High-Order Node Movement Discretization Error Control in Shape Optimization

Devina P. Sanjaya *
University of Tennessee, Knoxville, TN, 37996

Krzysztof J. Fidkowski†
University of Michigan, Ann Arbor, MI, 48188

We present mesh node movement as an effective discretization error control strategy for shape optimization problems on structured meshes. The context is aerodynamic shape optimization via mesh deformation on anisotropic, structured meshes suitable for Reynolds-averaged Navier-Stokes (RANS) simulations. Movement of interior nodes occurs concurrently with the shape optimization iterations, with the objective of minimizing the output error for a fixed mesh size. The output-based error estimates that drive the node movement rely on an adjoint solution that is also used by the gradient-based shape optimization. In the context of high-order solution approximation on curved meshes, the high-order mesh nodes provide additional degrees of freedom for controlling the discretization error. Even though the meshes of interest are structured, a metric-based mesh optimization algorithm is used in an intermediate step to provide element sizing and anisotropy information. The objective of the node movement is then to make the mesh better conform to the computed metric field. Results from two-dimensional aerodynamics shape optimization problems demonstrate the ability of the node movement to decrease discretization error, and the efficacy of the output error estimates.

I. Introduction

The increased demand for more accurate and robust computational fluid dynamics (CFD) simulations has largely motivated research in high-order CFD methods [1] and output-based mesh adaptation [2–4]. The growing interests in high-order CFD methods and their applications have recently led us to research in high-order mesh adaptation and optimization [4–6]. And thanks to the rapid growth of computing power, CFD is now widely recognized as an essential tool for engineering design and optimization [7, 8].

Shape optimization is an important problem in the aerospace engineering community. One approach to shape optimization is to formulate the problem as a trimmed optimization problem in which the parameter vector consists of distinct shape design parameters and trim parameters, and this is the approach taken in this work. Previous works in shape optimization emphasize the importance of controlling the discretization errors, as these errors can strongly affect the reliability of the optimization objective and the sensitivity calculations, which in the end, affect the overall quality of the optimization results [9–11]. Our focus here is on controlling the discretization errors on structured, high-order meshes with fixed element counts. The error control is done solely through mesh node movement, which is formulated as a metric-based mesh optimization driven by output error estimates. To control the discretization errors effectively, the shape and mesh node optimizations are done concurrently. Two optimization frameworks are proposed in order to offer different workflow and cost allocation for mesh node movement. While these frameworks are developed for general high-order, finite-element methods and governing equations, we present our results specifically for a high-order, discontinuous-Galerkin (DG) discretization of the Reynolds-averaged Navier-Stokes (RANS) equations. Finally, we note that hanging node adaptation is not performed in this work.

The reminder of this paper is outlined as follows. In Section II, we give a brief review on our chosen discretization, the DG method. In Section III, we describe the general shape optimization problem. Details of the error estimation, metric calculation, and mesh node movement are given in Sections IV and V. Section VI presents the proposed optimization frameworks. Preliminary results are shown in Section VII, and Section VIII concludes the present work and discusses our plans for the final paper.

*Assistant Professor, Mechanical, Aerospace & Biomedical Engineering Department, AIAA Young Professional Member.
†Professor, Department of Aerospace Engineering, AIAA Associate Fellow.

II. Discretization

A. Primal

We consider discretizations of differential equations in conservation form,

$$\nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{S}(\mathbf{u}) = \mathbf{0}, \tag{1}$$

where $\mathbf{u}(\vec{x}, t) \in \mathbb{R}^s$ is the conservative state vector containing mass, momentum, and energy per unit volume, s is the state rank, $\vec{\mathbf{H}}(\mathbf{u}, \nabla \mathbf{u}) = \vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u})$ is the total (inviscid and viscous) flux, and $\mathbf{S}(\mathbf{u})$ is a source term active when modeling turbulence. In this work, the Reynolds-averaged Navier-Stokes (RANS) equations with the Spalart-Allmaras (SA) closure [12, 13] are of interest.

To discretize Eqn. 1, we use a finite-element method, the discontinuous Galerkin (DG) discretization [14]. The computational domain Ω is divided into N_e elements, Ω_e, in a non-overlapping tessellation T_h. Inside element Ω_e, the state components are approximated by polynomials of order p, with no continuity constraints between elements. Formally, we write: $\mathbf{u}_h \in \mathcal{V}_h = [\mathcal{V}_h]^s$, where $\mathcal{V}_h = \left\{ u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \ \forall \Omega_e \in T_h \right\}$, and $\mathcal{P}^p$ denotes polynomials of order p on the reference space of element Ω_e.

The weak form of Eqn. 1 is obtained by multiplying the equation by test functions in the same approximation space, integrating by parts, and coupling elements via single-valued fluxes that are functions of the states on the two adjacent elements. We use the Roe-approximate Riemann solver [15] for the convective flux, and the second form of Bassi and Rebay (BR2) [16] for the viscous flux. Choosing a basis for the test and trial spaces yields a system of nonlinear equations,

$$\mathbf{R}(\mathbf{U}, \mathbf{x}) = \mathbf{0}, \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^N$ is the discrete state vector, $\mathbf{R}(\mathbf{U}) \in \mathbb{R}^N$ is the spatial residual vector, N is the total number of spatial degrees of freedom, and $\mathbf{x} \in \mathbb{R}^{n_{\text{par}}}$ is a vector of parameters that in this work consists of trim and shape design variables.

B. Adjoint

For a scalar output computed from the state vector, $J(\mathbf{U})$, the discrete steady adjoint vector, $\mathbf{\Psi} \in \mathbb{R}^N$, is the local sensitivity of J to perturbations in the steady residual, $\mathbf{R}$ [2]. Linearizing the residual and output yields the following linear adjoint equation,

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)^T \mathbf{\Psi} + \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T = \mathbf{0}. \tag{3}$$

This equation is solved using the same matrix solver used for the Newton-Raphson primal system. The adjoint vector can be used to efficiently compute PDE-constrained sensitivities of the output with respect to parameters, $\mathbf{x}$, of the problem. In general, if both the residuals and the output depend on $\mathbf{x}$, the PDE-constrained sensitivity of the output with respect to the parameter vector is

$$\frac{dJ}{d\mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \mathbf{\Psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}}. \tag{4}$$

In this work, derivatives of J and $\mathbf{R}$ with respect to $\mathbf{x}$ are evaluated using finite-differences. This poses no computational bottleneck in the present study with a small number of parameters. For larger numbers of parameters, analytical or algorithmic differentiation methods can be applied to increase the efficiency of this calculation.

III. Shape Optimization

We assume trimmed optimization problems in which the parameter vector consists of distinct shape design parameters, $\mathbf{x}^{\text{des}} \in \mathbb{R}^{n_{\text{des}}}$, and trim parameters, $\mathbf{x}^{\text{trim}} \in \mathbb{R}^{n_{\text{trim}}}$, concatenated into $\mathbf{x} = [\mathbf{x}^{\text{des}}; \mathbf{x}^{\text{trim}}] \in \mathbb{R}^{n_{\text{par}}}$. Formulated as a minimization statement for a scalar output J, the problem reads

$$\begin{aligned} \min_{\mathbf{x}^{\text{des}}} \quad & J(\mathbf{U}, \mathbf{x}) \\ \text{s.t.} \quad & \mathbf{R}(\mathbf{U}, \mathbf{x}) = \mathbf{0} \\ & \mathbf{R}^{\text{trim}} \equiv \mathbf{J}^{\text{trim}}(\mathbf{U}, \mathbf{x}) - \bar{\mathbf{J}}^{\text{trim}} = \mathbf{0}, \end{aligned} \tag{5}$$

where $\mathbf{J}^{\text{trim}} \in \mathbb{R}^{n_{\text{trim}}}$ is the vector of trim outputs, and $\bar{\mathbf{J}}^{\text{trim}} \in \mathbb{R}^{n_{\text{trim}}}$ is the vector of target trim values. In this work, we consider $n_{\text{trim}} = 1$: the lift coefficient, c_ℓ, trimmed by angle of attack, α.

A. Shape Optimization with Concurrent Trimming

To optimize the design, we use a gradient-based method: the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [17] algorithm with a backtracking line search. The implementation features concurrent trimming and optimization with a constrained gradient calculation. A given initial design, $\mathbf{x}_0^{\text{des}}$, is first trimmed using multiple solver iterations, where at each iteration, the trim parameters are updated using the the PDE-constrained sensitivity matrix $\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}^{\text{trim}}} \in \mathbb{R}^{n_{\text{trim}} \times n_{\text{trim}}}$, computed using adjoints of the trim outputs, and a user-specified trim tolerance.

