AIAA JOURNAL
Vol. 54, No. 12, December 2016

Improving High-Order Finite Element Approximation Through Geometrical Warping

Devina P. Sanjaya* and Krzysztof J. Fidkowski†
University of Michigan, Ann Arbor, Michigan 48109

DOI: 10.2514/1.J055071

Polynomial basis functions are the ubiquitous workhorse of high-order finite element methods, but their generality comes at a price of high computational cost and fragility in the face of underresolution. In this paper, a method is presented for constructing a posteriori tailored, generally nonpolynomial, basis functions for approximating a solution and computing outputs of a system of equations. This method is similar to solution-based adaptation, in which elements of the computational mesh are sized and oriented based on characteristics of the solution. The method takes advantage of existing infrastructure in high-order methods: the reference-to-global mapping used in constructing curved elements. By optimizing this mapping, elements are warped to make them ideally suited for representing a target solution or computing a scalar output from the solution. Guidelines on generating a good initial guess and choosing a generalized set of optimization parameters are provided to minimize tuning time and to introduce automation into the process. For scalar advection–diffusion and Navier–Stokes problems, it is shown that warped elements can offer significant accuracy benefits without increasing the degrees of freedom in the system.

Nomenclature

c	=	chord
c_e	=	constraint on element e
d	=	spatial dimension
$\boldsymbol{F}(\boldsymbol{u})$	=	convective flux
f^{adapt}	=	target fraction of elements with largest error indicator
$\boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u})$	=	viscous flux
$\boldsymbol{H}$	=	total flux
$\mathcal{J}$	=	scalar output of interest
$\underline{\boldsymbol{J}}$	=	reference-to-global mapping Jacobian
$\underline{\boldsymbol{K}}$	=	viscous diffusivity tensor
M	=	Mach number
N_e	=	number of elements
N_g	=	Gauss points
N_p	=	number of basis functions per element
N_q	=	total number of degrees of freedom in an element
Pe	=	Peclet number
Pr	=	Prandtl number
$\mathcal{P}^p$	=	polynomials of order p on an element
p	=	solution approximation order
q	=	geometry approximation order
$\boldsymbol{R}$	=	residual vector
Re	=	Reynolds number
$\mathcal{R}_h$	=	semilinear weak form
S	=	source term
s	=	state rank
T_h	=	set of elements in a nonoverlapping tessellation of the domain Ω
$\boldsymbol{U}_{en}$	=	coefficients for the nth basis function on element e
$\boldsymbol{u}$	=	state vector
$\boldsymbol{u}_h(\boldsymbol{x})\|_{\Omega_e}$	=	approximated state on element e
u^{exact}	=	exact solution
$u^{\text{manufactured}}$	=	manufactured solution
$\|\boldsymbol{V}\|$	=	velocity magnitude
$\mathcal{V}_h$	=	solution approximation space
V_0	=	initial element volume
$\boldsymbol{v}_h$	=	test function
w_g	=	weights at Gauss points
$\boldsymbol{x}$	=	geometry node coordinates in global space
α	=	angle of attack
$\boldsymbol{\delta}$	=	design variables
$\delta\mathcal{J}$	=	output error
$\epsilon_e^{\mathcal{J}}$	=	least-squares output error estimate on element e
ϵ_e^{LS}	=	least-squares error on element e
ϵ_{e0}	=	initial error on element e
η_V	=	prescribed nondimensional minimum determinant of Jacobian as fraction of element volume
μ_b	=	nondimensional barrier penalty factor
ν	=	kinematic viscosity
$\boldsymbol{\xi}$	=	geometry node coordinates in reference space
π_e	=	unconstrained optimization problem on element e
$\boldsymbol{\Phi}$	=	Lagrange basis functions based on displaced reference-space coordinates
ϕ	=	Lagrange basis functions
$\boldsymbol{\Psi}$	=	discrete adjoint solution

I. Introduction

HIGH-ORDER finite element methods, such as discontinuous Galerkin, offer accuracy benefits for many problems due to their reliance on high-order polynomial functions for representing the solution. Polynomials have excellent approximation properties, at least for smooth functions; and when accuracy is important, high-order approximation can beat low-order approximation in terms of degrees of freedom (DOFs) and even computational cost [1].

However, polynomial approximation is not always the best choice. High-accuracy requirements may necessitate very high polynomial orders that make solutions computationally intractable. In addition, certain features of the solution may be too underresolved on a given mesh for robust polynomial approximation. One remedy is adaptation (in particular, of the hp variety), in which mesh elements h are refined where high order is not advantageous [2–5]. Another option is test space optimization, the goal of which is typically to improve accuracy in a certain error norm or an output of interest [6–9]. Yet another possibility is to tailor finite element basis functions to the problem at hand. This idea has been recognized in numerous previous works, including the partition of unity method

Presented as Paper 2015-2605 at the 22nd AIAA Computational Fluid Dynamics Conference, Dallas, TX, 21–26 January 2015; received 28 January 2016; revision received 18 May 2016; accepted for publication 10 July 2016; published online 8 September 2016.

*Ph.D. Candidate, Department of Aerospace Engineering. Member AIAA.
†Associate Professor, Department of Aerospace Engineering. Senior Member AIAA.

[10], the extended finite element method [11], isogeometric analysis [12], and the discontinuous enrichment method [13]. Tailoring basis functions a priori is possible for some problems but it is hard in general, especially for complex flows in which the locations of features such as shocks and shear layers are not known ahead of time. On the other hand, tailoring basis functions a posteriori is a more robust alternative: one that we pursue in this work.

Our proposed approach uses basis functions parametrized by reference-to-global mappings used in the definition of curved elements. Indeed, many high-order methods already do not employ global-space high-order polynomials. Polynomials are used on a reference element, but the reference-to-global mapping distorts the approximation. This distortion was recognized previously and an approach was designed to correct it via a linear shadow map [14]. In this work, we take an alternate position and embrace the distortion produced by the mapping. That is, we attempt to tune the reference-to-global coordinate maps in a mesh to produce elements that are customized for representing a particular solution. We refer to this process as element warping because we are changing the (internal) shape of an element. The mapped basis functions will no longer constitute a complete polynomial set in global space; instead, for a given solution order p, the basis functions will contain certain high-order modes that enable accurate approximation of the target solution. This is not the only way to create parametrized basis functions, but it is one that uses machinery (i.e., curved-element mappings) already available in many high-order codes. This proposed method is similar to r-refinement methods [15–17], which redistribute mesh points to minimize certain error measures, often dynamically in time [18–21]. However, unlike r refinement, our proposed method moves the high-order geometry nodes within an element to an optimal location to warp the element while keeping the mesh elements fixed; changing the element shape is left to a separate h-adaptation step. We will see later that moving the high-order geometry nodes is equivalent to tailoring the basis functions. Our eventual goal is to fully integrate this method with hp output-based adaptation to create customized approximation spaces geared for predicting a desired output to high accuracy.

The outline for the remainder of this paper is as follows. Section II reviews the discontinuous Galerkin (DG) finite element discretization, with particular emphasis on solution approximation and curved elements. Section III introduces the idea of intentionally curving the interior structure of an element to improve approximation for a given solution order, and Sec. IV presents our approach for optimizing the associated reference-to-global coordinate transformation. Section V shows results obtained from this method, and Sec. VI presents conclusions and plans for further work.

II. Discontinuous Finite Element Discretization

The idea of warping an element to improve its approximation properties can be applied to any method that supports high-order curved elements. We focus on the discontinuous Galerkin method because we have experience with it and because it is a relatively mature high-order method suitable for convection-dominated flows that are prevalent in aerospace engineering: our target application. In this section, we present the discretization, with particular attention to the curved-element treatment.

A. Conservation Law

Consider a conservation law given by the partial differential equation (PDE)

$$\partial_t \boldsymbol{u} + \nabla \cdot \boldsymbol{H}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{0} \tag{1}$$

where $\boldsymbol{u} \in \mathbb{R}^s$ is the state vector, $\boldsymbol{H} \in \mathbb{R}^{d\times s}$ is the total flux, s is the state rank, and d is the spatial dimension. We decompose the flux into convective and diffusive parts via $\boldsymbol{H} = \boldsymbol{F}(\boldsymbol{u}) + \boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u})$, where $\boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u}) = -\underline{\boldsymbol{K}}(\boldsymbol{u})\nabla \boldsymbol{u}$ is the viscous flux and $\underline{\boldsymbol{K}} \in \mathbb{R}^{d^2\times s^2}$ is the viscous diffusivity tensor.

B. Discretization

The DG [14,22,23], as a finite element method, approximates the state $\boldsymbol{u}$ in functional form using linear combinations of basis functions on each element. No continuity constraints are imposed on the approximations on adjacent elements. Denoting by T_h the set of N_e elements in a nonoverlapping tessellation of the domain Ω, the state on element e, Ω_e, is approximated as

$$\boldsymbol{u}_h(\boldsymbol{x})|_{\Omega_e} = \sum_{n=1}^{N_p} \boldsymbol{U}_{en}\phi_{en}^{\text{glob}}(\boldsymbol{x}) \tag{2}$$

In this equation, N_p is the number of basis functions per element and $\boldsymbol{U}_{en}$ is the vector of s coefficients for the nth basis function on element e: $\phi_{en}^{\text{glob}}(\boldsymbol{x})$. Formally, we write $\boldsymbol{u}_h \in \mathcal{V}_h = [\mathcal{V}_h]^s$, where, if the elements are not curved,

$$\mathcal{V}_h = \{u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \ \forall\ \Omega_e \in T_h\}$$

and $\mathcal{P}^p$ denotes polynomials of order p on the element. A caveat here is that, for elements that are curved, the polynomial approximation is usually performed on a master reference element so that, following the reference-to-global mapping, the state approximation on curved elements is not strictly of order p. We take advantage of this observation when we optimize the curved-element shape to yield better approximation properties compared to polynomials.

We obtain a weak form of Eq. (1) by multiplying the PDE by test functions $\boldsymbol{v}_h \in \mathcal{V}_h$ and integrating by parts to couple elements via interface fluxes. We use the Roe scheme [24] for convective fluxes and the second form of Bassi and Rebay [25] for viscous fluxes, yielding the final semilinear weak form

$$\mathcal{R}_h(\boldsymbol{u}_h, \boldsymbol{v}_h) = \sum_{e=1}^{N_e} \mathcal{R}_h(\boldsymbol{u}_h, \boldsymbol{v}_h|_{\Omega_e}) = 0, \quad \forall\ \boldsymbol{v}_h \in \mathcal{V}_h \tag{3}$$

where $\mathcal{R}_h(\boldsymbol{u}_h, \boldsymbol{v}_h|_{\Omega_e})$ is the weak form on element e. Details on the terms included in the weak form can be found in the literature [14]. Using the trial basis functions as test functions yields the final discrete system $\boldsymbol{R}(\boldsymbol{U}) = \boldsymbol{0}$, where $\boldsymbol{U}$ is the vector of unknown basis function coefficients and $\boldsymbol{R}$ is the vector of residuals, i.e., the discrete equations.

C. Curved Elements

An element is geometrically linear if its shape is defined by the location of its primary vertices. For example, in two dimensions, three points define a triangle and four points define a quadrilateral. In between the vertices, the geometry is interpolated (bi-/tri)linearly. Such elements are simple to work with but, when used on curved domain boundaries, they do not approximate the boundary well enough for use with high-order solution approximation in the DG [26–28]. A remedy is to curve the elements by equipping each element with additional geometry information, typically in the form of extra high-order geometry nodes.

A standard and relatively simple way to curve elements is to use a polynomial mapping from the reference element to the global element, as illustrated in Fig. 1. The formula for the mapping function is given in the figure, where q is the order of this polynomial, $N_q = (q+1)(q+2)/2$ is the total number of degrees of freedom in the mapping, $\boldsymbol{\xi} = [\xi, \eta]^T$ is the coordinate in reference space, and $\boldsymbol{x} = [x, y]^T$ is the coordinate in global space. Using Lagrange basis functions, $\phi_i^{\text{Lag}}(\boldsymbol{\xi})$, in the mapping allows for an intuitive specification of the high-order element: the coordinates of the N_q nodes $\boldsymbol{x}_i$ fully define the mapping function, and $\boldsymbol{x}(\boldsymbol{\xi}_i) = \boldsymbol{x}_i$.

