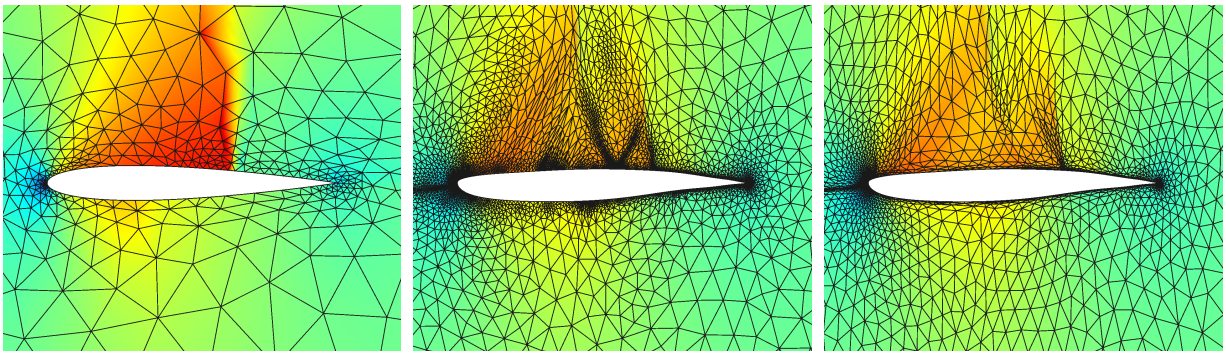
(d) Initial and optimized shapes

Fig. 7. Optimization history and final designs for different adaptation mechanics (inviscid, transonic)

output-relevant anisotropy over the computational domain through local error sampling. Hence we observe many refinements on the airfoil surface in Figure 8(b) which cannot be seen in Figure 8(c). With regard to optimization algorithm, error-based multifidelity optimization can be inefficient since the mesh gets refined every time the error is higher than the optimization tolerance, as the shape changes the mesh may get adapted in many areas that are important for different shapes, accumulating over-refinement during the optimization. On the other hand, the mesh is not necessary to be adapted as long as the error estimates are still below the tolerance even when the shape and flow field change. This effect can be found in the optimization here, some intermediate optimization steps of these two different methods are shown in Figure 9. In the 9^{th} optimization step with Hessian-based mesh adaptation, we can see there is a refinement aligned to the weak shock near the trailing edge in Figure 9(a). Nevertheless, this refinement stays there at following optimization steps even though the weak shock has moved forward since the error estimates are still below the optimization tolerance. Thus we lose the approximation capacity of these degrees of freedom. On the contrary, for the shape optimization combined with mesh optimization, we have refinements that always align to the weak shock on the top surface, and we can also observe that more anisotropic elements are added to the airfoil surface to improve the outputs accuracy, with the total degrees of freedom remaining the same. Therefore, for optimization with dramatic design changes or with complex systems involving strong anisotropy, MOESS will be more efficient and effective.

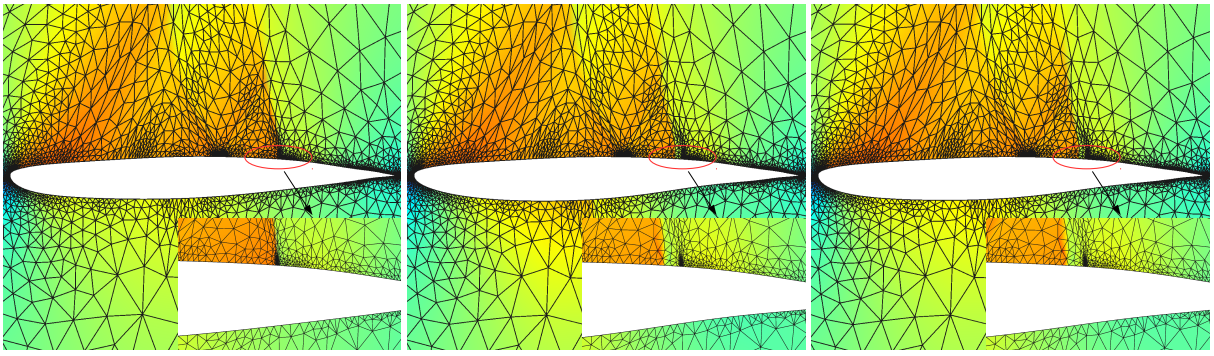
Table 3. Inviscid transonic airfoil optimization results on different meshes (DG, $p = 1$)