Shape optimization follows the initial trimming. Each optimization iteration consists of solutions for the state $\mathbf{U}_k$, and adjoints of the objective and trim outputs. The adjoints yield PDE-constrained sensitivities with respect to the design and trim parameters. The trimmed gradient of the objective with respect to the design parameters is then

$$\left.\frac{dJ}{d\mathbf{x}^{\text{des}}}\right|_{\text{trim}} = \frac{dJ}{d\mathbf{x}^{\text{des}}} + \frac{dJ}{d\mathbf{x}^{\text{trim}}}\frac{d\mathbf{x}^{\text{trim}}}{d\mathbf{x}^{\text{des}}}, \qquad \frac{d\mathbf{x}^{\text{trim}}}{d\mathbf{x}^{\text{des}}} = -\left[\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}^{\text{trim}}}\right]^{-1}\frac{d\mathbf{J}^{\text{trim}}}{d\mathbf{x}^{\text{des}}}, \tag{6}$$

The gradient of the objective output as given in Eqn. 6 is then used in the BFGS algorithm.

B. Line Search

During the line search, an objective function evaluation accounts for a linearized trim parameter change and includes an off-trim correction. Let $\mathbf{p}$ be the design-space search direction, and μ the current line-search step length. Prior to running the flow solver, the design and trim parameters are modified according to

$$\mathbf{x}^{\text{des}}(\mu) = \mathbf{x}_k^{\text{des}} + \Delta\mathbf{x}^{\text{des}}, \quad \Delta\mathbf{x}^{\text{des}} = \mu\mathbf{p}, \quad \mathbf{x}^{\text{trim}}(\mu) = \mathbf{x}_k^{\text{trim}} + \frac{d\mathbf{x}^{\text{trim}}}{d\mathbf{x}^{\text{des}}}\Delta\mathbf{x}^{\text{des}}, \tag{7}$$

where the purpose of the $\mathbf{x}^{\text{trim}}$ modification is to maintain linearized trim conditions. The corresponding flow solution $\mathbf{U}(\mu)$ may not be exactly trimmed, as the state and outputs can be nonlinear, while the trim modification arises from linearizations about the zero step-length state and design. The objective calculation then includes an off-trim correction,

$$J(\mu) = J(\mathbf{U}(\mu), \mathbf{x}(\mu)) + \left.\frac{dJ}{d\mathbf{x}^{\text{trim}}}\right|_{\mathbf{U}(\mu),\mathbf{x}(\mu)} \Delta\mathbf{x}^{\text{trim}}(\mu), \tag{8}$$

where the trim parameter change is computed using $\mathbf{U}(\mu)$ and $\mathbf{x}(\mu)$.

Following the line search, the trim parameters are first modified according to Eqn. 7 for the linear modification, and are then subject to one trimming iteration using the trim sensitivity matrix already available from the last line-search iteration.

C. Airfoil Parametrization

We consider airfoils whose camber, $z_c(x)$, and thickness, $t(x)$, profiles are parametrized as polynomials with bounding multiplicative envelopes,

$$z_c(x) = x(1-x)\left[c_0 + c_1 x + \ldots + c_{p_c} x^{p_c}\right], \tag{9}$$

$$t(x) = a\ \text{abs}_{\text{smooth}}\left\{\sqrt{x}(1-x)\left[1 + t_1 x + \ldots + t_{p_t} x^{p_t}\right],\ 0.2x(1-x)\right\} \tag{10}$$

where c_i are the coefficients of the order p_c camber polynomial, t_i are the coefficients of the order p_t thickness polynomial, and the smooth absolute value function is defined for non-negative b as

$$\text{abs}_{\text{smooth}}(a,b) = \begin{cases} |a| & \text{if } |a| \geq b \\ (a^2+b^2)/(2b) & \text{otherwise} \end{cases} \tag{11}$$

The coefficient a enforces a given constant airfoil area, A,

$$\int_0^1 t(x)\,dx = A. \tag{12}$$

IV. Error Estimation and Metric Calculation

A. Adjoint-Weighted Residual

In this work, we use the adjoint solution not only to calculate shape sensitivities, but also to estimate the numerical error in the optimization objective J. This error estimation is done via the adjoint-weighted residual [2, 18]. Let the subscript H denote the current "coarse" spatial discretization, and h a fine one, obtained by increasing the approximation order by one, $p \longrightarrow p+1$. $\mathbf{U}_h^H$ is the state vector field prolongated from the coarse space to the fine space. The fine-space residual computed using the prolongated state and weighted by the fine-space adjoint gives an estimate of the output error between the coarse and fine spaces,

$$J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx -\delta\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H). \tag{13}$$

In this expression, $\delta\mathbf{\Psi}_h$ is the adjoint error, obtained by subtracting from the fine-space adjoint its projection onto the coarse space. The fine-space adjoint is solved "exactly" to a strict relative error tolerance, using the same preconditioned GMRES method as for the primal. The error estimate in Eqn. 13 is localized to elements, resulting in the elemental error indicators ε_e

$$J \approx \sum_{e=1}^{N_e} -\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H) \quad \Rightarrow \quad \varepsilon_e \equiv \left|\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)\right|, \tag{14}$$

where N_e is the number of elements, and the subscript e denotes components of the adjoint and residual associated with element e.

B. Mesh Optimization through Error Sampling and Synthesis (MOESS)

The adjoint-weighted residual error from Eqn. 14 drives a metric-based optimization based on MOESS [3, 19]. A Riemannian metric field, $\mathcal{M}(\vec{x})$, encodes information about the desired size and stretching of the elements in a computational mesh. The set of points at unit metric distance from $\vec{x}$ is an ellipse, the principal axes and lengths of which are determined by the eigenvectors and eigenvalues of $\mathcal{M}$.

Affine-invariant changes [20] to the metric field are made via a symmetric *step matrix*, $\mathcal{S} \in \mathbb{R}^{d\times d}$, where d is the spatial dimension, according to

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S}) \mathcal{M}_0^{\frac{1}{2}}. \tag{15}$$

Metric optimization requires models for how the elemental error indicator and the cost change as the metric changes. The error model is

$$\varepsilon_e = \varepsilon_{e0} \exp\left[\mathrm{tr}(\mathcal{R}_e \mathcal{S}_e)\right], \tag{16}$$

where ε_{e0} is the initial error on element e, and $\mathcal{R}_e$ is an element-specific rate tensor determined through a sampling procedure. The cost model is

$$C_e = C_{e0} \exp\left[\frac{1}{2}\mathrm{tr}(\mathcal{S}_e)\right], \tag{17}$$

where C_{e0} is the initial cost, measured in terms of degrees of freedom. Details of both of these models can be found in previous work [19].

Given a mesh with a mesh-implied metric $\mathcal{M}_0(\vec{x})$, elemental error indicators ε_{e0}, and elemental rate tensor estimates, $\mathcal{R}_e$, the goal of the metric optimization algorithm is to determine the step matrix field, $\mathcal{S}(\vec{x})$, that minimizes the error at a fixed cost. This algorithm iteratively equidistributes the marginal error to marginal cost ratio over the mesh elements. In practice, the mesh optimization and flow/adjoint solution are performed several times at a given target cost, C_{target}, until the error stops changing, at which point the cost can be increased.

V. Mesh Node Movement

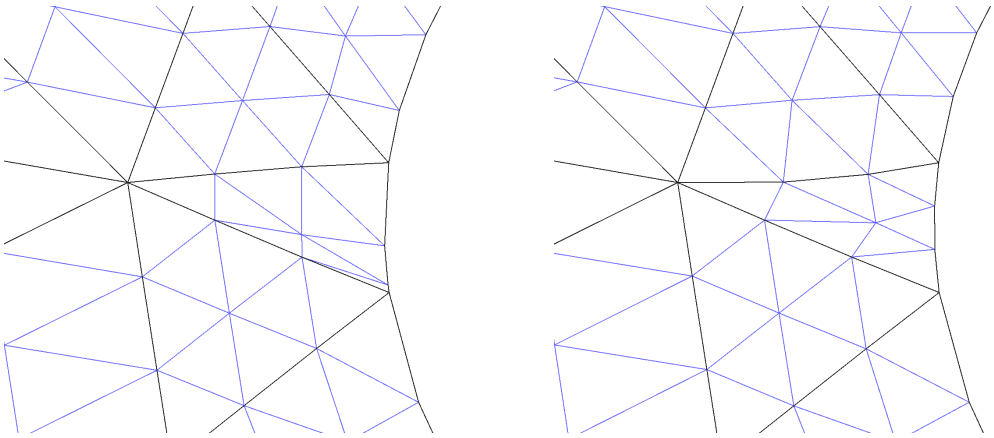
We use mesh node movement to control the discretization error during the shape optimization. The node movement is formulated as a metric-based mesh optimization, guided by output-based error estimates. The mesh optimization

essentially finds the optimal locations of vertices and high-order nodes with the goal of making the mesh better conform to a desired metric field. Conformity to a desired metric field translates to lower discretization error. In this work, we explore methods to simultaneously adjust the locations of the high-order nodes as vertices are moved. As a start, we will move the high-order nodes using warped-element refinement method. We will then implement a different optimization architecture that allows for more coupling between the vertex and high-order node movements.