The coordinates $\boldsymbol{x}_i$ should be chosen consistently with the corresponding reference-space nodes $\boldsymbol{\xi}_i$, which are equally spaced on the reference element. For example, in Fig. 1, $\boldsymbol{\xi}_6$ is the centroid of the reference triangle, so $\boldsymbol{x}_6$ should be located somewhere in the middle of the curved element. On edges/faces that are on domain boundaries, these nodes are typically on the geometry. However, these

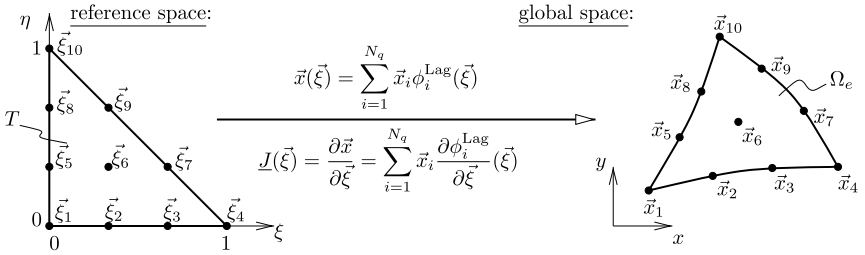


Fig. 1 Example of a $q = 3$ mapping from a reference triangle to a curved element in global space.

requirements do not pin down their locations, and heuristics or quality metrics such as maximizing the Jacobian determinant are often used in high-order node placement. In the next section, we discuss another choice: a figure of merit based on accurate solution approximation.

III. Warping High-Order Curved Elements

Curved elements are primarily used on domain boundaries to accurately define a geometry for use with high-order solution approximation [29]. For highly anisotropic boundary-layer meshes, curved elements are generally also needed inside the domain to prevent elements from self-intersecting and creating negative volumes [30,31]. Such curving is performed out of necessity in creating a valid mesh, driven ultimately by geometry representation requirements on the domain boundary. Curved elements do add computational expense (e.g., through element-specific mass matrices), but this cost can be mitigated by using the determinant of the mapping Jacobian matrix to scale basis functions [32].

Typically, not much attention is paid to the precise location of the high-order nodes, with the exception of those that have to lie on the boundary. Instead, heuristics often dictate locations that in some sense maximize the validity of the element, i.e., smoothly varying coordinates with no clustering of nodes. In this section, we present the idea of deliberately warping an element by moving high-order nodes to possibly clustered locations to optimize solution approximation.

Figure 2 illustrates the concept of element warping. Of interest are the locations of a curved element's geometry nodes, which dictate the mapping from reference space to global space. By moving these nodes, we can change the behavior of an approximated function (i.e., the state) in global space [33]. For example, consider a function that is a linear polynomial in the reference-space coordinates. This is what we typically refer to as a $p = 1$ solution approximation, since basis functions are most easily defined in reference space. If an element is geometrically linear, so that the reference-to-global mapping is affine, the $p = 1$ function remains linear in global space. However, if the element is curved, the mapped function will not necessarily remain linear in the global coordinates. Figure 2 illustrates this schematically for a $q = 3$ quadrilateral in which the middle nodes are placed close to each other so that a $p = 1$ function in reference space develops a shear-layer type of structure in global space.

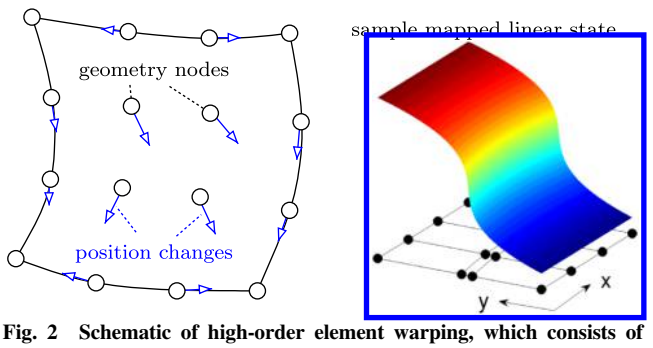


Fig. 2 Schematic of high-order element warping, which consists of intentional placement of high-order interior nodes to improve an element's approximation power for a particular solution.

In general, for arbitrary curved elements, a function that is an order p polynomial in reference space does not remain an order p polynomial, or even a polynomial at all, in the global-space coordinates. Specifically, a polynomial basis function in reference space $\phi^{\text{ref}}(\boldsymbol{\xi})$ maps to a global basis function according to

$$\phi^{\text{glob}}(\boldsymbol{x}(\boldsymbol{\xi})) = \phi^{\text{ref}}(\boldsymbol{\xi})$$

where $\boldsymbol{x}(\boldsymbol{\xi})$ is the geometry mapping given in Fig. 1. Moving an element's high-order geometry nodes changes this mapping and gives us control over the appearance of the high-order basis functions. Our goal is to optimize these global basis functions for the approximation of a particular solution, and we describe this optimization in the next section. Before moving on, however, we note that we are effectively working with a parametrized set of basis functions, where the parameters are the high-order geometry nodes. For a large value of q, we have many parameters, and we expect to be able to design custom basis functions that will allow us to accurately represent a solution even with low-order p. We expect increasing q to be computationally more desirable compared to increasing p, since the size of the system of equations is independent of q. Here, it is important to note that p and q are not interchangeable. A sufficient order p polynomial is still necessary for convergence and for obtaining significant benefits from increasing q.

IV. Warp Optimization

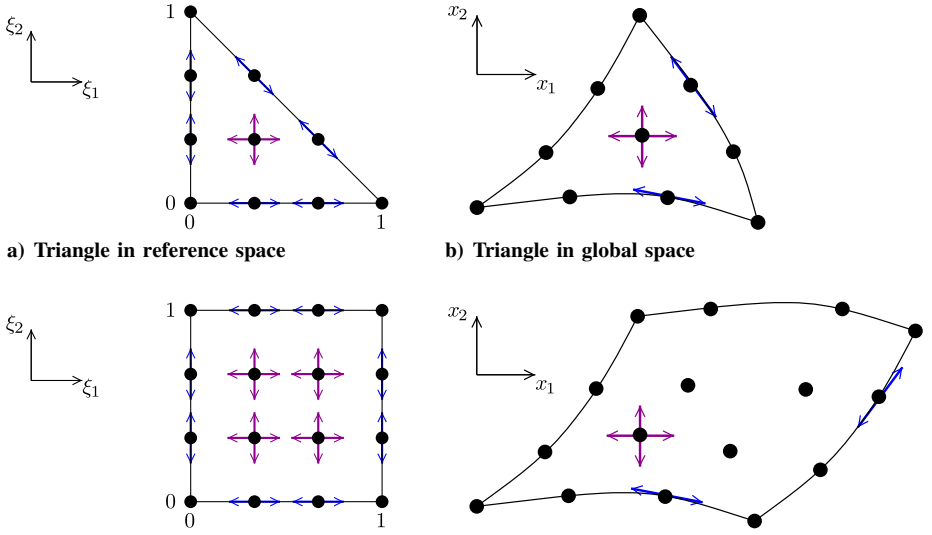
In the previous section, we introduced the idea that warping an element can change its approximation capabilities. In this section, we describe our approach to optimize the warp of an element by moving its high-order geometry nodes to optimal locations.

A. Design Variables

To keep computational costs low, we presently make the optimization problems local to each element. To minimize the influence of one element's optimization on its neighbor elements, we constrain the movement of the high-order geometry nodes so as not to affect the element shape (much). Thus, we do not move an element's primary vertices (three for a triangle) and we do not move edge nodes perpendicular to the edge.

For optimization, we need to choose the design variables. The global coordinates $\boldsymbol{x}_i$ of the mapped nodes are an obvious choice, but they are not ideal because they allow for arbitrary deformation. We would still have to impose the constraints that, for example, edge nodes move only along the edge, and this is hard to do in global space for curved elements. Instead, we turn to the reference element: we hold the global nodes $\boldsymbol{x}_i$ fixed, but we vary/optimize the reference-space coordinates $\boldsymbol{\xi}_i$ corresponding to these nodes. Normally, using an equally spaced nodal Lagrange basis for the reference-to-global mapping, the $\boldsymbol{\xi}_i$ are just evenly distributed on the reference element, with horizontal/vertical spacing of $1/(q+1)$. During optimization, we change the positions of the $\boldsymbol{\xi}_i$ in reference space, where imposing the edge motion constraint is trivial. As these $\boldsymbol{\xi}_i$ still map to the fixed $\boldsymbol{x}_i$, the element must warp. Figure 3 illustrates the allowable motions of nodes in $q = 3$ triangles and quadrilaterals.

The design variables are the allowable displacements of each $\boldsymbol{\xi}_i$ in reference space. Call $\boldsymbol{\delta}$ the vector of allowable displacements. The



a) Triangle in reference space b) Triangle in global space

c) Quadrilateral in reference space d) Quadrilateral in global space

Fig. 3 Example of allowable node motions for triangular and quadrilateral $q = 3$ elements.

size of this vector is $q^2 - 1$ for triangles and $2(q^2 - 1)$ for quadrilaterals. By the end of optimization, and during it for convenience of code reuse when calling certain functions, we must express the shape of the element using the standard equally spaced Lagrange basis. We do this by solving the following linear system:

$$\mathbf{\Phi} \boldsymbol{x}^{\text{new}} = \boldsymbol{x}^{\text{old}}$$

where $\mathbf{\Phi}$ is an $N_q \times N_q$ matrix with entries $\Phi_{ij} = \phi_j^{\text{Lag}}(\boldsymbol{\xi}_i')$, $\boldsymbol{\xi}_i'$ are the displaced reference-space coordinates of all the nodes, $\boldsymbol{x}^{\text{old}}$ is an $N_q \times d$ matrix of the original global-space node coordinates (one node per row), and $\boldsymbol{x}^{\text{new}}$ is an $N_q \times d$ matrix of the desired new global-space coordinates.

For multiple elements, we perform the optimization on each element independently and then average the (global-space) displacements of nodes that are shared between elements. In practical cases, optimization is not applied to every element but, rather, only to those elements with the largest errors.

B. Objective Function

Our goal here is to create a metric for measuring an element's approximation power, for use in optimization and as an error indicator telling us which elements in a mesh need to be warped. We consider the following two objective functions.

1. *Least-Squares Error*

Suppose that the exact solution $u^{\text{exact}}(\boldsymbol{x})$ is known. This is the case when testing with manufactured solutions; although, for practical cases, we could consider a solution or reconstruction in a higher-order space (e.g., $p + 1$). For a scalar problem ($s = 1$), the least-squares error ε_e on element Ω_e is defined via

$$\begin{aligned}(\varepsilon_e^{\text{LS}})^2 &= \min_{u_h \in \mathcal{V}_h} \int_{\Omega_e} [u_h(\boldsymbol{x}) - u^{\text{exact}}(\boldsymbol{x})]^2 \, \mathrm{d}A \\ &= \min_{u_h \in \mathcal{V}_h} \int_E |\underline{J}(\boldsymbol{\xi})| [u_h(\boldsymbol{\xi}) - u^{\text{exact}}(\boldsymbol{x}(\boldsymbol{\xi}))]^2 \, \mathrm{d}E \end{aligned} \tag{4}$$

where $\underline{J}(\boldsymbol{\xi})$ is the mapping Jacobian matrix, E is the reference-space element, and the minimization is taken over all possible u_h in the solution approximation space $\mathcal{V}_h$. The integral in reference space is evaluated using Gauss quadrature with N_g Gauss points $\boldsymbol{\xi}_g$ and weights w_g.

Note that integrating in reference space allows us to precompute and reuse evaluations of the basis functions at the quadrature points. Furthermore, negative Jacobian determinants encountered during integration indicate infeasible regions of the design space to the optimizer. Finally, for systems of equations, the least-squares error can be computed for each state component separately and summed if a single number is desired, though this introduces dependence on arbitrary variable scaling.

2. *Computing Outputs of a System*

In aerospace applications, we often deal with systems of equations ($s > 1$) and we generally only care about one or several outputs instead of the solution everywhere. In this case, reducing the state approximation error everywhere in the domain via the least-squares error metric can be inefficient because the state may not need to be accurate everywhere to predict accurate outputs. A more efficient approach, and one that naturally handles systems, is to use an output-based error measurement, as described in this section.

Output-based methods rely on the solution of an output adjoint, which acts as a weight on the residual to produce the error estimate and adaptive indicator [3]. The discrete adjoint solution $\mathbf{\Psi}$ satisfies

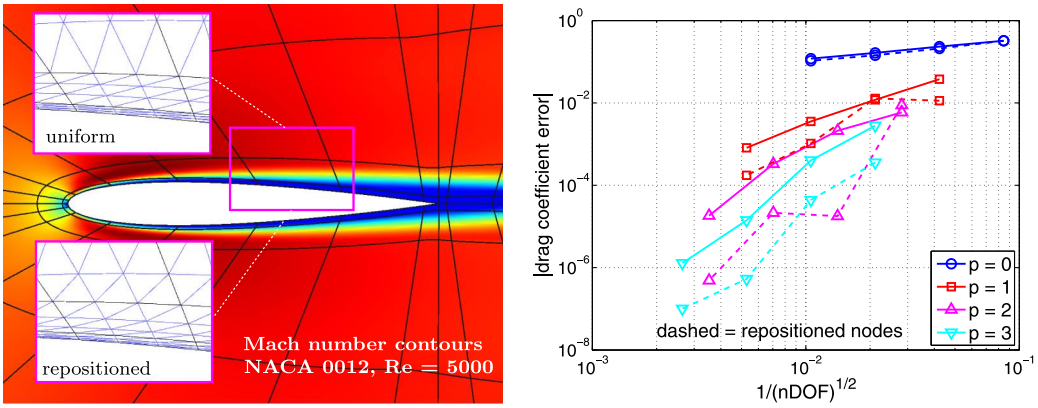
$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^T \mathbf{\Psi} + \frac{\partial \mathcal{J}^T}{\partial \boldsymbol{U}} = \mathbf{0} \tag{5}$$

where $\mathcal{J}$ is the scalar output of interest. Solving this equation yields coefficients of the adjoint solution, which can then be used to approximate the continuous adjoint $\boldsymbol{\psi}(\boldsymbol{x})$.