	$c_{d,0}$	$c_{d,\text{opt}}$	$\delta c_{d,\text{opt}}$	$\|c_{d,\text{opt}} - c_{d,0}\|/c_{d,0}$	Cost
Hessian-based	2.242E − 2	1.140E-3	6.708E-5	94.92% ± 0.30%	1.00
MOESS		1.136E-3	8.752E-5	94.93% ± 0.39%	0.53

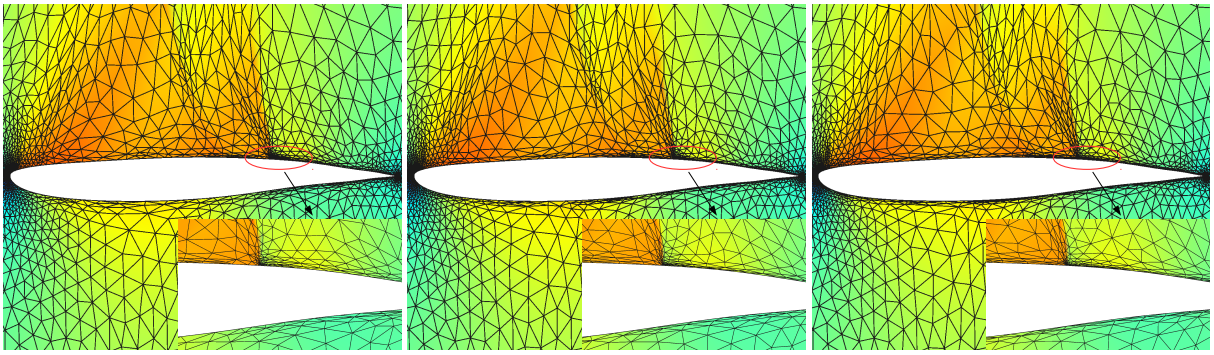


(a) NACA 0012 airfoil (b) Final design (Hessian-based adaptation) (c) Final design (MOESS)

Fig. 8. Local Mach number $(0 \sim 1.6)$ **for the initial and final designs (inviscid, transonic)**



(a) Hessian-based adaptation, 9^{th}, 10^{th} and 12^{th} optimization step from left to right



(b) MOESS, 16^{th},17^{th} and 19^{th} optimization step from left to right

Fig. 9. Intermediate meshes for different adaptation mechanics (inviscid, transonic)

7.3. Turbulent, transonic high-lift airfoil

The final problem considered in this paper is a more sophisticated turbulent case: optimization of a transonic airfoil with a high lift demand, starting with the RAE 2822 airfoil. The initial angle of attack is 2.79° with a freestream Mach number of $M_\infty = 0.734$ and Reynolds number $Re = 6.5 \times 10^6$. The target lift coefficient is set to be $c_l^* = 0.824$, and the airfoil volume is restricted to be no smaller than the initial value. The starting mesh for the turbulent case consists of 1448 elements, which is finer than the initial mesh used for the laminar and transonic runs. An approximation order of $p = 2$ is used in this case.

The initial mesh and the meshes used in the optimization are summarized in Figure 10. For the turbulent case, most of the adaptation focuses on resolving the boundary layer, which is highly anisotropic. Therefore, many degrees of freedom are put into the boundary layer with anisotropic elements around the airfoil boundary. Once the boundary layer is properly resolved, the objective error drops down very quickly. Then, more degrees of freedom are added to the upper surface to resolve the flow field changes along the flow direction.