A. Vertex Position Optimization

As a stepping stone towards optimizing the positions of both vertices (i.e. "Q1 nodes") and high-order nodes, we first simplify the problem by considering only the vertices for the optimization problem. We reposition the vertices to better conform to a requested metric, which comes from MOESS. The algorithm for this repositioning is presented in detail in [21]. It formulates an unconstrained optimization problem for the displacement of vertices with the error in the elemental step matrix as an objective function. This large optimization problem, which is not overly large for coarse high-order meshes, is solved using a limited-memory (L-BFGS) algorithm [22]. Presently, we combine this vertex movement with the shape optimization in a simple coupling: the vertices are moved after each BFGS shape optimization iteration. No vertex movement occurs within the BFGS line-search iterations. The advantage of applying node movement instead of remeshing is that the latter changes the size of the discrete problem, requiring parallel repartitioning and remapping of states and adjoints, and that remeshing with anisotropic curved elements lacks robustness for complex geometries. In shape optimization, where many iterations of adaptation on changing geometries are required, robustness of mesh changes is paramount.

In the node movement procedure described in [21], the high-order node displacements are interpolated from the vertex displacements. Over many shape-optimization iterations, in which vertices can propagate large distances around a changing geometry, bunching of high-order nodes can occur, as illustrated in Fig. 1. In this work, we therefore modify the high-order node placement via a new repositioning step, described below, which is applied after each vertex movement iteration.



(a) Bunched high-order nodes (b) After repositioning

Fig. 1 Alleviation of high-order node bunching during vertex movement via distance-based repositioning.

Let $\vec{x}_k$, $1 \le k \le N_Q$, denote the current positions of all N_Q nodes, including the high-order ones, inside one element. Here we assume that this element is a triangle, but the same procedure can be applied to other element types. Suppose that we know the reference-space coordinates, (ξ_k, η_k), of these nodes. We seek new node positions, $\tilde{\vec{x}}_l$, $1 \le k \le N_Q$, for equally-spaced reference coordinates $(\tilde{\xi}_l, \tilde{\eta}_l)$ that yield a Lagrange interpolant of $\vec{x}_k$ at (ξ_k, η_k). That is, the desired Lagrange interpolant is

$$\vec{x}(\xi, \eta) = \sum_{l=1}^{N_Q} \tilde{\vec{x}}_l \phi_l(\xi, \eta), \tag{18}$$

where $\phi_l(\xi, \eta)$ are the Lagrange basis functions on equally-spaced reference nodes. In order to interpolate the original

nodes, we require $\tilde{\vec{x}}_l$ to solve the following linear system:

$$\sum_{l=1}^{N_Q} \phi_l(\xi_k, \eta_k)\tilde{\vec{x}}_l = \vec{x}_k. \tag{19}$$

The solution involves inverting a matrix whose entries are the Lagrange basis functions evaluated at (ξ_k, η_k).

What remains is specifying the reference coordinates, (ξ_k, η_k), of the original nodes. We do this by a distance-based edge formula, in which the reference coordinates are calculated using distances between nodes in global space, as illustrated in Figure 2. The distance-based weighting approximates the final arc-length position of each interpolated original point in global space. Edge nodes shared between elements are displaced by an average of the repositioning calculation on the adjacent elements.

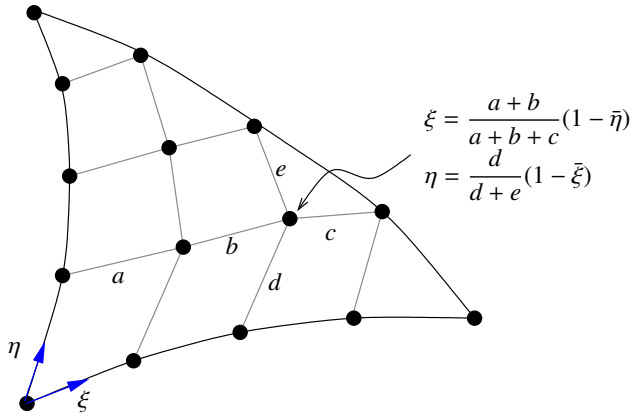


Fig. 2 Illustration of distance-based reference coordinate calculation for face node repositioning. $\bar{\xi}$ and $\bar{\eta}$ are calculated by a weighted average of unscaled distance-based reference coordinates along the shared column and row of nodes, respectively.

B. Warped-Element Refinement Method

In this section, we briefly review the warped-element refinement method (q-adaptation). The key idea here is to strategically place the high-order nodes within a mesh element given fixed vertex locations and without changing the original outer shape of the element. The goal of the high-order node movement can be set to minimizing errors in solution approximations [23], engineering output predictions [4], or metric approximations [5, 6, 24]. This refinement/adaptation method embraces clustering of the high-order nodes within an element as some clustering (when done properly) can offer accuracy and robustness benefits. Of course, the validity of the element must always be maintained.

In a q-adapted mesh, warping of mesh elements occurs in regions where complex geometries and/or important flow features are present. Here, we note that *warping* of mesh elements specifically refers to the changes in the *inner* shape of the elements. That is, the edges of these elements remain unchanged and the vertices are at the same locations as in the initial mesh (i.e., the one before q-adaptation is applied). This avoids mesh tangling issues, which can be particularly difficult to resolve for high-order curved meshes. Figure 3 illustrates the concept of element warping and the current status of q adaptation.

Warping of mesh elements is done through distorting the reference-to-global mapping used in the geometry approximation:

$$\vec{x}(\vec{\xi}) = \sum_{i=1}^{N_q} \vec{x}_i \phi_i(\vec{\xi}) \tag{20}$$

$$\underline{J} = \frac{\partial \vec{x}}{\partial \vec{\xi}} = \sum_{i=1}^{N_q} \vec{x}_i \frac{\partial \phi_i}{\partial \vec{\xi}}(\vec{\xi}), \tag{21}$$

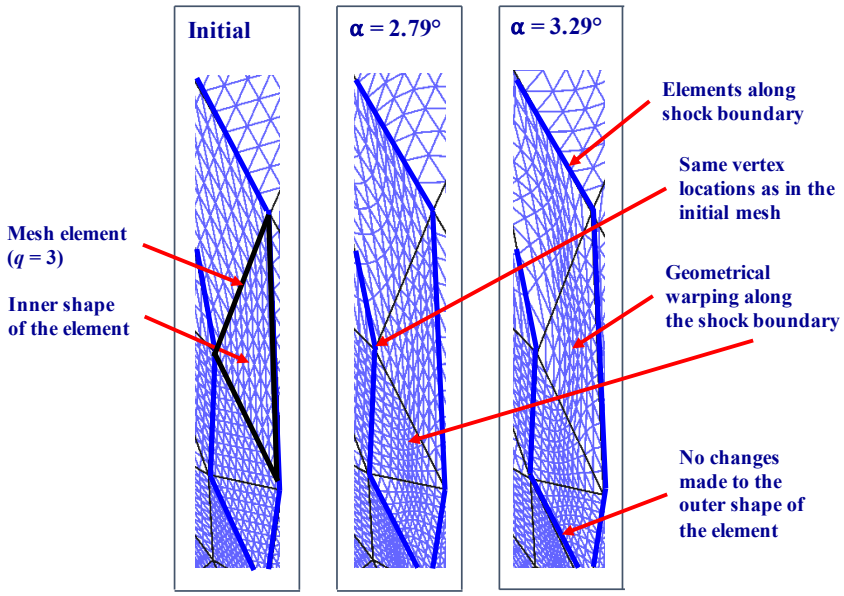


Fig. 3 Sample q-adapted mesh around a shock boundary in a RANS problem over an RAE 2822 airfoil. The different angles of attack (α) are shown to highlight the changes made to the *inner* shapes of the initial mesh elements. The outer shapes of the elements and the vertex locations remain the same as in the initial mesh [4].

where $\vec{x}$ is the global-space coordinate, $\vec{\xi}$ is the reference-space coordinate, ϕ_i are the basis functions for geometry approximation, N_q is the number of geometry nodes in a mesh element, and $\underline{J}$ is the mapping Jacobian matrix. This approach is desirable since it allows us to take advantage of the existing infrastructure that is already available in most high-order codes. In this paper, we use Lagrange basis functions with uniform node distribution in the reference space since they allow for an intuitive specification of the high-order element. Other basis functions could also be used to define the element geometry, as long as the resulting mapped elements constitute a complete, non-overlapping tessellation of the domain.