A simple approach for incorporating output-based adjoint information into the objective function is to modify the least-squares error estimate in Eq. (4) to use a weighted combination of primal and adjoint errors:

$$\varepsilon_e^{\mathcal{J}} = \frac{1}{2} \|\boldsymbol{R}\|_e^T \|\boldsymbol{\psi} - \boldsymbol{\psi}^{\text{fine}}\|_{\text{LS}} + \frac{1}{2} \|\boldsymbol{R}^{\psi}\|_e^T \|\boldsymbol{u} - \boldsymbol{u}^{\text{fine}}\|_{\text{LS}} \tag{6}$$

where $\boldsymbol{\psi}^{\text{fine}}$ and $\boldsymbol{u}^{\text{fine}}$ are approximate/reconstructed fine-space ($p + 1$) solutions, $\|\boldsymbol{R}\|_e$ is the fine-space primal residual norm on element e, and $\|\boldsymbol{R}^{\psi}\|_e$ is the fine-space adjoint residual norm on element e. The residuals are computed using primal and adjoint states before the optimization (i.e., $\boldsymbol{\delta} = \mathbf{0}$). Norms are taken separately for each equation for systems, and the subscript LS indicates the least-squares error; i.e., $\|\boldsymbol{\psi} - \boldsymbol{\psi}^{\text{fine}}\|_{\text{LS}}$ is the error between the "truth" (fine) adjoint and its projection into the order p approximation space of the current warped element. Several methods to compute fine-space adjoint were discussed in previous work [34–37].



a) Two types of curved elements b) Drag convergence for two mesh types

Fig. 4 Effect of interior node placement on the calculation of drag on an airfoil.

Equation (6) is motivated by the observation that low output error is achieved when both the primal and adjoint solutions are approximated well in an element [34], and the primal/ adjoint residuals indicate the relative importance of each and make the combination dimensionally consistent.

C. Constraint

As mentioned previously, we need all Jacobian determinants to be nonnegative to ensure that elements in the mesh are valid and to obtain a physical solution [38]. We find it more robust to go a step further and enforce a minimum Jacobian determinant over an element (measured at integration points). Thus, we impose the following constraint on element e:

$$c_e \equiv \frac{\min(|\underline{J}(\boldsymbol{\xi}_g)|)}{V_0} - \eta_V > 0 \tag{7}$$

where V_0 is the initial element volume, and η_V is the prescribed nondimensional minimum determinant of Jacobian as a fraction of the element volume. A natural question is then whether this constraint alone is enough to ensure that Jacobian determinants are nonnegative over the entire element. This constraint does not guarantee nonnegative Jacobian determinants over the entire element, but robustness improves with larger N_q and η_V.

D. Optimization Problem

Now, we can formulate our constrained optimization problem on an element as follows:

$$\text{minimize } \frac{\varepsilon_e(\boldsymbol{\delta})}{\varepsilon_{e0}} \quad \text{with respect to } \boldsymbol{\delta} \quad \text{subject to } c_e \tag{8}$$

where we explicitly indicate the dependence of the error on the design variables $\boldsymbol{\delta}$, and where $\varepsilon_{e0} = \varepsilon_e(\boldsymbol{\delta} = \mathbf{0})$ is the initial error on the element. We solve this constrained optimization problem via an interior penalty method by using an inverse barrier function with μ_b as the nondimensional barrier penalty factor. This turns Eq. (8) into the following unconstrained optimization problem on element e:

$$\text{minimize } \pi_e(\boldsymbol{\xi}(\boldsymbol{\delta}), \mu_b) = \frac{\varepsilon_e(\boldsymbol{\delta})}{\varepsilon_{e0}} + \mu_b \frac{V_0}{\min(|\underline{J}(\boldsymbol{\xi}_g)|) - \eta_V V_0} \tag{9}$$

$$\text{with respect to } \boldsymbol{\delta}$$

The solution to Eq. (9) approaches that to Eq. (8) as μ_b approaches zero. To keep the computational cost low, optimization is only performed on a fraction f^{adapt} of elements with the largest error indicator.

The optimization problem on each element is solved using a gradient-based method: the Broyden–Fletcher–Goldfarb–Shanno (BFGS) [39] algorithm with a backtracking line search. We treat the optimization problem locally, in that the optimization is performed on each element independently. When there are multiple elements, the (global-space) displacements of nodes that are shared between elements are averaged before moving all nodes to their new locations. Note that treating the optimization problem as a global problem would make it much more computationally expensive.

Although treating the optimization problem in element-local terms is computationally advantageous, it does require care when averaging to produce the global mesh. Due to the node averaging process, the new node locations are no longer optimal, but our assumption is that they are better than the original node locations, in the sense that these new locations improve the approximation power of the finite element method. However, to reduce the risk of obtaining an invalid mesh, we must avoid overoptimizing locally. Local overoptimization increases the probability of large node displacements that, upon averaging, may cause self-tangling (negative Jacobians), resulting in an invalid mesh. We avoid local overoptimization by setting μ_b to a sufficiently small constant to ensure that the constraint is active for all BFGS iterations, and by performing only a few BFGS iterations. Currently, all required gradients are calculated using a finite difference approximation.

V. Results

A. Boundary-Layer Approximation for a Laminar Airfoil

Before presenting the results of the node movement optimization, we offer a heuristic example of the potential benefit of moving high-order nodes. Consider a NACA 0012 airfoil in $M = 0.5$, $Re = 5000$ flow. In these conditions, a boundary layer (albeit not a very thin one) develops near the airfoil wall. Within this boundary layer, several flow properties change rapidly in the wall-normal direction, and an accurate representation of this boundary-layer flow is important for predicting the drag.

We investigate two types of $q = 3$ meshes for calculating drag at several different values of p. The first mesh (uniform) is one in which the high-order nodes are spaced uniformly in the elements. The

Table 1 Least-squares error for solution approximation on a single element with a manufactured solution

q	p	Unoptimized ε^{LS}	Optimized ε^{LS}
3	3	0.1747	0.0240
	4	0.0988	0.0110
4	3	0.1756	0.0136
	4	0.0985	0.0062

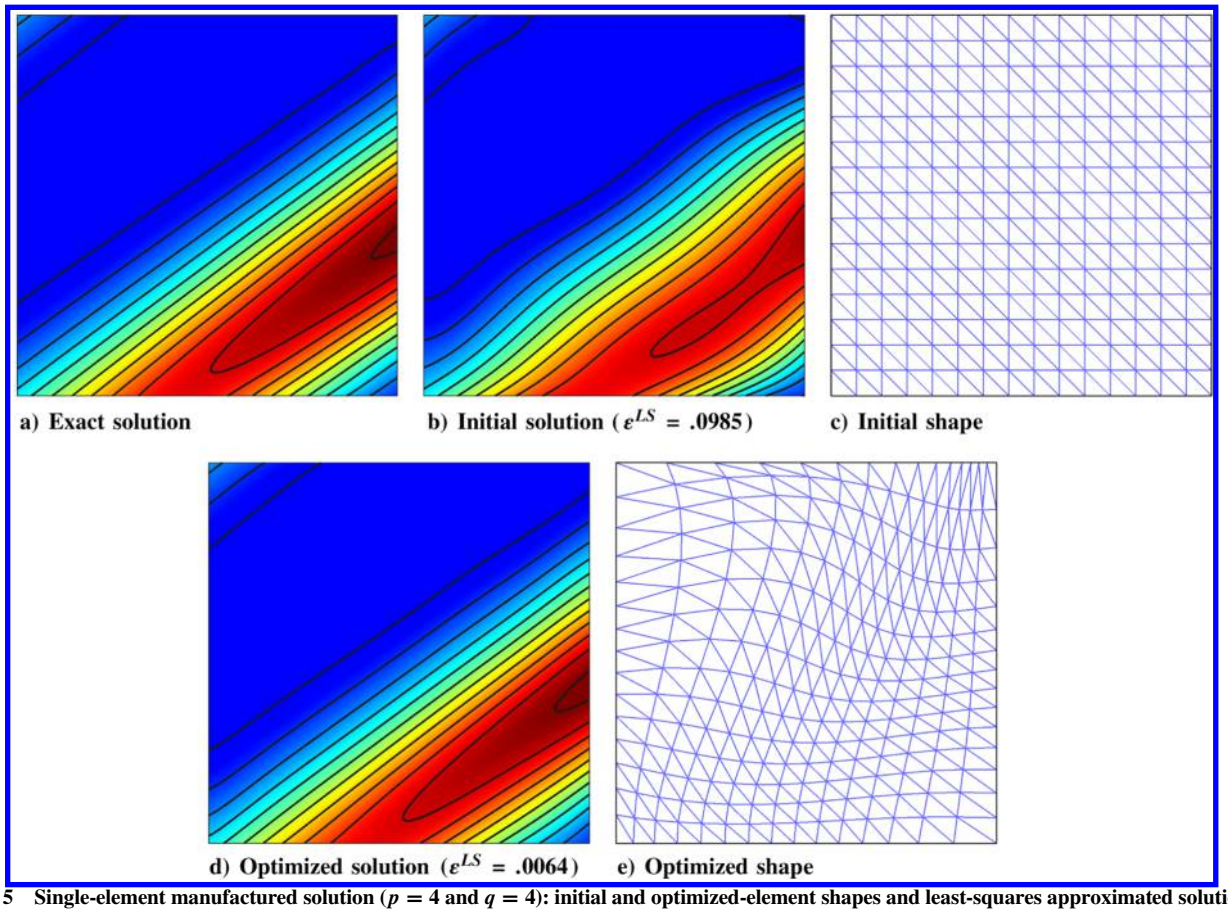


Fig. 5 Single-element manufactured solution ($p = 4$ and $q = 4$): initial and optimized-element shapes and least-squares approximated solutions.

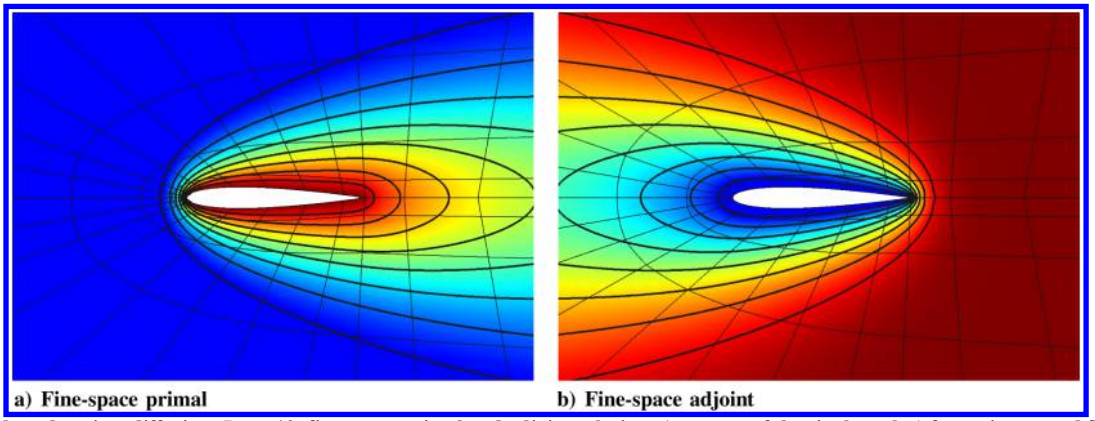


Fig. 6 Scalar advection–diffusion, $Pe = 10$: fine-space primal and adjoint solutions (contours of the single scalar) for an integrated flux output.

second mesh (repositioned) is one in which the high-order nodes are heuristically clustered toward the airfoil, which improves the ability of basis functions to accurately capture the rapid variation of flow quantities near the airfoil.

Figure 4 shows a sample flowfield and the convergence of drag for uniform refinements of the two families of meshes considered. For approximation orders $p = 1, 2, 3$, we see a benefit of using the meshes with the repositioned nodes. Specifically, the drag coefficient error drops by one or more orders of magnitude compared to the meshes with the uniformly distributed high-order geometry nodes, for the same computational cost. These results are for uniform refinements of a single guessed repositioning: we expect further improvements from an optimization algorithm.