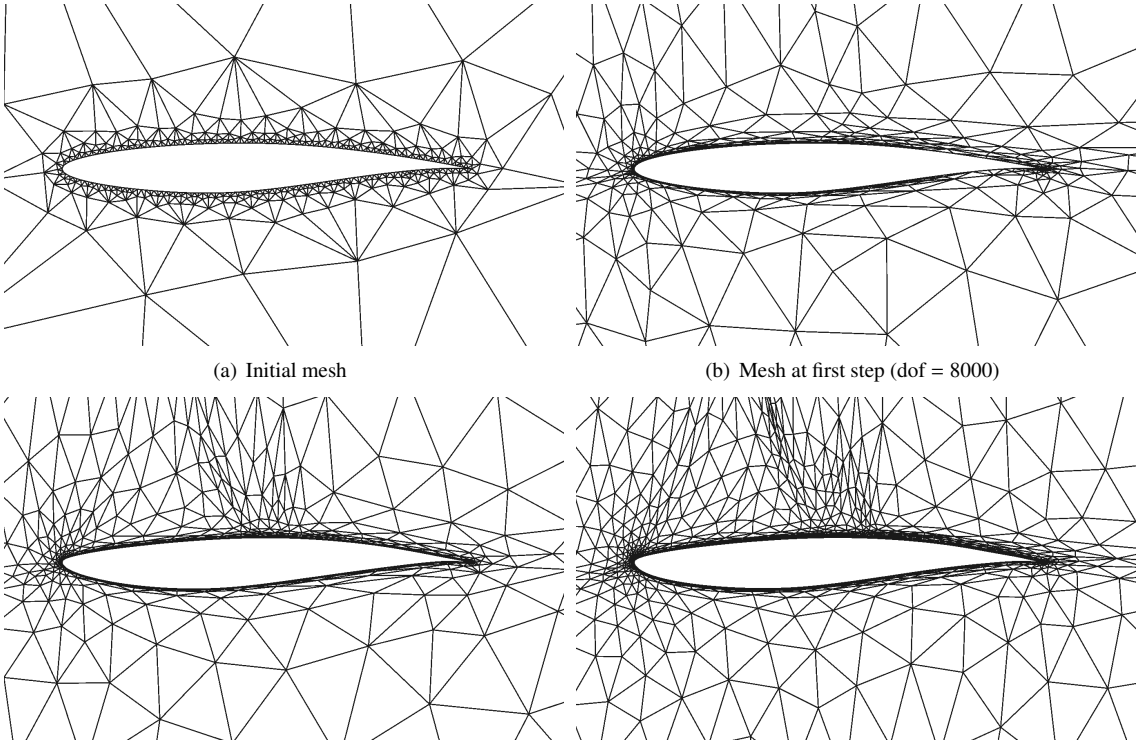
The mesh size and objective are collected at each optimization step as shown in Figure 11. Here we assume that at a given fixed cost, the objective error on optimized meshes should be close if the physics are similar. This can also be observed in Figure 11(a), however we do see some oscillations of the error at coarse and medium meshes, which are caused by the shape changes and hence the different physics (shock strength and location). At the highest fidelity we expect only small changes in the shape, and the objective error remains almost the same, which means that both the shape and the mesh converge to an optimum. The initial and optimized airfoils are compared in Figure 12. As shown in Figure 12(c), the optimization flattens the upper surface near the forward section, while curving the lower surface and increasing the thickness in the aft section. The curvature reduction on the top surface is trying to smooth the flow acceleration region to weaken the shock. The thickened lower surface and aft section are required to maintain the lift and area constraints. This can be further observed in the Mach contours shown in Figure 12(a) and Figure 12(b), where the initial strong shock is significantly reduced. Hence the strong discontinuity is absent in the airfoil surface pressure distribution in Figure 12(d). Therefore the final design yields a drag coefficient of $c_{d,opt} = 1.0506\text{E}{-2} \pm 2.8000\text{E}{-5}$, achieving a $43.73\% \pm 0.15\%$ drag reduction compared with the initial design which produces a drag of $c_{d,0} = 1.8671\text{E}{-2}$ (the initial drag coefficient is obtained with an optimized finer mesh on the original design).

8. Conclusion

In most PDE-constrained optimizations, we work with discretized governing equations. Thus, numerical error should be carefully controlled to ensure convergence to the "true" optimal design at a prescribed fidelity. Without properly controlling this error, the optimizer may arrive at a sub-optimal design or even at an incorrect spurious optimum with inaccurate information provided by the PDE solver and gradient analysis as shown in the test cases.

In this work, we presented frameworks that integrate output-based error estimation and mesh adaptation with a traditional gradient-based algorithm. A coupled adjoint is also introduced, offering a way to include the constraints error into the objective error estimation. The multifidelity optimization approach consists of progressive refinement of the computational mesh and is capable of preventing over-optimizing and over-refining. The mesh adaptation (fidelity increase) is tightly coupled with the optimization algorithm either with error-based or cost-based strategy. The optimization with mesh optimization via error sampling and synthesis (MOESS) is shown to be more efficient and effective by fulfilling the approximation potential of given cost. This benefit can become more significant when higher fidelity is required, or highly anisotropic physics governs the system.