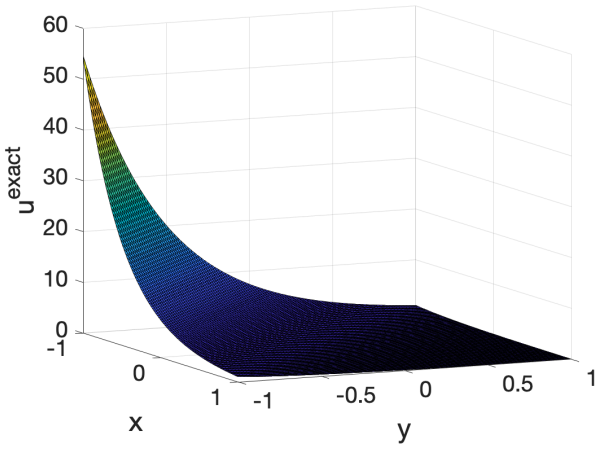
C. Coupling Between Vertices and High-Order Nodes

In the context of high-order meshes, the optimal locations of high-order nodes depend on the optimal locations of vertices and vice versa. Thus, ideally the mesh node movement strategy fully captures the global coupling between vertices and high-order nodes. That is, once a vertex or a high-order node is moved, the rest of the mesh nodes are automatically readjusted. From this viewpoint, the warped-element refinement method cannot capture global coupling, since it restricts the high-order node movement to maintain an element's outer shape and vertex locations. Nevertheless, the warped-element refinement method is still a convenient and powerful tool to improve upon linear meshes or heuristics used to determine the locations of high-order nodes.

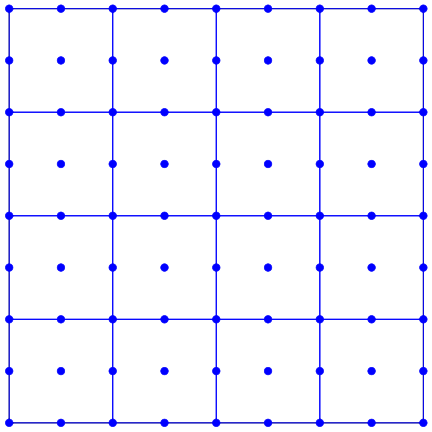
As an attempt to capture the aforementioned global coupling (and to curve an element's edges), we study global mesh optimization for solution approximation problems on structured meshes. Here, we use all-at-once and sequential optimization approaches to minimize the least-squares solution errors by moving the vertices and high-order nodes globally. All mesh nodes are movable as long as they do not alter the computational domain. The initial mesh is uniform, and the optimization parameters are set to the same values for all cases presented in Figs. 4 and 5. No parameter tuning is performed here. The main takeaways from this study are:

- Global mesh optimization is more expensive, but improves accuracy and robustness, especially for coarse meshes or higher geometry approximation order (q).
- The all-at-once approach tends to find a better optimum than the warped-element refinement method and sequential optimization, but not always.

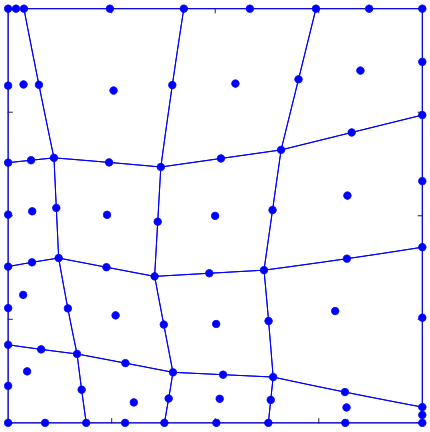
For the final paper, we will implement one of the global optimization approaches and perform a shape optimization on a coarse mesh, but with high p and q orders.



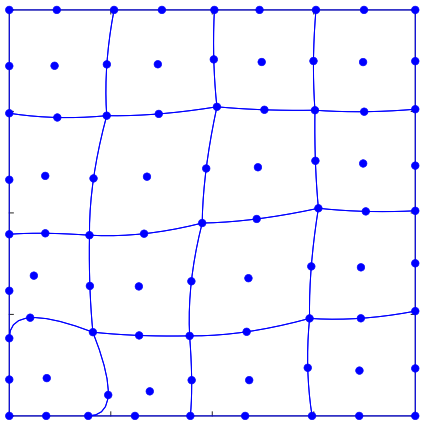
(a) Exact solution



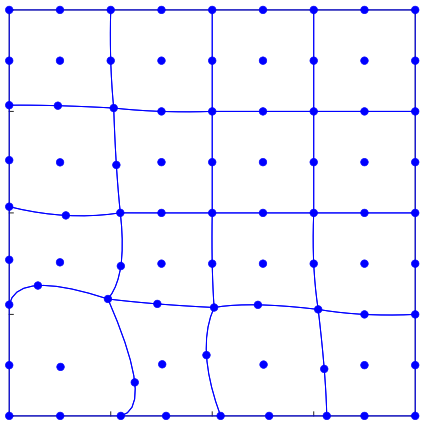
(b) Initial mesh ($\varepsilon_0/\varepsilon = 1$)



(c) Warped-element refinement ($\varepsilon_0/\varepsilon \approx 2.7$)

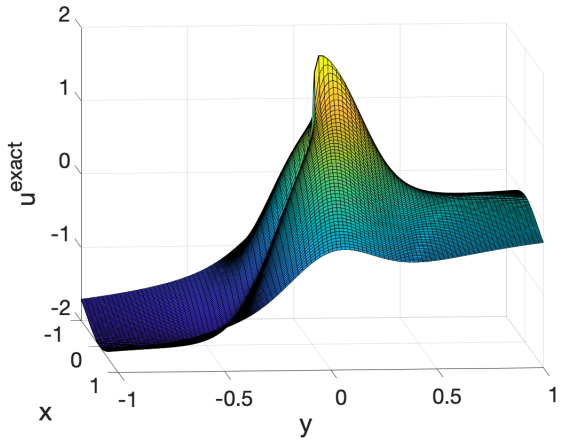


(d) All-at-once ($\varepsilon_0/\varepsilon \approx 34.4$)

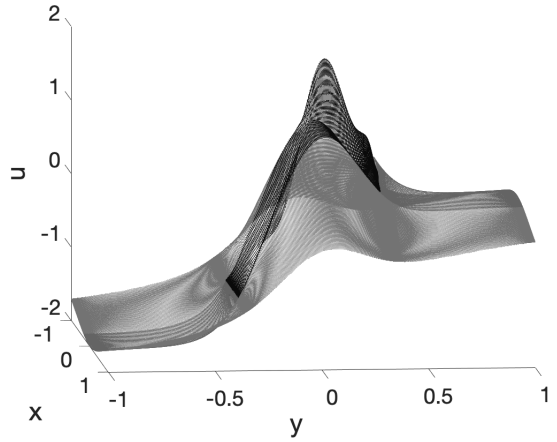


(e) Sequential optimization ($\varepsilon_0/\varepsilon \approx 11.1$)

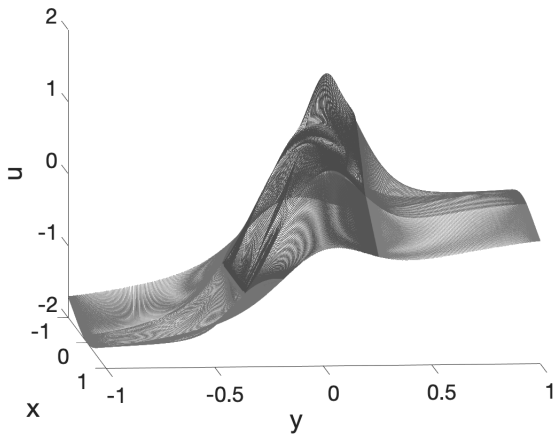
Fig. 4 Comparison of $q = 2$ meshes for approximating an exponential function with $p = 2$ and 16 mesh elements. $\varepsilon_0/\varepsilon$ shows the error reduction factor compared to the initial, uniform mesh.