B. Single-Element Optimization with a Manufactured Solution

We consider a two-dimensional diffusion equation with source on a $[0, 1]^2$ domain

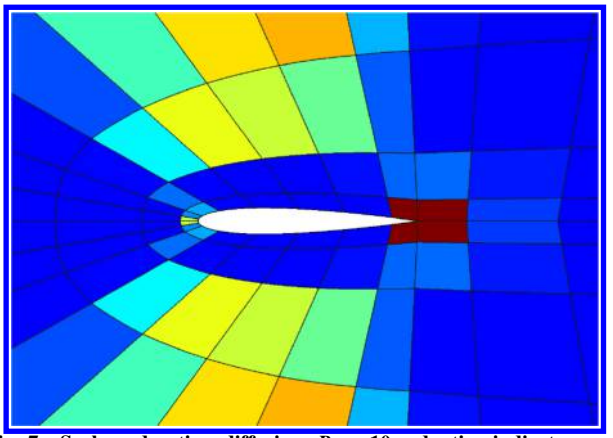


Fig. 7 Scalar advection–diffusion, $Pe = 10$: adaptive indicator on a quadrilateral mesh.

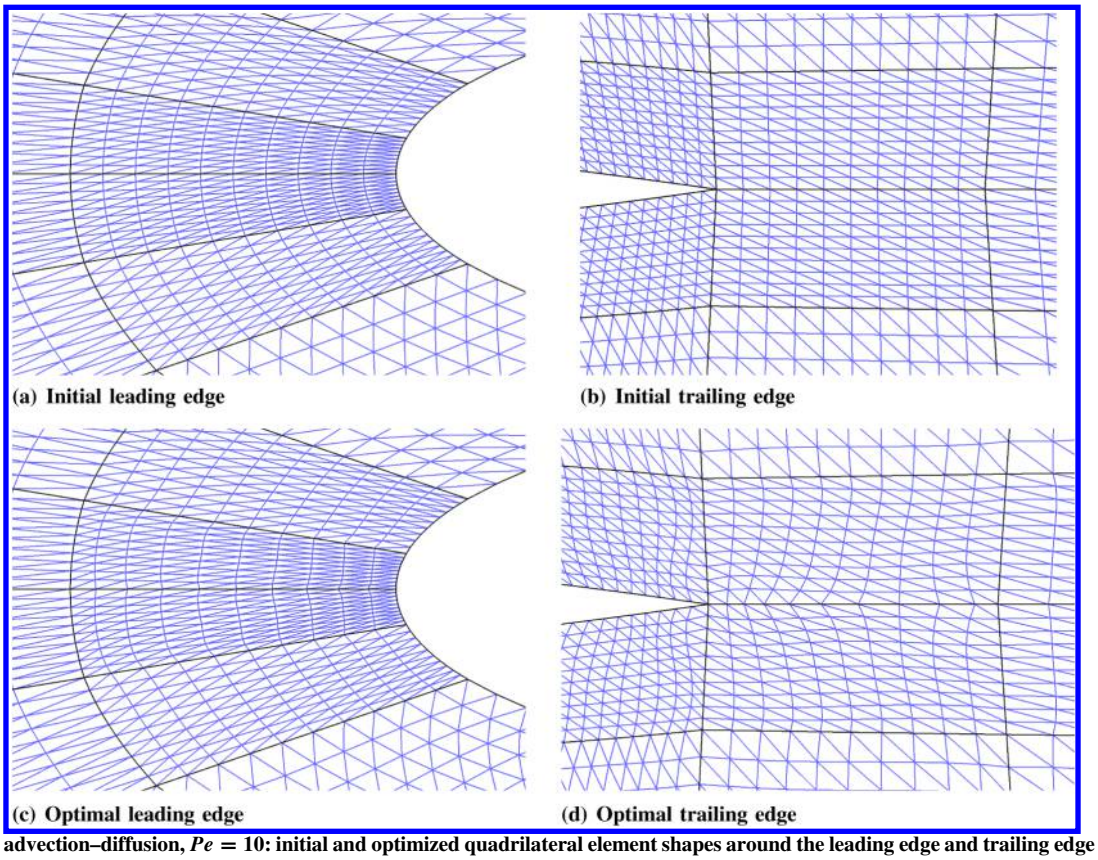


Fig. 8 Scalar advection–diffusion, $Pe = 10$: initial and optimized quadrilateral element shapes around the leading edge and trailing edge of the NACA 0012 airfoil.

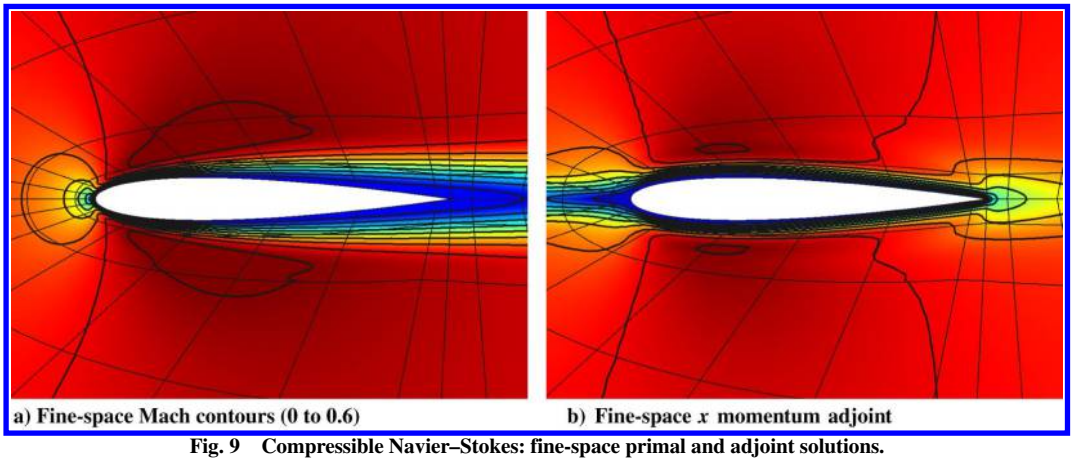


Fig. 9 Compressible Navier–Stokes: fine-space primal and adjoint solutions.

$$\nabla^2 u + S(\boldsymbol{x}) = 0$$

where $S(\boldsymbol{x})$ is a source term that makes the following function a manufactured solution:

$$u^{\text{manufactured}}(x, y) = \exp[a_1 \sin(a_2 x + a_3 y) + a_4 \cos(a_5 x + a_6 y)]$$

with

$$[a_1, a_2, a_3, a_4, a_5, a_6] = [1, 3, -4, .5, -2, 3.5]$$

We consider a single element with $q = [3, 4]$ and $p = [3, 4]$ solution approximation. We use least-squares error optimization with parameters $\eta_V = 0.1$ and $\mu_b = 0.1$. Table 1 shows that, without optimization, increasing p reduces the least-squares error by a factor of two, and optimization reduces the error further. For a given q, increasing p reduces the error by a factor of two, whereas for a given

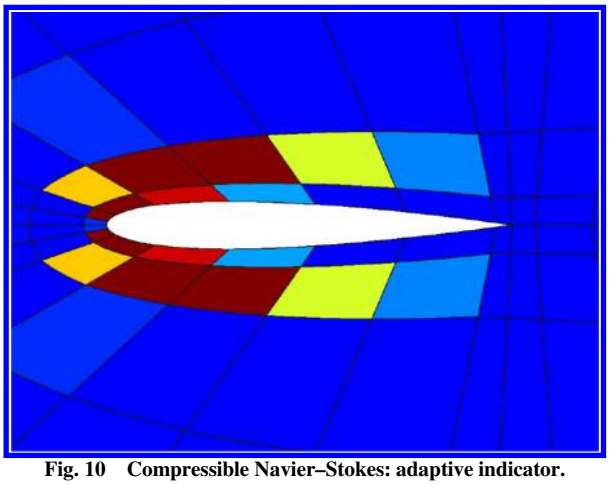


Fig. 10 Compressible Navier–Stokes: adaptive indicator.

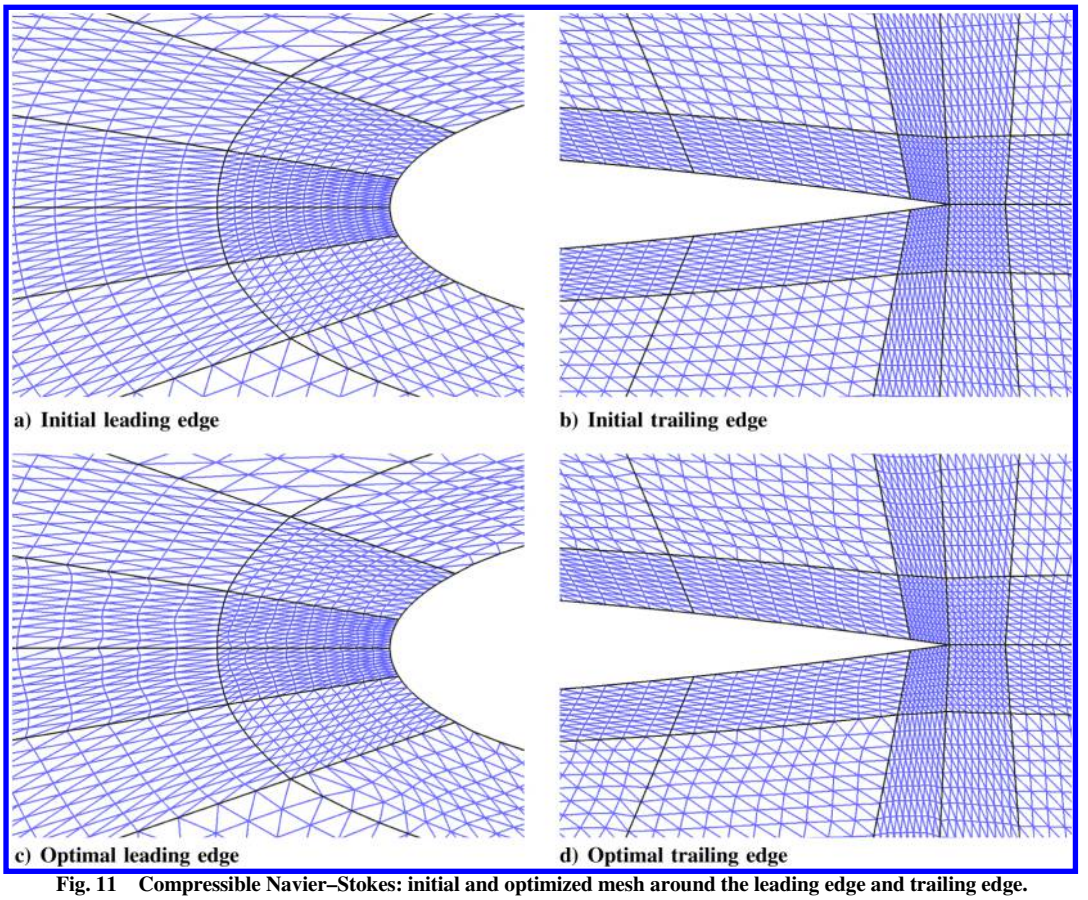


Fig. 11 Compressible Navier–Stokes: initial and optimized mesh around the leading edge and trailing edge.

Table 2 Optimization parameters used for Euler cases ($q = 3$)

	$p = 1$	$p = 2$	$p = 3$
Line search backtracking factor options	[0.004, 0.02, 0.1]	[0.005, 0.02, 0.1]	[0.006, 0.05, 0.8]
μ_b	0.04	0.4	4
f^{adapt}	1.0	1.0	1.0
η_V	0.2	0.15	0.1
BFGS iterations for SEQs 1 and 2	1	1	1
BFGS iterations for SEQ 3	3	3	3

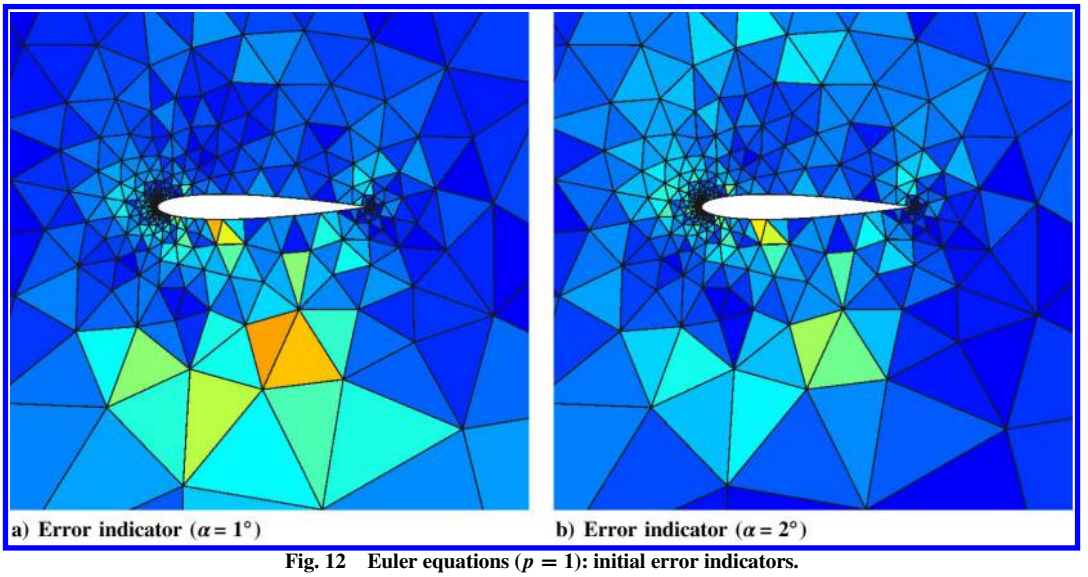


Fig. 12 Euler equations ($p = 1$): initial error indicators.