With more judicious considerations of the objective functions and constraints, and additional design parameters, the new methods can provide realistic configurations in practical design scenarios. The fidelity increase is presently driven by an adaptation tolerance or a computational cost that increases or decreases by a fixed factor each time, or has to be specified by the user. However, for more practical problems, without a priori knowledge of the objective convergence, an improved and automated fidelity increase strategy should be developed to fulfill the potential of the present optimization framework to increase the accuracy and efficiency for PDE-constrained optimization problems. Furthermore, only the mesh adaptation ($h-$adaptation) is considered here to control the numerical error. More efficient adaptation mechanics like approximation order increment ($p-$adaptation), and combinations ($hp-$adaptation) can also be applied to current methods in the future.



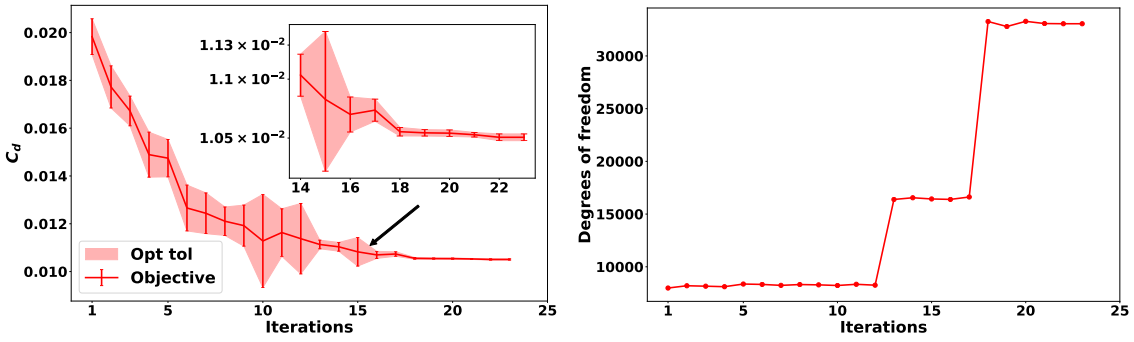
(a) Initial mesh

(b) Mesh at first step (dof = 8000)

(c) Mesh at 17^{th} step (dof = 16000)

(d) Final mesh (dof = 32000)

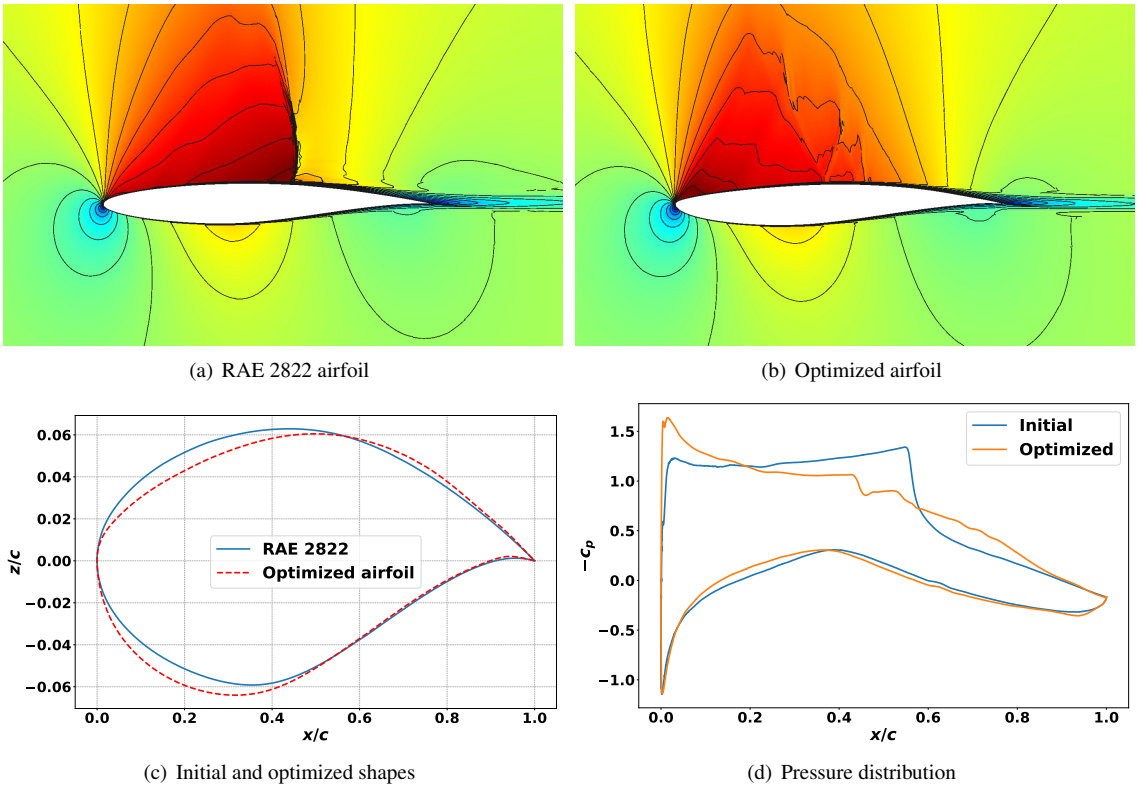
Fig. 10. Initial mesh and intermediate meshes during the optimization (turbulent, transonic)