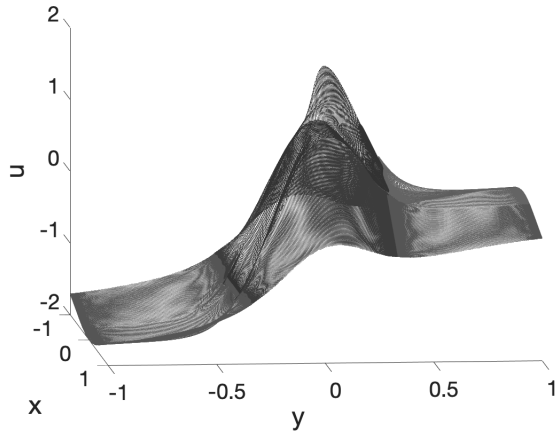
(a) Exact solution



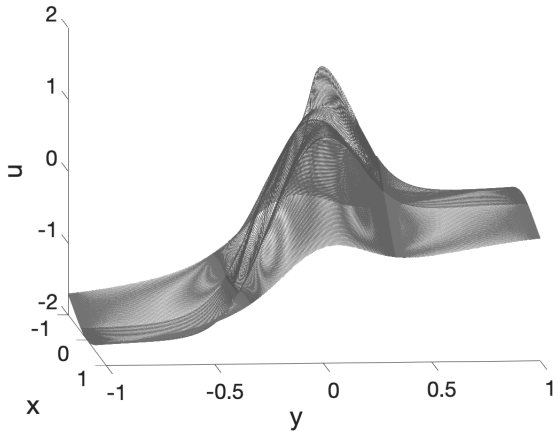
(b) Initial mesh ($\varepsilon_0/\varepsilon = 1$)



(c) Warped-element refinement ($\varepsilon_0/\varepsilon \approx 64.4$)



(d) All-at-once ($\varepsilon_0/\varepsilon \approx 7.4$)



(e) Sequential optimization ($\varepsilon_0/\varepsilon \approx 157.6$)

Fig. 5 Comparison of $q = 8$ meshes for approximating a "shock-like" function with $p = 8$ and 9 mesh elements. $\varepsilon_0/\varepsilon$ shows the error reduction factor compared to the initial, uniform mesh. Lighter shade indicates lower error.

VI. Implementation

Our proposed frameworks for shape optimization with concurrent mesh node movement are summarized in Algorithms 1 and 2. The main difference lies in the workflow and cost allocation for mesh node movement. Algorithm 1 is computationally less expensive, and aims to localize the effect of node movement. Algorithm 2 is more computationally expensive since it aims to capture the global coupling between the vertices and high-order nodes. Visually, we will see that the resulting mesh from Algorithm 1 will maintain the original outer shapes of the elements, though clustering of high-order nodes is allowed. Changes to the outer shapes of the elements (including curving elements' edges) will only occur in meshes resulting from Algorithm 2.

Algorithm 1: Optimization with error estimation and local node movement

input: initial design $\mathbf{x}_0$, initial mesh T_h, mesh adaptation iterations N_{adapt};
output: adapted mesh T_h^{adapt}, optimal design $\mathbf{x}^*$;
while *not converged* **do**
  move the vertices in the mesh T_h as if all elements are $q = 1$ ▷ MOESS/Hessian-based adaptation;
  for $i = 1, 2, \cdots, N_{adapt}$ **do**
    move the high-order nodes given the optimized vertices $\rightarrow T_h^{\text{adapt}}$ ▷ warped-element refinement;
    check the validity of the high-order mesh ▷ backtracking (if needed);
  end
  update $\mathbf{x}^{\text{trim}}$ to meet trim constraints ▷ Eqn. 8;
  compute the objective J and the error estimates δJ;
  calculate objective gradient $\left.\frac{dJ}{d\mathbf{x}^{\text{des}}}\right|_{\text{trim}}$ ▷ Eqn. 6;
  update $\mathbf{x}^{\text{des}}$ on T_h^{adapt} ▷ line search;
end

Algorithm 2: Optimization with error estimation and global node movement

input: initial design $\mathbf{x}_0$, initial mesh T_h, mesh adaptation iterations N_{adapt};
output: adapted mesh T_h^{adapt}, optimal design $\mathbf{x}^*$;
while *not converged* **do**
  for $i = 1, 2, \cdots, N_{adapt}$ **do**
    move the vertices and high-order nodes in the mesh $T_h \rightarrow T_h^{\text{adapt}}$ ▷ all-at-once or sequential;
    check for the validity of the high-order mesh ▷ backtracking (if needed);
  end
  update $\mathbf{x}^{\text{trim}}$ to meet trim constraints ▷ Eqn. 8;
  compute the objective J and the error estimates δJ;
  calculate objective gradient $\left.\frac{dJ}{d\mathbf{x}^{\text{des}}}\right|_{\text{trim}}$ ▷ Eqn. 6;
  update $\mathbf{x}^{\text{des}}$ on T_h^{adapt} ▷ line search;
end

VII. Results

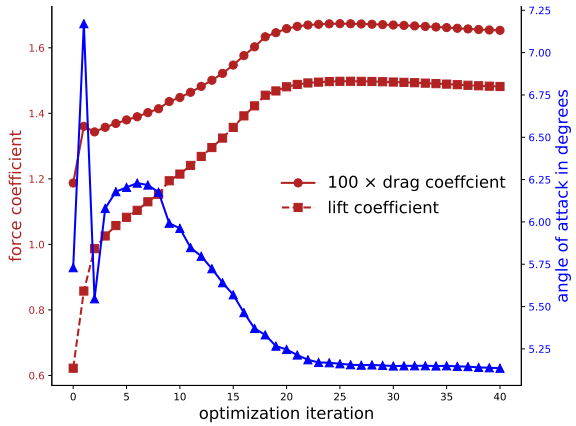
The two-dimensional shape optimization cases presented here are chosen to demonstrate the potential benefits of node movement as a discretization error control strategy for shape optimization on structured meshes.

A. Effect of Boundary-Layer Resolution in a RANS Optimization

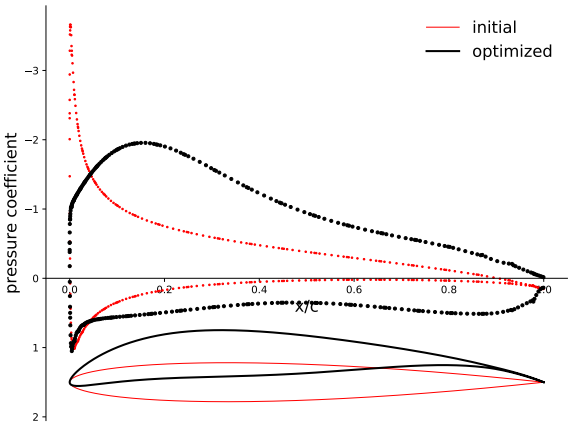
As a first demonstration, we consider optimizing the shape of an airfoil flying at $M = 0.2$ and $Re = 10^6$, with area $A = 0.06c^2$, to produce the highest lift-to-drag ratio, L/D. The physical model is the Spalart-Allmaras RANS

equations. In this case, we do not employ any of the node node optimization algorithms presented, and instead consider optimization on static meshes with different element sizing. The goal is to demonstrate that optimization can take advantage of numerical errors and converge to an incorrect shape and an infeasible L/D.

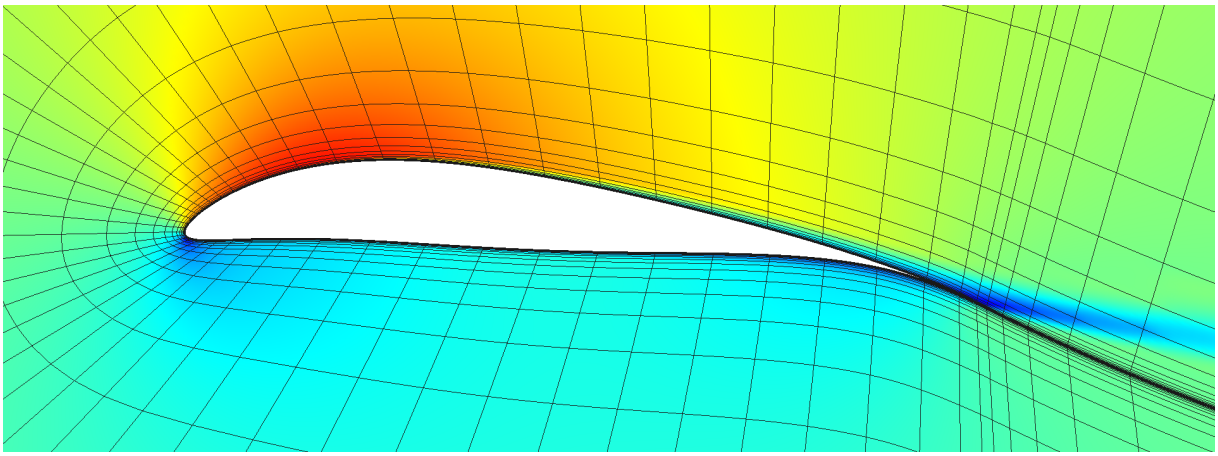
The airfoil is parametrized as described in Section III.C, with 4 camber and 3 thickness parameters. The angle of attack is an additional optimization parameter, for a total of 8. The initial airfoil has all shape parameters set to zero. Fig. 6 shows the results of shape optimization performed on a fine mesh of 2816 elements, using $p = 2$ solution approximation. We observe the following characteristics of the optimized airfoil: increased camber, a shift of the area towards the middle of the airfoil, a thin flap-like trailing edge, and a narrower, slightly drooping leading edge. The optimal L/D is calculated to be 89.6 after 40 optimization iterations.