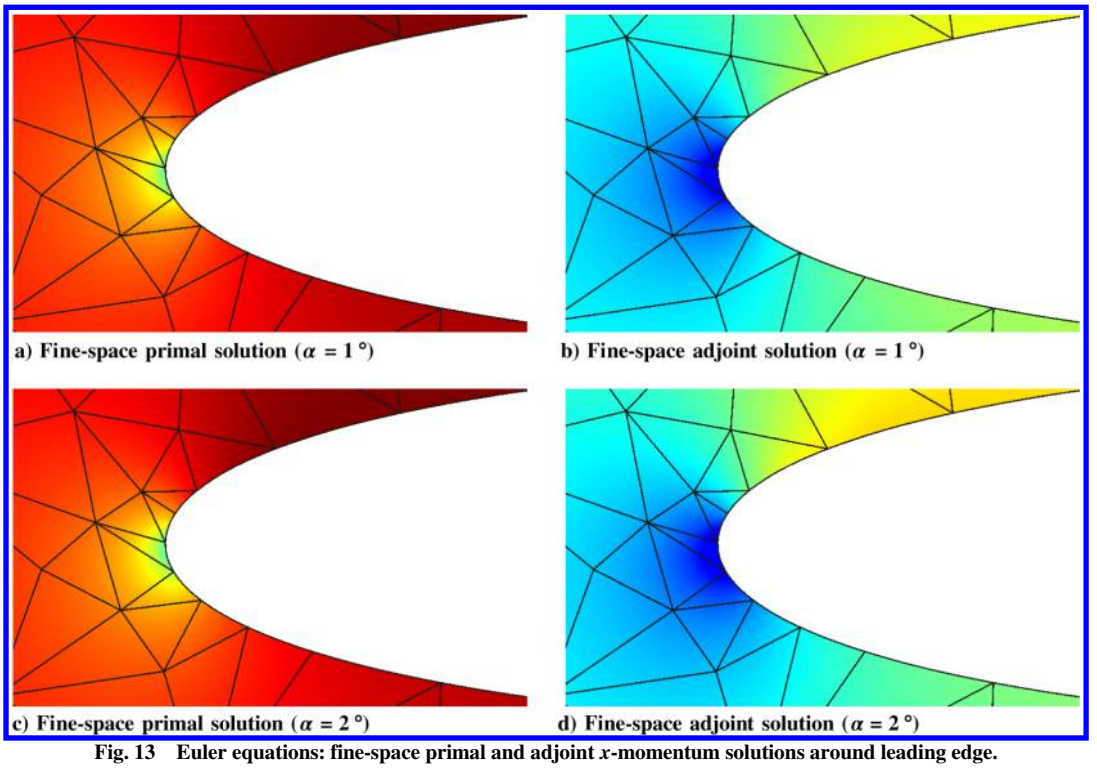
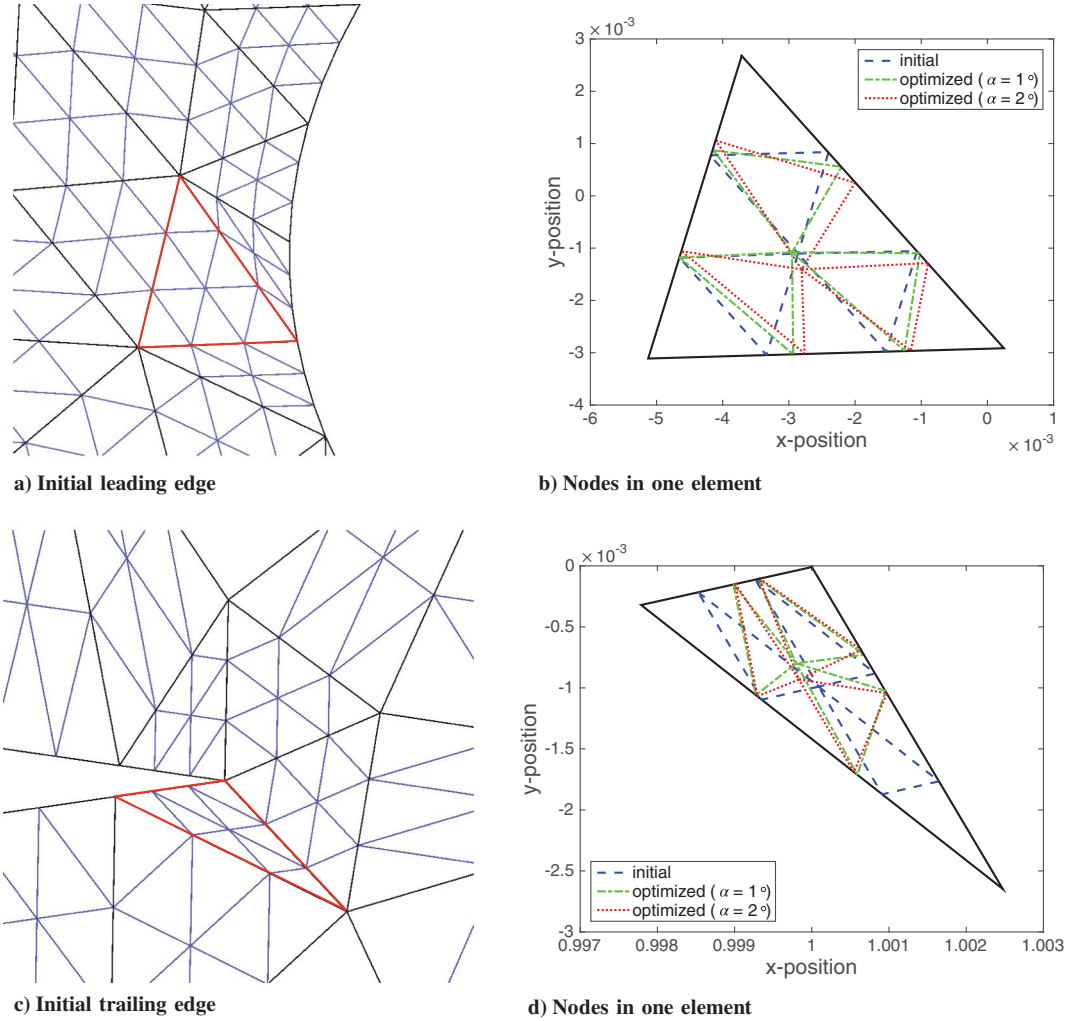


Fig. 13 Euler equations: fine-space primal and adjoint x-momentum solutions around leading edge.



a) Initial leading edge **b) Nodes in one element**

c) Initial trailing edge **d) Nodes in one element**

Fig. 14 Euler equations ($p = 1$ and $q = 3$): initial mesh and zoomed-in views of optimized elements.

Table 3 Euler equations: final error reduction factor based on $(p + \Delta p)$ fine-space solution with $\Delta p = 1$ and $q = 3$[a]

α, deg	Fine-space solution	$p = 1$	$p = 2$	$p = 3$
1	Exact	13.55	2.55	1.29
	Approximate	11.50	2.97	1.01
2	Exact	12.32	3.06	1.17
	Approximate	13.89	2.53	1.26

[a]The exact solution is computed by converging residuals to machine zero, whereas the approximate solution is computed using 5 iterations of element-block Jacobi smoothing.

p, increasing q reduces the error by an order of magnitude. However, this does not imply that increasing p and q are interchangeable. Notice that there is a coupling between p and q to obtain significant benefit from increasing q: with $p = 4$, we obtain more benefit from the warp optimization.

Figure 5 begins with the exact (manufactured solution) and the baseline $p = 4$ solution on the unoptimized, equal node-spacing element. The figure then shows the optimized-element solution, which is visually closer to the exact solution and has a much lower least-squares error. The triangular mesh plots in Fig. 5 show the initial and optimized shape of the single element obtained by subdividing the reference element into many $[2 \times (15 \times 15)]$ equally spaced triangles and plotting the mapped positions of these triangles in global space. We see that the optimized-element shape shows marked stretching and twisting in the reference-to-global mapping; it is this distortion that is responsible for the improved approximation ability of the element even when using the same $p = 4$ space on the reference element.

C. Multiple-Element Optimization for the Scalar Advection–Diffusion Equation

We now consider a steady, scalar advection–diffusion problem on a unit-chord ($c = 1$) NACA 0012 airfoil:

$$\nabla \cdot (\boldsymbol{V}u) - \nu \nabla^2 u = 0 \tag{10}$$

where $\boldsymbol{V} = (1, 0)$, and $Pe \equiv |\boldsymbol{V}|c/\nu = 10$. Dirichlet boundary conditions are applied: $u = 1$ on the airfoil surface and $u = 0$ on the far field. In this problem, we use the output-based optimization metric, where the output of interest $\mathcal{J}$ is the integrated flux of u through the airfoil surface. Figure 6 shows the fine-space primal and adjoint solutions for this output on a quadrilateral mesh.

We run our high-order node optimization algorithm for the quadrilateral mesh shown in Fig. 6 using $p = 2$, $q = 4$, $f^{\text{adapt}} = 0.2$, $\eta_V = 0.15$, and $\mu_b = 0.2$. Note that only 20% of the elements are optimized: the remaining 80% with low error are skipped. Following optimization of the targeted elements, we solve the problem again on the optimized mesh and compute the new output. Denote by $\delta\mathcal{J}$ the error in the output relative to a truth ($p = 4$) solution. We find that this output error reduces from $|\delta\mathcal{J}| = 9.0 \times 10^{-4}$ on the initial mesh to $|\delta\mathcal{J}| = 1.3 \times 10^{-5}$ on the mesh with optimized-element shapes.

Figure 7 shows the error indicator, i.e., $\varepsilon_e^{\mathcal{J}}$ for each element in the baseline mesh. We see that the area near the trailing edge of the airfoil has the largest error; thus, elements near the trailing edge will be targeted for warping. In addition, elements near the leading edge and away from the airfoil above and below it will be targeted.

Figure 8 shows the initial and optimized quadrilateral element shapes in the leading-edge and trailing-edge regions of the airfoil. We see pronounced distortion of the trailing-edge elements and some distortion of the leading-edge ones.

D. Multiple-Element Optimization for the Two-Dimensional Navier–Stokes Equations

Now, we consider a system of equations: steady, compressible Navier–Stokes:

$$\nabla \cdot \boldsymbol{F}(\boldsymbol{u}) + \nabla \cdot \boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{0} \tag{11}$$

where $\boldsymbol{F}$ and $\boldsymbol{G}$ are, respectively, the inviscid and viscous fluxes. The geometry is a NACA 0012 airfoil with a closed trailing edge, and the flow conditions are $M = 0.5$ and $Re = |\boldsymbol{V}|c/\nu = 1000$. A Prandtl number of $Pr = 0.71$ is used, and the boundary conditions are adiabatic walls on the airfoil and freestream conditions in the far field. The output of interest is the drag.

Figure 9 shows the primal Mach contours and the x-momentum component of the drag adjoint computed with high-order ($p = 4$) approximation on the baseline quadrilateral mesh. We see boundary-layer structures in both the primal and adjoint.

Following the baseline solution, we run an optimization using $p = 2$ solution approximation, $f^{\text{adapt}} = 0.5$, $\eta_V = 0.1$, and $\mu_b = 0.2$. Figure 10 shows the error indicator: regions targeted for adaptation include above and below the airfoil, whereas the trailing edge has a relatively low error in this case, possibly due to the already small elements there. Note that, in this case, we optimize half of the element shapes.