(a) Objective convergence

(b) Mesh evolution

Fig. 11. Objective convergence history and mesh evolution (turbulent, transonic)



(a) RAE 2822 airfoil

(b) Optimized airfoil

(c) Initial and optimized shapes

(d) Pressure distribution

Fig. 12. Local Mach Number ($0 \sim 1.3$) **and pressure distribution for the initial and final designs (turbulent, transonic)**

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.

References

[1] P. Hajela, Nongradient methods in multidisciplinary design optimization-status and potential, Journal of aircraft 36 (1999) 255–265.
[2] J. R. Martins, J. J. Alonso, J. J. Reuther, A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design, Optimization and Engineering 6 (2005) 33–62.
[3] J. R. Martins, P. Sturdza, J. J. Alonso, The complex-step derivative approximation, ACM Transactions on Mathematical Software (TOMS) 29 (2003) 245–262.
[4] A. Griewank, A. Walther, Evaluating derivatives: principles and techniques of algorithmic differentiation, volume 105, Siam, 2008.
[5] A. Jameson, Aerodynamic design via control theory, Journal of scientific computing 3 (1988) 233–260.
[6] 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 (1999) 51–60.
[7] W. K. Anderson, V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, Computers & Fluids 28 (1999) 443–480.
[8] M. B. Giles, N. A. Pierce, An introduction to the adjoint approach to design, Flow, turbulence and combustion 65 (2000) 393–415.
[9] O. C. Zienkiewicz, J. Z. Zhu, A simple error estimator and adaptive procedure for practical engineerng analysis, International journal for numerical methods in engineering 24 (1987) 337–357.
[10] I. Babuška, A. Miller, A feedback finite element method with a posteriori error estimation: Part i. the finite element method and some basic properties of the a posteriori error estimator, Computer Methods in Applied Mechanics and Engineering 61 (1987) 1–40.
[11] O. Zienkiewicz, J. Zhu, Adaptivity and mesh generation, International Journal for Numerical Methods in Engineering 32 (1991) 783–810.
[12] R. Mueller, D. Gross, G. Maugin, Use of material forces in adaptive finite element methods, Computational Mechanics 33 (2004) 421–434.
[13] N. Kikuchi, K. Y. Chung, T. Torigaki, J. E. Taylor, Adaptive finite element methods for shape optimization of linearly elastic structures, in: The Optimum Shape, Springer, 1986, pp. 139–169.
[14] N. Banichuk, F. Barthold, A. Falk, E. Stein, Mesh refinement for shape optimization, Structural optimization 9 (1995) 46–51.
[15] A. Schleupen, K. Maute, E. Ramm, Adaptive fe-procedures in shape optimization, Structural and Multidisciplinary Optimization 19 (2000) 282–302.