(a) Optimization history



(b) Pressure coefficient distribution

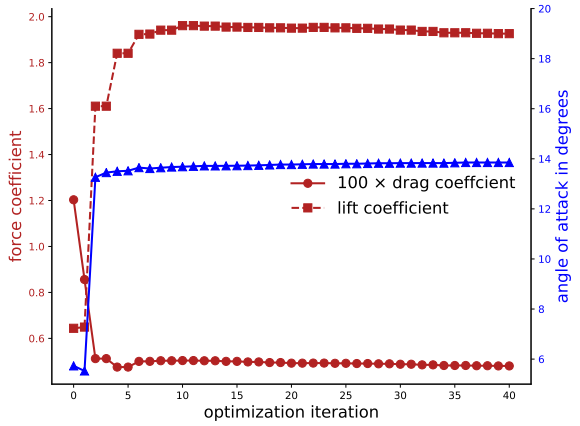


(c) Optimized-shape Mach number contours, range 0-0.4

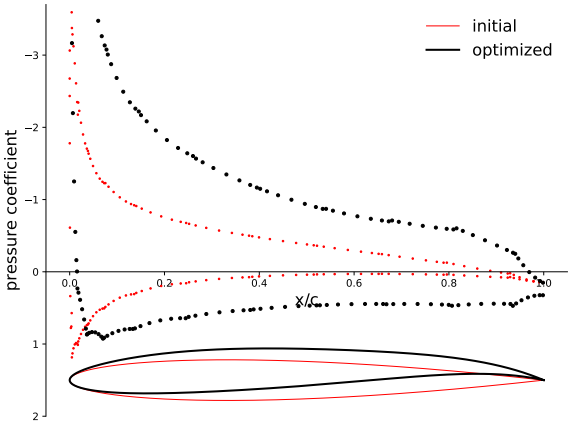
Fig. 6 Fine-mesh optimization: $L/D = 89.6$.

We now repeat the same optimization but on a coarse mesh of 704 elements, still with $p = 2$. Fig. 7 shows the mesh and shape optimization result. While the mesh has highly anisotropic elements near the boundary, the discretization error of the coarser mesh pollutes the results significantly. The drag coefficient is severely underpredicted due to the lack of sufficient boundary-layer resolution, leading to a spurious optimum with $L/D = 407$, almost five times higher than the fine-mesh result. A fine-mesh analysis of this design, shown in Fig. 8, indicates that the flow does not stay attached as long as predicted on the coarse mesh, and that the lift-to-drag ratio is actually over an order of magnitude lower, $L/D = 39.4$.

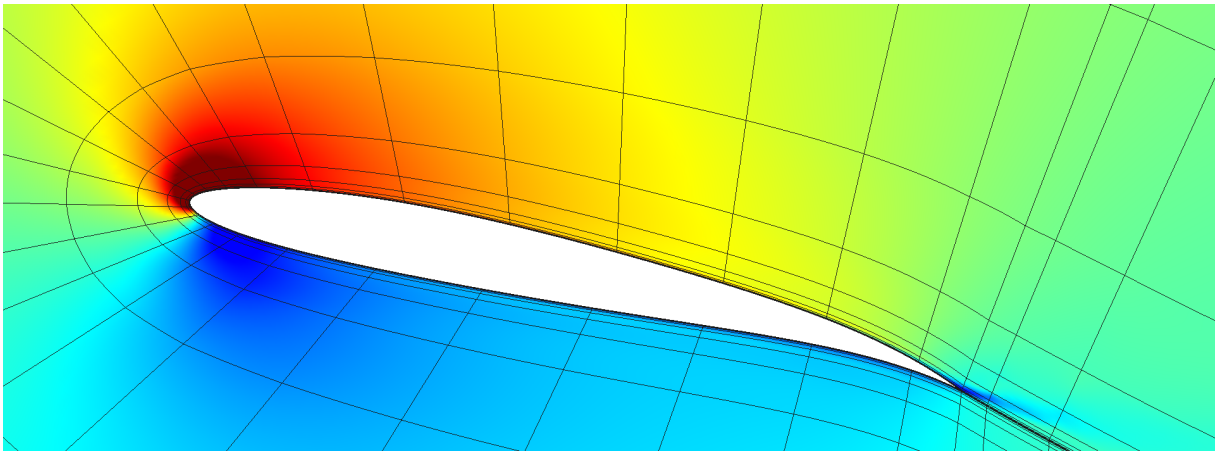
Slightly repositioning the mesh nodes by decreasing the distance of the first layer of nodes away from the wall, without changing the element count of 704, results in the optimized design shown in Figure 9. This design is much closer to the fine-mesh optimum, as is the L/D value of 85.6. From this example, we see that accurate analyses and



(a) Optimization history



(b) Pressure coefficient distribution



(c) optimized-shape Mach number contours, range 0-0.4

Fig. 7 Coarse-mesh optimization: $L/D = 407$**.**

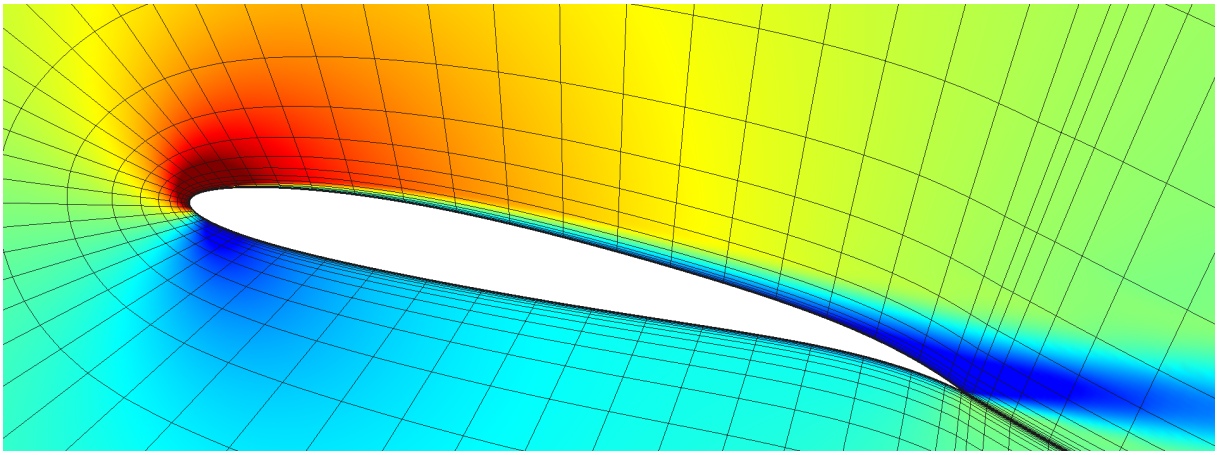
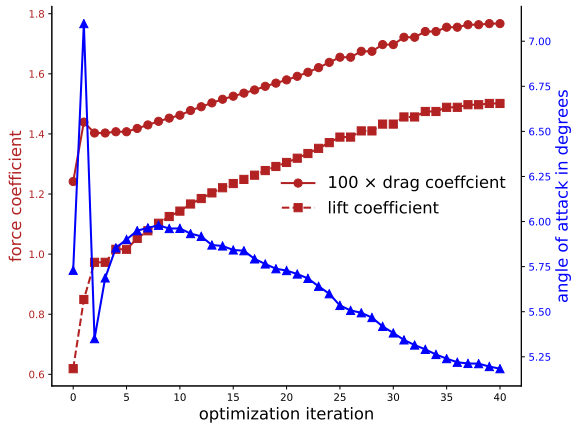
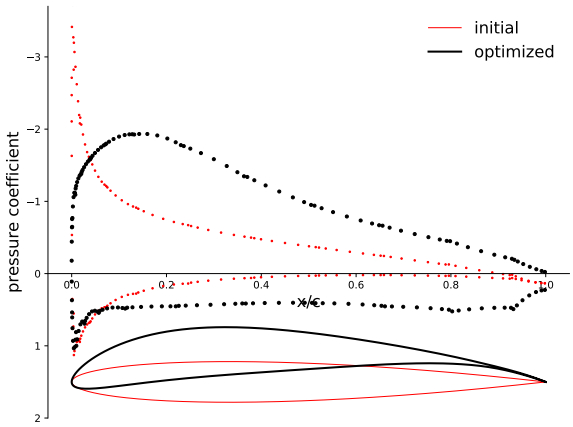