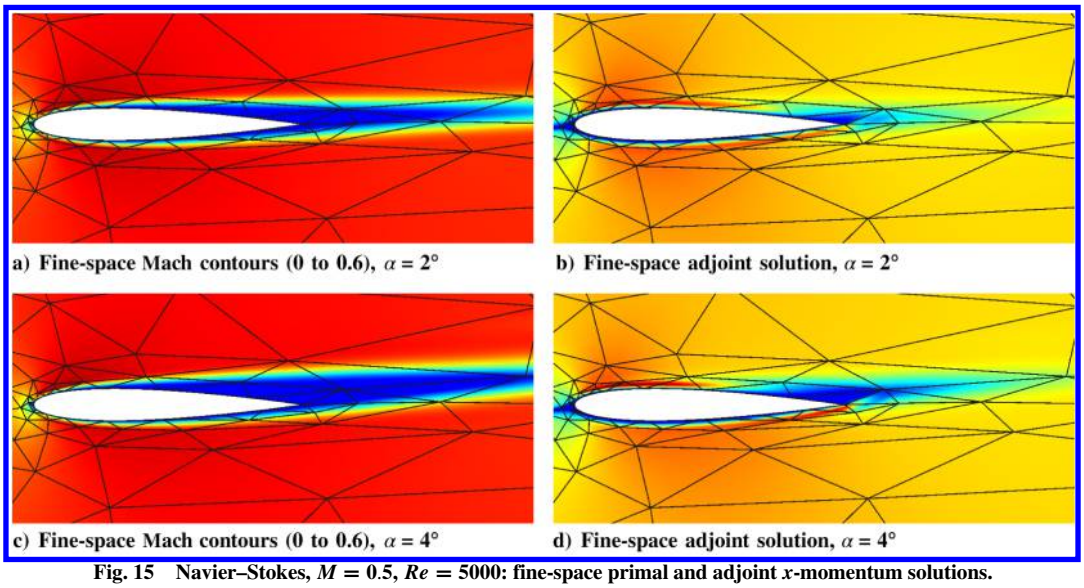
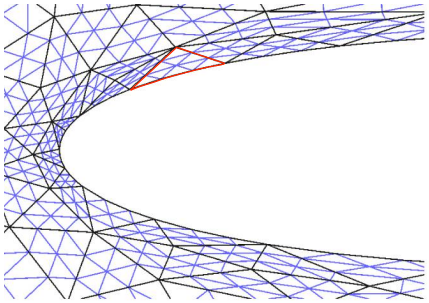


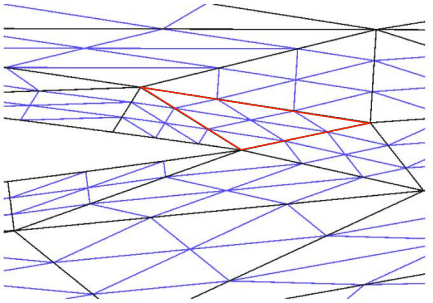
Fig. 15 Navier–Stokes, $M = 0.5$, $Re = 5000$: fine-space primal and adjoint x-momentum solutions.

Table 4 Optimization parameters used for viscous Navier–Stokes cases ($q = 3$)

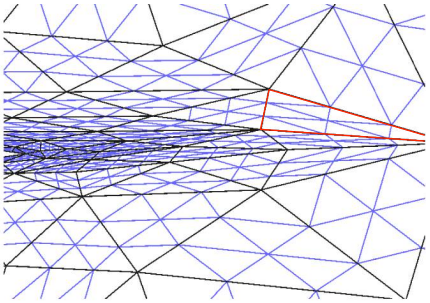
	$p = 1$	$p = 2$	$p = 3$
Line search backtracking factor options	[0.004, 0.02, 0.1]	[0.003, 0.08, 0.1]	[0.006, 0.05, 0.2]
μ_b	0.05	0.12	0.2
f^{adapt}	1.0	1.0	1.0
η_V	0.2	0.15	0.1
BFGS iterations for SEQs 1 and 2	1	1	1
BFGS iterations for SEQ 3	3	3	3



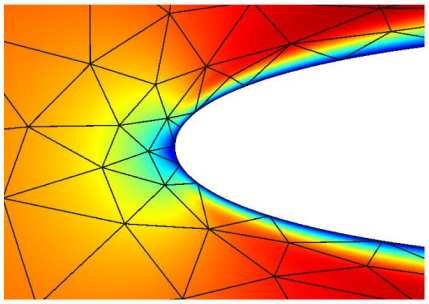
a) Initial leading edge



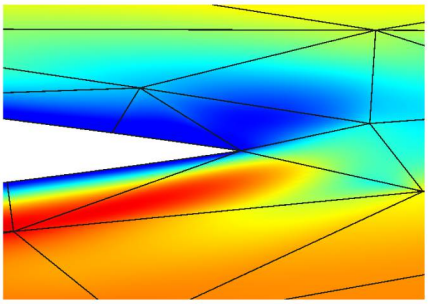
b) Initial trailing edge



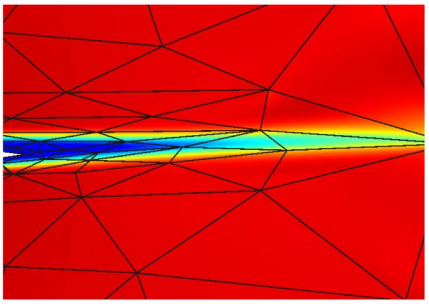
c) Initial wake



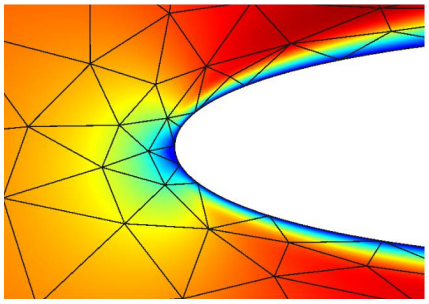
d) Initial $p = 3$ solution around leading edge ($\alpha = 2°$)



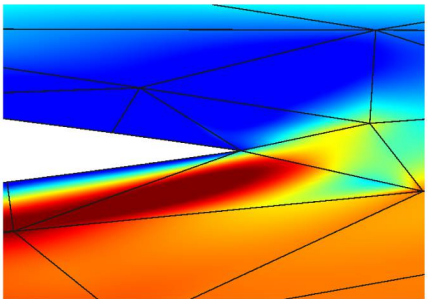
e) Initial $p = 3$ drag adjoint around trailing edge ($\alpha = 2°$)



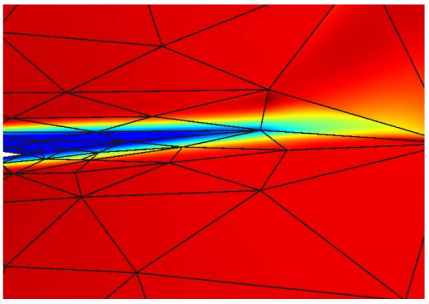
f) Initial $p = 3$ solution around wake ($\alpha = 2°$)



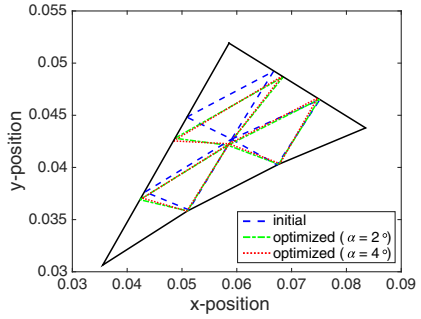
d) Initial $p = 3$ solution around leading edge ($\alpha = 4°$)



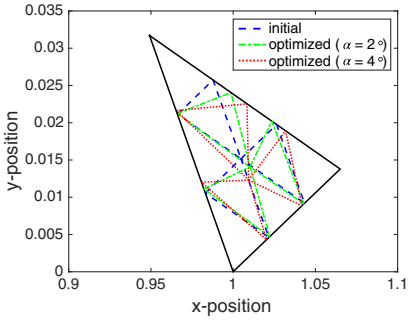
e) Initial $p = 3$ drag adjoint around trailing edge ($\alpha = 4°$)



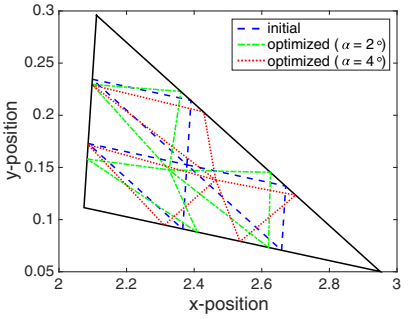
f) Initial $p = 3$ solution around wake ($\alpha = 4°$)



j) Element A

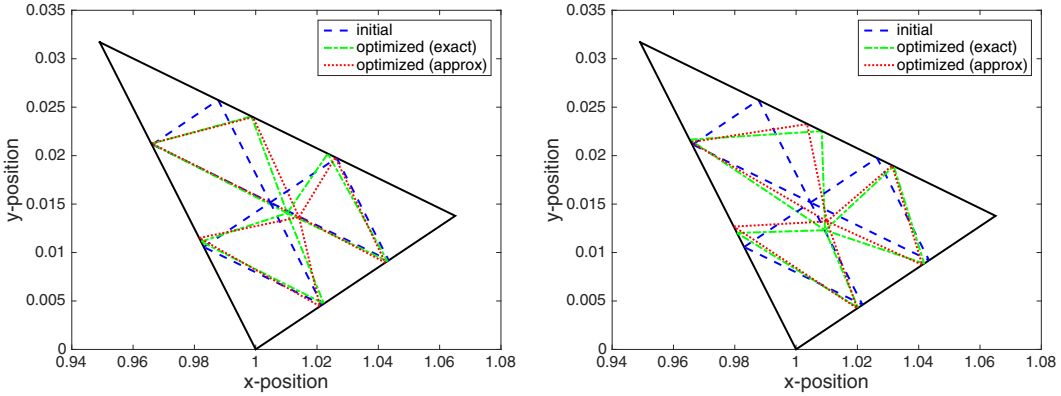


k) Element B



i) Element C

Fig. 16 Navier–Stokes, $M = 0.5$, $Re = 5000$ ($p = 2$ and $q = 3$): initial mesh, fine solution/drag adjoint, and sample optimized elements.



a) $\alpha = 2°$ **b)** $\alpha = 4°$

Fig. 17 Navier–Stokes, $M = 0.5$, $Re = 5000$ ($p = 2$ and $q = 3$): comparison of node movement of an element around the trailing edge.

Figure 11 shows the initial and optimized-element shapes around the leading edge and trailing edge of the airfoil. We see discernible and nonintuitive node movement, mostly near the leading edge. After solving again on the optimized mesh, we find that the error in the drag reduces from $|\delta\mathcal{J}| = 1.1 \times 10^{-3}$ on the baseline mesh to $|\delta\mathcal{J}| = 4.8 \times 10^{-4}$ on the mesh with optimized-element shapes. The reduction in output error is not as large in this case as in the previous scalar cases. A possible reason for this is that this example is a system of equations, and different components of the system may impose different demands on the optimal element shape. In addition, we are not allowing cancellation of errors between system components in our error indicator, since we compute the least-squares errors in Eq. (6) componentwise. Relaxing this conservative calculation may lead to larger error drops.

E. Multiple-Element Optimization for the Euler Equations

So far, we have seen the benefits of curved elements on coarse meshes for solution approximation and output computation. Now, we are in the position to address some questions pertaining to the nature of optimization algorithm, such as how to provide a good initial guess and how to automate the optimization process.

As expected, the optimization algorithm tends to perform better given a good initial guess. One good initial guess for the background mesh is an h-adapted mesh generated for a given p. To further improve the quality of initial mesh, three sequences of optimization are performed, in which the solution of each optimization provides the initial guess for the next optimization. For the first two optimizations, only one BFGS iteration is performed because the only purpose for these two optimizations is to provide a good initial guess for the third (last) optimization. For the last optimization, a larger number of BFGS iterations is performed to find the optimum mesh for the problem of interest. However, the iteration number is still kept relatively small to prevent local overoptimization.

In general, tuning of optimization parameters helps improve performance of optimization algorithm for a given problem. However, automation is necessary for robustness in an hp-adaptive setting. One aspect of the optimization algorithm that we have found to be most "tunable" has been the backtracking factor in the line search algorithm. To improve automation, we therefore provide several options of backtracking factors from which an element can choose the best-performing one. In addition, we found from experience that it is advantageous to increase μ_b and lower η_V as p increases. We found that having each element automatically pick a suitable backtracking factor gives us a good compromise between automation capability and optimization performance reduction due to generalization. Furthermore, to ensure nonnegative Jacobian determinants in the optimized mesh, we need to adjust how aggressive the optimizer can be. As p increases, the optimizer generally has access to a fairly accurate solution representation, which means that we can lower η_V as p increases because the chance of having negative Jacobian determinants is less. However, at the same time, we need to make sure that the optimizer is not so aggressive that it will overoptimize locally; thus, higher μ_b is needed for higher p. Based on these simple heuristics, we find a general set of optimization parameters that gives reasonable error reduction for a particular problem type and a given p. Table 2 shows the general setting of optimization parameters for Euler problems with $q = 3$. Note that we have decided to separate settings by physics of the problem, due to the different solution features observed with the different model equations.