[16] N. A. Pierce, M. B. Giles, Adjoint recovery of superconvergent functionals from pde approximations, SIAM review 42 (2000) 247–264.
[17] R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, Acta numerica 10 (2001) 1–102.
[18] R. Hartmann, P. Houston, Adaptive discontinuous galerkin finite element methods for the compressible euler equations, Journal of Computational Physics 183 (2002) 508–532.
[19] D. A. Venditti, D. L. Darmofal, Grid adaptation for functional outputs: application to two-dimensional inviscid flows, Journal of Computational Physics 176 (2002) 40–69.
[20] M. A. Park, Adjoint-based, three-dimensional error prediction and grid adaptation, AIAA journal 42 (2004) 1854–1862.
[21] K. Mani, D. J. Mavriplis, Error estimation and adaptation for functional outputs in time-dependent flow problems, Journal of Computational Physics 229 (2010) 415–440.
[22] K. J. Fidkowski, Y. Luo, Output-based spacetime mesh adaptation for the compressible navierstokes equations, Journal of Computational Physics 230 (2011) 5753 – 5773.
[23] K. J. Fidkowski, D. L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, AIAA journal 49 (2011) 673–694.
[24] 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.
[25] 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, 2005.
[26] 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, 2013, p. 865.
[27] D. Li, R. Hartmann, Adjoint-based airfoil optimization with discretization error control, International Journal for Numerical Methods in Fluids 77 (2015) 1–17.
[28] J. E. Hicken, J. J. Alonso, Pde-constrained optimization with error estimation and control, Journal of Computational Physics 263 (2014) 136–150.
[29] G. Chen, K. Fidkowski, Airfoil shape optimization using output-based adapted meshes, in: 23rd AIAA Computational Fluid Dynamics Conference, 2017, p. 3102.
[30] M. Yano, D. L. Darmofal, An optimization-based framework for anisotropic simplex mesh adaptation, Journal of Computational Physics 231 (2012) 7626–7649.
[31] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.
[32] G. Biros, O. Ghattas, Parallel lagrange–newton–krylov–schur methods for pde-constrained optimization. part i: The krylov–schur solver, SIAM Journal on Scientific Computing 27 (2005) 687–713.
[33] S. R. Allmaras, F. T. Johnson, Modifications and clarifications for the implementation of the spalart-allmaras turbulence model, in: Seventh international conference on computational fluid dynamics (ICCFD7), 2012, pp. 1–11.
[34] C. E. Baumann, J. T. Oden, A discontinuous hp finite element method for convection-diffusion problems, Computer Methods in Applied Mechanics and Engineering 175 (1999) 311 – 341.
[35] P. Houston, E. Süli, hp-adaptive discontinuous galerkin finite element methods for first-order hyperbolic problems, SIAM Journal on Scientific Computing 23 (2001) 1226–1252.
[36] L. Wang, D. J. Mavriplis, Adjoint-based h-p adaptive discontinuous galerkin methods for the 2d compressible euler equations, Journal of Computational Physics 228 (2009) 7643 – 7661.
[37] P. L. Roe, Approximate riemann solvers, parameter vectors, and difference schemes, Journal of computational physics 43 (1981) 357–372.
[38] F. Bassi, S. Rebay, Gmres discontinuous galerkin solution of the compressible navier-stokes equations, in: Discontinuous Galerkin Methods, Springer, 2000, pp. 197–208.
[39] 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, 2014, p. 2576.
[40] A. Loseille, F. Alauzet, Continuous mesh model and well-posed continuous interpolation error estimation, Ph.D. thesis, INRIA, 2009.
[41] F. Hecht, Bamg: bidimensional anisotropic mesh generator, User Guide. INRIA, Rocquencourt (1998).
[42] X. Pennec, P. Fillard, N. Ayache, A riemannian framework for tensor computing, International Journal of computer vision 66 (2006) 41–66.
[43] M. Castro-Diaz, F. Hecht, B. Mohammadi, O. Pironneau, Anisotropic unstructured mesh adaption for flow simulations, International Journal for Numerical Methods in Fluids 25 (1997) 475–491.
[44] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows, Journal of Computational Physics 187 (2003) 22–46.
[45] 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 (2007) 1653–1672.
[46] K. Fidkowski, A local sampling approach to anisotropic metric-based mesh optimization, in: 54th AIAA Aerospace Sciences Meeting, 2016, p. 0835.
[47] 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).
[48] C. G. Broyden, A class of methods for solving nonlinear simultaneous equations, Mathematics of computation 19 (1965) 577–593.
[49] R. M. Hicks, P. A. Henne, Wing design by numerical optimization, Journal of Aircraft 15 (1978) 407–412.
[50] S. Jakobsson, O. Amoignon, Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization, Computers & Fluids 36 (2007) 1119 – 1136.
[51] T. Rendall, C. Allen, Efficient mesh motion using radial basis functions with data reduction algorithms, Journal of Computational Physics 228 (2009) 6231 – 6249.
[52] J. E. Hicken, D. W. Zingg, Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement, AIAA journal 48 (2010) 400–413.