Fig. 8 Fine-mesh analysis of coarse-mesh design: $L/D = 39.4$**.**

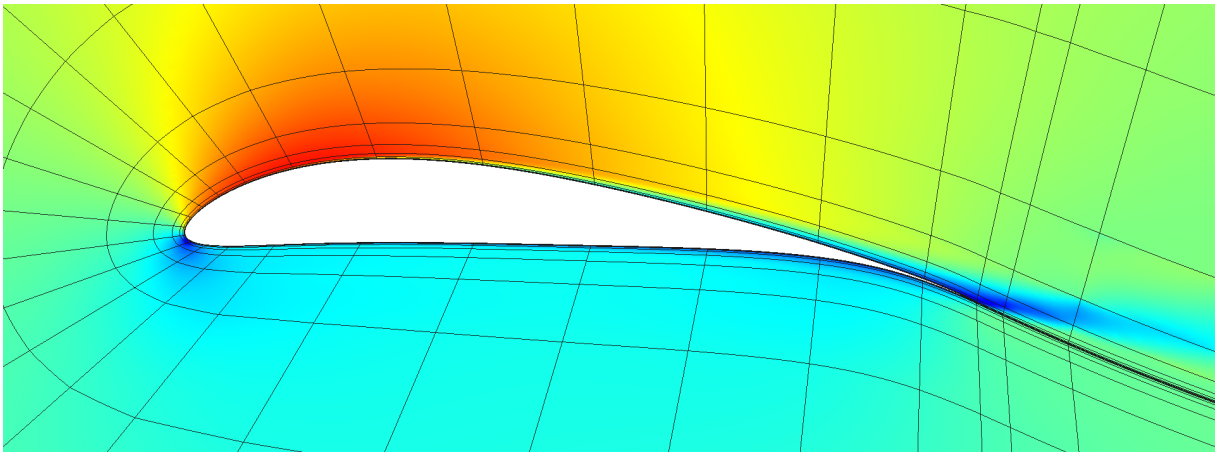
optimizations are possible on coarse meshes, as long as the nodes are positioned appropriately – i.e. as long as the mesh is optimal.



(a) Optimization history



(b) Pressure coefficient distribution



(c) Optimized-shape Mach number contours, range 0-0.4

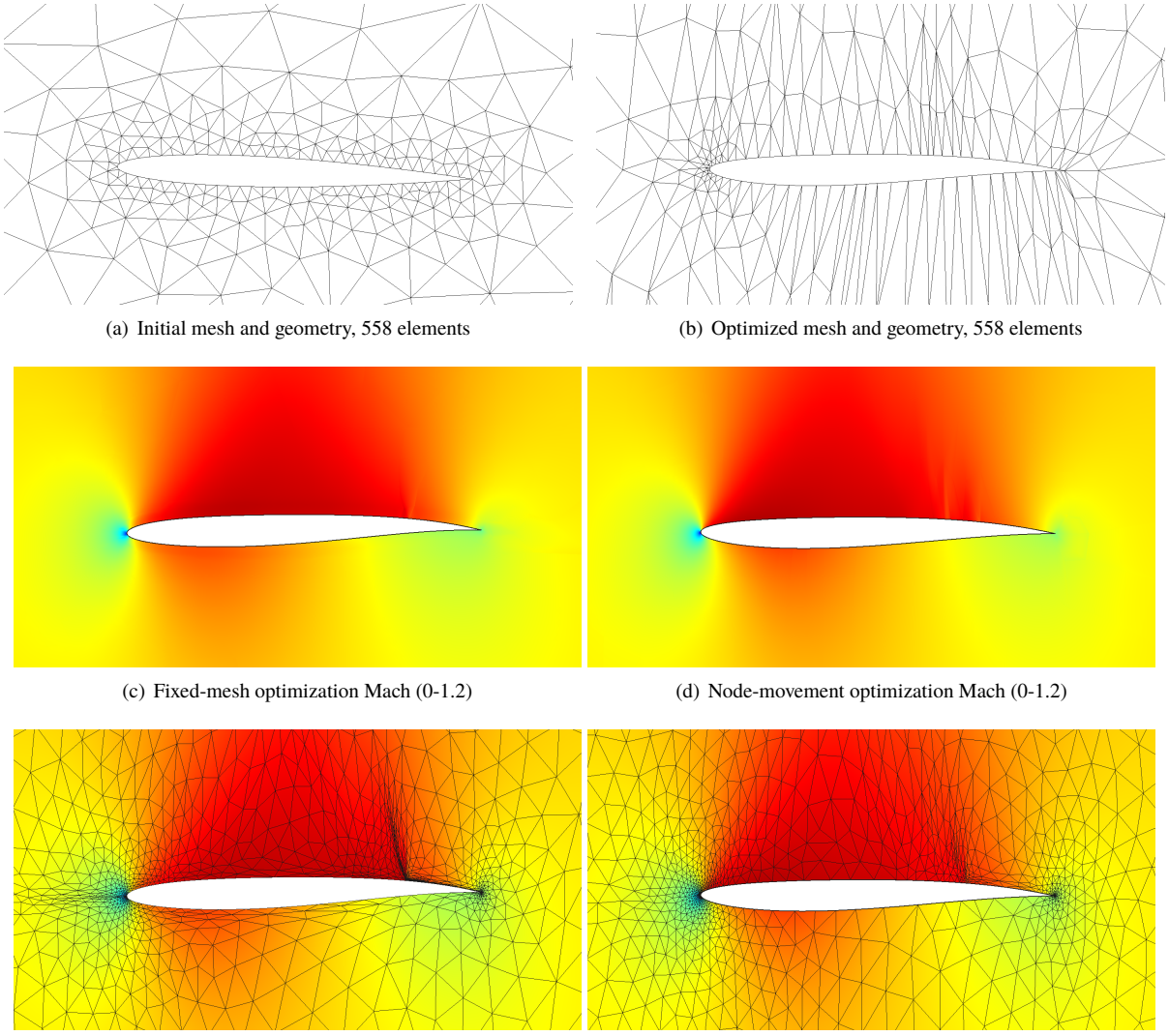
Fig. 9 Repositioned-node, coarse-mesh optimization: $L/D = 85.6$**.**

B. Transonic Airfoil Optimization With Node Movement

In this problem we consider optimizing the shape of an airfoil with area $A = .06c^2$ to produce the minimum drag at a fixed lift coefficient of $c_\ell = 0.5$ in $M = 0.8$ flow. The physical model consists of the Euler equations, with artificial-viscosity shock capturing [25]. The meshes are unstructured in this case, and mesh deformation following shape changes is handled using radial basis functions. $Q = 3$ curved elements are used in all of the meshes.

The airfoil is parametrized as described in Section III.C, with 3 camber and 2 thickness parameters. The angle of attack is a trim parameter used to enforce the lift-coefficient constraint. We start the optimization on a coarse mesh of 558 elements, shown in Fig. 10a using $p = 2$ approximation. Fig. 10c shows the shape optimization result using the initial mesh. This is called the fixed-mesh optimization because the initial mesh does not change during the optimization. The Mach contours show a region of supersonic flow over the upper surface, but no strong shocks. However, a fine-mesh analysis with approximately 3000 elements, trimmed to the same lift coefficient and shown in Fig. 10e, indicates the presence of a shock, which then increases the drag coefficient. The initial mesh is too coarse to capture this shock, and hence the optimizer is not aware of its impact on the drag.