Here, we consider steady, inviscid flow over a NACA 0012 airfoil:

$$\nabla \cdot \boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0} \tag{12}$$

where $\boldsymbol{F}$ are the inviscid fluxes. The boundary conditions are the inviscid wall on the airfoil and freestream conditions in the far field. The output of interest is the drag.

To test robustness of using the generalized optimization parameters, we analyze the flow at two different angles of attack: $\alpha = 1$ deg and $\alpha = 2$ deg, while keeping other settings the same. A relatively small change in angle of attack is chosen to ensure that the same initial mesh can be used as a good initial guess for both cases. The initial guess to the optimizer is an h-adapted mesh for $\alpha = 2$ deg. Figure 12 shows that the initial errors are slightly higher for the case with $\alpha = 1$ deg, as expected. One region of the flow with high error is the leading edge. Figure 13 shows the primal x-momentum solution and the x-momentum component of the drag adjoint around the leading edge computed with high order ($p = 4$). This confirms that the small change in angle of attack only changes the flow slightly; thus, the same initial mesh can be used as a good initial guess to the optimizer for both cases.

Figure 14 shows the initial mesh and a zoomed-in view of an optimized-element shape around the leading edge and trailing edge of the airfoil when an exact fine-space solution is used to drive the optimizer. Since the small difference in angle of attack only changes the flow slightly, most optimized elements have very similar shapes

Table 5 Navier–Stokes, $M = 0.5$, $Re = 5000$: final error reduction factor based on $(p + \Delta p)$ fine-space solution with $\Delta p = 1$ and $q = 3$[a]

α, deg	Fine-space solution	$p = 1$	$p = 2$	$p = 3$
2	Exact	3.60	60.34	7.40
	Approximate	3.36	2.02	2.68
4	Exact	43.87	3.04	3.24
	Approximate	22.44	1.43	28.57

[a]The exact fine-space solution is computed using GMRES, and the approximate fine-space solution is computed using an iterative method with element-block Jacobi smoothing.

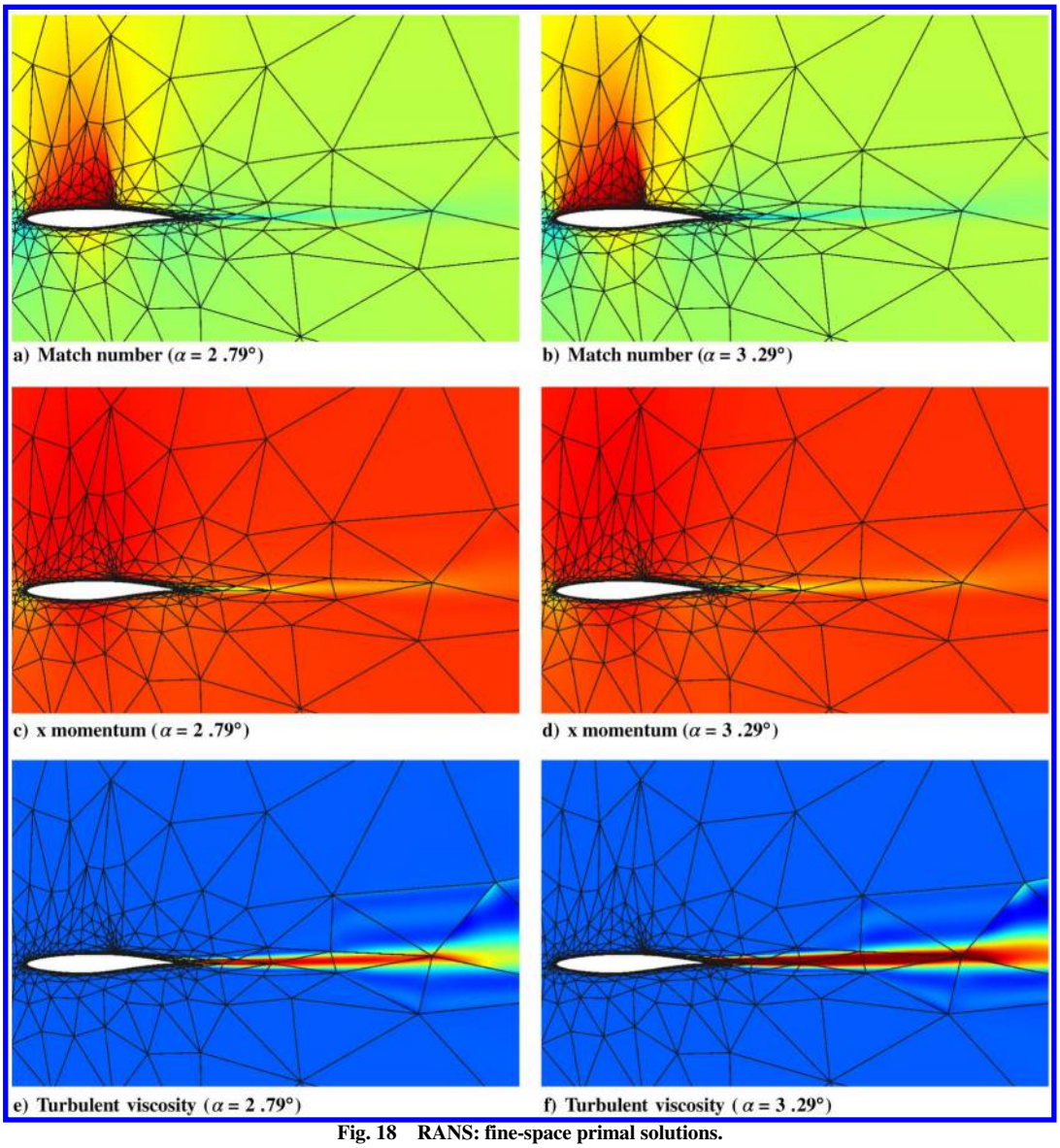


Fig. 18 RANS: fine-space primal solutions.

in both meshes. Hence, we provide a zoomed-in view of an optimized element here.

As in previous cases, we see discernible and nonintuitive node movement. In addition to using the exact fine-space solution, we also run the same cases using a fine-space surrogate to drive the optimizer, and the resulting optimized mesh is similar. The final error reduction factors for both sets of runs are shown in Table 3. For the case with $\alpha = 2$ deg, eight iterations of element-block Jacobi are used. For the case with $\alpha = 1$ deg, we increase the number of iterations to 50 because the starting mesh is not optimized for this case. Note that we obtain the most benefit of the optimization at the coarsest approximation order ($p = 1$) because, in this case, the solution is the least accurate and stands to gain the most from optimization.

F. Multiple-Element Optimization for the Viscous Navier–Stokes Equations

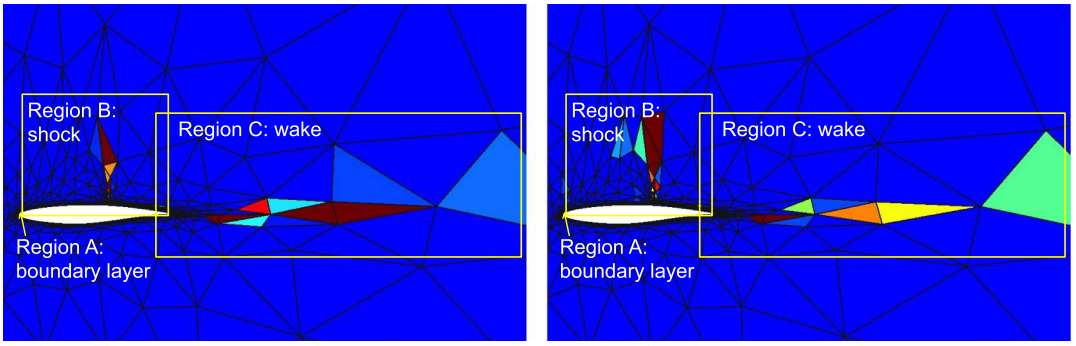
Next, we consider steady, viscous flow over a unit-chord ($c = 1$) NACA 0012 airfoil:

$$\nabla \cdot \boldsymbol{F}(\boldsymbol{u}) + \nabla \cdot \boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{0} \tag{13}$$

where $\boldsymbol{F}$ and $\boldsymbol{G}$ are, respectively, the inviscid and viscous fluxes. The flow conditions are $M = 0.5$ and $Re = |\boldsymbol{V}|c/\nu = 5000$. The

Table 6 Optimization parameters used in RANS cases ($q = 3$)

	$p = 1$	$p = 2$	$p = 3$
Line search backtracking factors	[0.002, 0.04, 0.9]	[0.008, 0.04, 0.5, 1.2]	[0.002, 0.06, 0.1, 1.5]
μ_b	6	1.2	0.6
f^{adapt}	0.25	0.25	0.25
η_V	0.3	0.15	0.1
BFGS iterations for SEQs 1 and 2	1	1	1
BFGS iterations for SEQ 3	2	5	5



a) $\alpha = 2.79°$ b) $\alpha = 3.29°$

Fig. 19 RANS ($p = 3$): initial error indicators.

boundary conditions are adiabatic walls on the airfoil and freestream conditions in the far field. The output of interest is the drag.

Similar to the inviscid Navier–Stokes problem, we consider two different angles of attack: $\alpha = 2$ deg and $\alpha = 4$ deg. The initial guess to the optimizer is an h-adapted mesh for $\alpha = 2$ deg. Figure 15 shows the primal x-momentum solution and the x-momentum component of the drag adjoint around the leading edge computed with high order ($p = 4$). We see boundary-layer and wake structures in both the primal and adjoint. Furthermore, this figure shows that the small change in angle of attack results in relatively small change in the flow; thus, the same initial mesh can be used as a good initial guess for both cases. Table 4 shows the optimization parameters used for our analysis. Note that we only make small changes in the parameter settings compared to ones used for the inviscid problem; only the line search backtracking factor options and μ_b are changed.

Figure 16 shows a zoomed-in view of the mesh and the flow around the leading edge, trailing edge, and wake. An exact fine-space solution is used to drive the optimizer. First, let us take a look at the boundary-layer structures around the leading edge, where there is a rapid change of velocity close to the airfoil surface. We expect that the high-order nodes of the elements in the boundary-layer structure will move toward the airfoil surface, and this is shown in Fig. 16j. Second, we take a look at the x-momentum component of the drag adjoint and notice that the flow in the middle element (B) changes appreciably between $\alpha = 2$ deg and $\alpha = 4$ deg (see Figs. 16e and 16h). This difference in the drag adjoint causes different node movement in this element, as shown in Fig. 16k. Third, notice that the flow at the end of the wake in element C changes significantly due to the change in the angle of attack (see Figs. 16f and 16i). This causes different node movement in this element for $\alpha = 2$ deg and $\alpha = 4$ deg, as shown in Fig. 16l.

Unlike for Euler, in the Navier–Stokes case, the difference in the optimized mesh is more visible in certain elements of the mesh when the optimizer is driven by the exact fine-space solution or by the fine-space surrogate. Figure 17 shows the difference in node movement for an element around the trailing edge due to accuracy of the fine-space solution used in driving the optimizer. Table 5 shows the final error reduction factors for the viscous Navier–Stokes problem obtained with various p and $q = 3$. Similar to Euler, the case that starts with an optimum starting mesh ($\alpha = 2$ deg) is provided with a cheaper iterative solver. The number of iterations for element-block Jacobi smoothing is kept the same as before.