The right column of Fig. 10 shows the result of shape optimization combined with the proposed vertex movement at every optimization iteration. Compared to the fixed-mesh optimized shape, the airfoil is flatter on the upper surface,



(a) Initial mesh and geometry, 558 elements

(b) Optimized mesh and geometry, 558 elements

(c) Fixed-mesh optimization Mach (0-1.2)

(d) Node-movement optimization Mach (0-1.2)

(e) Fine-mesh analysis of fixed-mesh result, Mach (0-1.2)

(f) Fine-mesh analysis of node-movement result, Mach (0-1.2)

Fig. 10 Transonic airfoil shape optimization results using a fixed mesh (left column) and the proposed node movement within the shape optimization iterations (right column). The approximation order is $p = 2$.

thicker towards the front, and more cambered towards the rear. The corresponding mesh with vertex movement has more resolution at the leading edge, and anisotropic, vertically-aligned elements on the upper surface, which are better for capturing shocks and features of the adjoint. A fine-mesh analysis of this geometry, also with approximately 3000 elements, indicates lower Mach numbers over the upper surface terminated by a shock that is weaker compared to the fixed-mesh geometry. This result indicates that, with vertex movement, even the coarse 558-element mesh can be tuned to approximate the features of the transonic flow around the optimized shape.

Tab. 1 presents the drag coefficient and angle of attack results from optimizations at two different orders, $p = 1$ and $p = 2$. The mesh for the $p = 1$ runs was finer, at 1199 elements, compared to that for the $p = 2$ runs, 558 elements, to make the number of degrees of freedom similar with the discontinuous-Galerkin discretization. The table also presents a fine-mesh trimmed analysis result, obtained using an adapted mesh with over six times as many degrees of freedom. We first note that all of the drag coefficient values are small, as the problem is inviscid and the shock can be mostly eliminated with an optimal shape. We observe that the $p = 2$ fixed-mesh has the largest fine-grid analysis drag coefficient, at 1.40 counts (1 count = 0.0001 c_d), followed by the $p = 1$ fixed-mesh shape at 1.16 counts. Next is the $p = 1$ shape obtained on a mesh with node movement, at 1.10 counts, followed by the best performance from the $p = 2$ optimization with node movement, at 0.99 counts. The reason for the improved performance with node movement is the lower discretization error on these meshes, which leads to the optimizer working with more accurate outputs. In addition, adaptive $p = 2$ outperforms adaptive $p = 1$, even though it has slightly fewer degrees of freedom (3348 versus 3597), which indicates that the $p = 2$ approximation makes a more efficient use of its degrees of freedom. Finally, we note that the drag coefficients predicted on the coarse meshes still greatly over-estimate the drag coefficient, due to increased numerical dissipation, but that the results with node movement are closer to the fine-mesh analyses, as expected.

Table 1 Transonic airfoil shape-optimization results on fixed meshes and on meshes with node movement. The fine-grid columns refer to trimmed analysis of the optimized shape using a finer adapted mesh.

Case	α **(deg)**	c_d **(counts)**	**fine-grid** α **(deg)**	**fine-grid** c_d **(counts)**
$p = 1$ fixed-mesh	.029	27.8	0.147	1.16
$p = 1$ node movement	-0.28	6.46	-0.20	1.10
$p = 2$ fixed-mesh	0.50	5.94	0.58	1.40
$p = 2$ node movement	-0.27	3.68	-0.18	0.99

The node movement in these cases is performed by solving a global optimization problem for the positions of the vertices, which are the $Q = 1$ nodes. The high-order node positions are not included in the optimization, and their displacements are interpolated from the $Q = 1$ node displacements, with the repositioning correction discussed in Section V.A. Including the high-order nodes in the optimization is the subject of ongoing work.

VIII. Conclusions and Future Work

In this paper, we present node movement as a discretization error control strategy for shape optimization problems on high-order, curved meshes. For these cases, we show that slight repositioning of the elements' vertices can significantly improve the solution fields, even without changing the element count. On coarse meshes, repositioning of the vertices becomes more critical since the solution fields may be captured incorrectly when vertices are not placed optimally. We have shown this in a decoupled manner, by first considering the optimization of vertices, in which the high-order nodes are interpolated, and then by presenting algorithms for optimizing the high-order node locations. The results indicate accuracy and robustness benefits of moving vertices and high-order nodes for solution approximation problems. In future work, we plan to combine both ideas and show that moving vertices and high-order nodes is an effective discretization control strategy for shape optimization problems on high-order meshes.

IX. Acknowledgement

The authors acknowledge the financial support of the University of Tennessee and the University of Michigan.

References

[1] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845. https://doi.org/10.1002/fld.3767.

[2] Fidkowski, K. J., and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694. https://doi.org/10.2514/1.J050073.

[3] Yano, M., "An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.

[4] Sanjaya, D. P., and Fidkowski, K. J., "Improving High-Order Finite Element Approximation Through Geometrical Warping," *AIAA Journal*, Vol. 54, No. 12, 2016, pp. 3994–4010. https://doi.org/10.2514/1.J055071.

[5] Sanjaya, D. P., "Towards Automated, Metric-Conforming, Mesh Optimization for High-Order, Finite-Element Methods," Ph.D. thesis, University of Michigan: Ann Arbor, 2019.

[6] Sanjaya, D. P., Fidkowski, K. J., and Murman, S. M., "Comparison of Algorithms for High-Order Metric-Based Mesh Optimization," AIAA Paper 2020–1141, 2020. https://doi.org/10.2514/6.2020-1141.

[7] Venditti, D. A., and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal for Computational Physics*, Vol. 176, No. 1, 2002, pp. 40–69. https://doi.org/10.1006/jcph.2001.6967.

[8] Nemec, M., and Aftosmis, M. J., "Adjoint Sensitivity Computations for an Embedded-Boundary Cartesian Mesh Method," *Journal for Computational Physics*, Vol. 227, No. 1, 2008, pp. 2724–2742. https://doi.org/10.1016/j.jcp.2007.11.018.

[9] Chen, G., and Fidkowski, K. J., "Airfoil Shape Optimization Using Output-Based Adapted Meshes," AIAA Paper 2017–3102, 2017. https://doi.org/10.2514/6.2017-3102.

[10] Chen, G., and Fidkowski, K. J., "Discretization error control for constrained aerodynamic shape optimization," *Journal of Computational Physics*, Vol. 387, 2019, pp. 163–185. https://doi.org/10.1016/j.jcp.2019.02.038.

[11] Chen, G., and Fidkowski, K. J., "Variable-Fidelity Multipoint Aerodynamic Shape Optimization With Output-Based Adapted Meshes," *Aerospace Science and Technology*, Vol. 105, 2020, p. 106004. https://doi.org/10.1016/j.ast.2020.106004.

[12] Allmaras, S., Johnson, F., and Spalart, P., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902, 2012.

[13] Ceze, M. A., and Fidkowski, K. J., "High-Order Output-Based Adaptive Simulations of Turbulent Flow in Two Dimensions," AIAA Paper 2015–1532, 2015. https://doi.org/10.2514/6.2015-1532.

[14] Reed, W., and Hill, T., "Triangular Mesh Methods for the Neutron Transport Equation," Los Alamos Scientific Laboratory Technical Report LA-UR-73-479, 1973.

[15] Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[16] Bassi, F., and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by B. Cockburn, G. Karniadakis, and C.-W. Shu, Springer, Berlin, 2000, pp. 197–208.

[17] J. E. Dennis, J., and More, J. J., "Quasi-Newton Methods, Motivation and Theory," *Society for Industrial and Applied Mathematics Review*, Vol. 19, 1977, pp. 359 – 372. URL http://dx.doi.org/10.1137/1019005.

[18] Becker, R., and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.

[19] Fidkowski, K. J., "A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization," AIAA Paper 2016–0835, 2016. https://doi.org/10.2514/6.2016-0835.

[20] Pennec, X., Fillard, P., and Ayache, N., "A Riemannian framework for tensor computing," *International Journal of Computer Vision*, Vol. 66, No. 1, 2006, pp. 41–66.

[21] Fidkowski, K. J., "Output-Based Mesh Optimization Using Metric-Conforming Node Movement," AIAA Paper, 2023.

[22] Liu, D. C., and Nocedal, J., "On the Limited Memory BFGS Method for Large Scale Optimization," *Mathematical Programming*, Vol. 45, 1989, pp. 503 – 528. https://doi.org/10.1007/BF01589116.

[23] Sanjaya, D. P., and Fidkowski, K. J., "Improving High-Order Finite Element Approximation Through Geometrical Warping," AIAA Paper 2015–2605, 2015. https://doi.org/10.2514/6.2015-2605.

[24] Sanjaya, D. P., Fidkowski, K. J., Diosady, L. T., and Murman, S. M., "Error Minimization via Metric-Based Curved-Mesh Adaptation," AIAA Paper 2017–3099, 2017. https://doi.org/10.2514/6.2017-3099.

[25] Persson, P.-O., and Peraire., J., "Sub-cell shock capturing for discontinuous Galerkin methods," AIAA Paper 2006-112, 2006. https://doi.org/https://doi.org/10.2514/6.2006-112.