G. Multiple-Element Optimization for the Reynolds-Averaged Navier–Stokes Equations

Finally, we consider a Reynolds-averaged Navier–Stokes (RANS) problem, closed with a negative-turbulent-viscosity modification of the Spalart–Allmaras (SA) one-equation model [40]

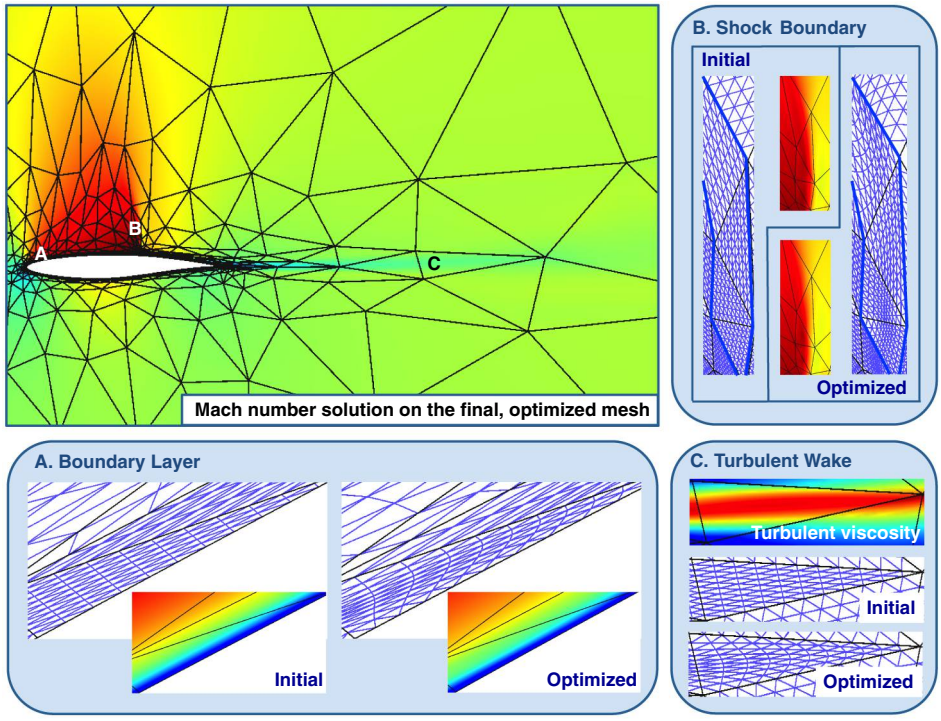
$$\nabla \cdot \boldsymbol{F}(\boldsymbol{u}) + \nabla \cdot \boldsymbol{G}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{S}(\boldsymbol{u}) \tag{14}$$

where $\boldsymbol{F}$ and $\boldsymbol{G}$ are, respectively, the inviscid and viscous fluxes; and $\boldsymbol{S}$ is the SA source. The geometry is the RAE 2822 airfoil, and the flow conditions are $M = 0.734$ and $Re = |\boldsymbol{V}|c/\nu = 6.5 \times 10^6$. The boundary conditions are adiabatic walls on the airfoil and freestream conditions in the far field. The output of interest is the drag.

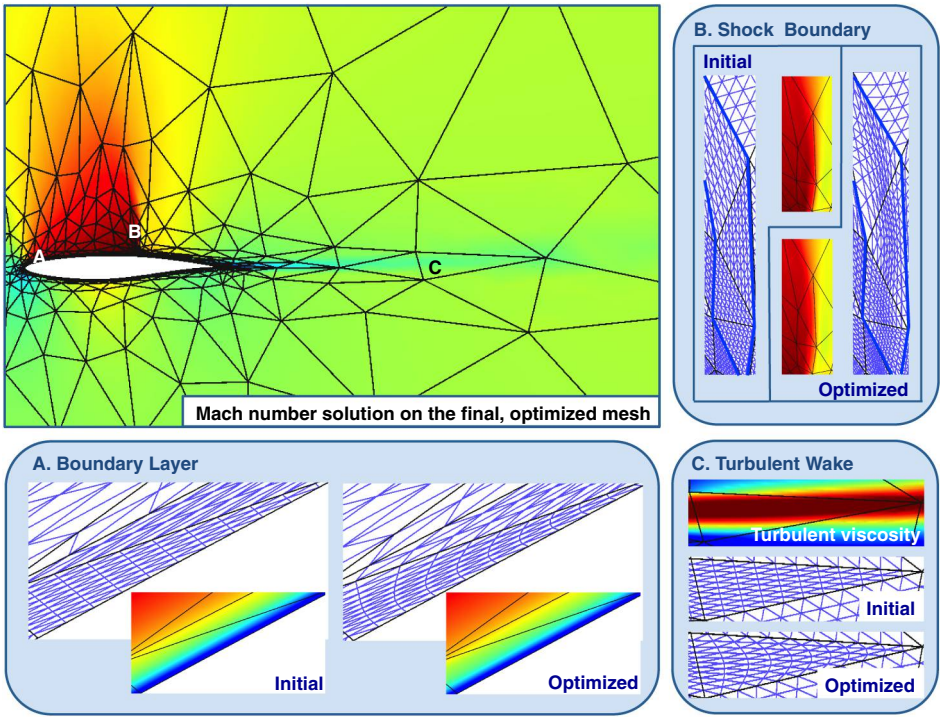
Two different angles of attack are considered here: $\alpha = 2.79$ deg and $\alpha = 3.29$ deg. The initial guess to the optimizer is an h-adapted mesh for $\alpha = 2.79$ deg. Figure 18 shows the primal Mach number, x momentum, and turbulent-viscosity solution computed with high-order approximation ($p = 4$). We see a shock structure on the upper surface of the airfoil, a boundary-layer structure around the airfoil, and a turbulent wake structure. Figure 19 shows regions of the flow where high error occurs.

Table 6 shows the optimization parameters used for our analysis. First, note that, unlike in the previous two cases, the settings for $p = 1$ are slightly different than the ones for $p = 2$ and $p = 3$. With $p = 1$, the problem is underresolved, particularly because of the thin boundary and shear layers, and it is therefore more prone to having negative Jacobian determinants and local overoptimization. This is why, in Table 5, μ_b and η_V are increased, whereas the number of BFGS iterations for SEQ 3 (SEQ denotes the optimization iteration) is decreased compared to the previous two cases. Furthermore, notice that μ_b decreases as p increases. With sharper solution features of RANS, more aggressive optimization is needed and can be used here. We have also observed that the probability of overoptimizing locally is lower compared to the previous two cases. Also, notice that, for $p = 2$ and $p = 3$, there are four options of line search backtracking factors instead of three. We find that having one additional option tends to improve the global performance of the optimizer. Moreover, f^{adapt} is now set to 0.25. In addition to reducing computational cost, lower f^{adapt} is better here because there is a larger range of errors within the mesh and only targeting some elements with high errors improves the global performance of the optimizer. Finally, notice that the number of BFGS iterations for SEQ 3 for $p = 2$ and $p = 3$ are increased. More BFGS iterations are needed here because RANS cases are generally more complex than Euler or laminar Navier–Stokes cases.

Figure 20 compares the change in flow solution and mesh due to the change in α. The Mach number solutions show that the small change in α causes changes in the shock location and the angle of the turbulent wake. In region A, we can see that boundary-layer structure formed on the upper surface of the airfoil and, similar to the previous case, this results in high-order nodes moving closer to the airfoil surface. In region B, we can see how elements around the shock are curved to improve the output calculation. The node movement is more vigorous for $\alpha = 3.29$ deg because the initial mesh is optimized for $\alpha = 2.79$ deg. The relatively small change in α causes a slight change in the shock location, and it is up to the curved elements to improve the approximation. Lastly, in region C, we see that the change in turbulent viscosity results in different node movement of the element in the wake. Table 7 shows the benefits obtained from curved elements for RANS.



a) $\alpha = 2.79°$



b) $\alpha = 3.29°$

Fig. 20 RANS ($p = 3$): Mach number solution on the final, optimized mesh along with mesh comparison on regions A–C.

Table 7 RANS: final error reduction factor based on $(p + \Delta p)$ exact fine-space solution with $\Delta p = 1$ and $q = 3$

α, deg	$p = 1$	$p = 2$	$p = 3$
2.79	1.08	1.80	2.56
3.29	1.08	1.07	7.97

VI. Conclusions

In this paper, a method is presented for tailoring basis functions in a finite element discretization to better approximate a solution. This tailoring requires virtually no additional infrastructure beyond that already available to support curved elements in a high-order discretization: in this case, discontinuous Galerkin. Instead, the locations of high-order geometry nodes become tunable parameters that warp the reference-to-global coordinate mapping and allow for accurate approximation of high-order solution features using low-

order polynomials in reference space. An element-local optimization algorithm is introduced for determining the ideal positions of these nodes, driven by both least-squares and output-based error metrics. For scalar problems, at least a 10-fold reduction in the error is observed, both in least-squares and in output measures. For the Euler, Navier–Stokes, and Reynolds-averaged Navier–Stokes equations, the benefits varied more with physical model and approximation order: though, in general, at least a factor of two error reduction for most runs. Some tuning is performed of the optimization parameters, specific to the approximation order and modeling physics, and the elimination of this tuning in the interest of full automation is the subject of ongoing work. It is noted that the proposed element shape optimization does not add any degrees of freedom to the system of equations. It does require fine-space information, though this could be reused from output-based $h/p/hp$ adaptation. Future work includes implementation of metric-based node placement and combination with hp adaptation.

Acknowledgments

The authors acknowledge the financial support of the University of Michigan, the Department of Energy (grant DE-FG02-13ER26146/DE-SC0010341), the U.S. Air Force Office of Scientific Research (grant: HPC2UM-AFOSR-2015-01), and the Michigan Institute for Computational Discovery and Engineering Fellowship.

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, No. 8, 2013, pp. 811–845. doi:10.1002/fld.v72.8

[2] Houston, P., Senior, B., and Süli, E., "hp-Discontinuous Galerkin Finite Element Methods for Hyperbolic Problems: Error Analysis and Adaptivity," *International Journal for Numerical Methods in Fluids*, Vol. 40, Nos. 1–2, 2002, pp. 153–169. doi:10.1002/fld.271

[3] 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. doi:10.2514/1.J050073

[4] Wang, L., and Mavriplis, D., "Adjoint-Based h-p Adaptive Discontinuous Galerkin Methods for the 2D Compressible Euler Equations," *Journal of Computational Physics*, Vol. 228, No. 20, 2009, pp. 7643–7661. doi:10.1016/j.jcp.2009.07.012

[5] Fidkowski, K., "High-Order Output Based Adaptive Methods for Steady and Unsteady Aerodynamics," *37th Advanced CFD Lectures Series*, edited by Deconinck, H., and Abgrall, R., von Kármán Inst. for Fluid Dynamics 2014-03, 2013.

[6] Demkowicz, L., and Gopalakrishnan, J., "A Class of Discontinuous Petrov-Galerkin Methods. Part I: The Transport Equation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 199, Nos. 23–24, 2010, pp. 1558–1572. doi:10.1016/j.cma.2010.01.003

[7] Demkowicz, L., and Gopalakrishnan, J., "A Class of Discontinuous Petrov-Galerkin Methods. Part II: Optimal Test Functions," *Numerical Methods for Partial Differential Equations*, Vol. 27, No. 1, 2011, pp. 70–105. doi:10.1002/num.v27.1

[8] Demkowicz, L., Gopalakrishnan, J., and Niemi, A., "A Class of Discontinuous Petrov-Galerkin Methods. Part III: Adaptivity," *Applied Numerical Mathematics*, Vol. 62, No. 4, 2012, pp. 396–427. doi:10.1016/j.apnum.2011.09.002

[9] Kast, S. M., Dahm, J. P., and Fidkowski, K. J., "Optimal Test Functions for Boundary Accuracy in Discontinuous Finite Element Methods," *Journal of Computational Physics*, Vol. 298, No. 1, 2015, pp. 360–386. doi:10.1016/j.jcp.2015.05.048

[10] Melenk, J., and Babuška, I., "The Partition of Unity Finite Element Method: Basic Theory and Applications," *Computer Methods in Applied Mechanics and Engineering*, Vol. 139, Nos. 1–4, 1996, pp. 289–314. doi:10.1016/S0045-7825(96)01087-0

[11] Dolbow, J., and Belytschko, T., "A Finite Element Method for Crack Growth Without Remeshing," *International Journal for Numerical Methods in Engineering*, Vol. 46, No. 1, 1999, pp. 131–150. doi:10.1002/(ISSN)1097-0207

[12] Benson, D., Bazilevs, Y., De Luycker, E., Hsu, M.-C., Scott, M., Hughes, T., and Belytschko, T., "A Generalized Finite Element Formulation for Arbitrary Basis Functions: From Isogeometric Analysis to XFEM," *International Journal for Numerical Methods in Engineering*, Vol. 83, No. 6, 2010, pp. 765–785.

[13] Farhat, C., Harari, I., and Franca, L. P., "The Discontinuous Enrichment Method," *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, No. 48, 2001, pp. 6455–6479. doi:10.1016/S0045-7825(01)00232-8

[14] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid Solution of High–Order Discontinuous Galerkin Discretizations of the Compressible Navier–Stokes Equations," *Journal of Computational Physics*, Vol. 207, No. 1, 2005, pp. 92–113. doi:10.1016/j.jcp.2005.01.005

[15] Capon, P. J., and Jimack, P. K., "On the Adaptive Finite Element Solution of Partial Differential Equations Using h-r Refinement," Univ. of Leeds, School of Computing TR-96.03, Leeds, England, U.K., 1996.

[16] Bank, R. E., and Smith, R. K., "Mesh Smoothing Using a Posteriori Error Estimates," *SIAM Journal on Numerical Analysis*, Vol. 34, No. 3, 1997, pp. 979–997. doi:10.1137/S0036142994265292

[17] Schneider, R., and Jimack, P. K., "Toward Anisotropic Mesh Adaptation Based Upon Sensitivity of a Posteriori Estimates," Univ. of Leeds, School of Computing TR 2005.03, Leeds, England, U.K., 2005.

[18] Thompson, J. F., "General Curvilinear Coordinate Systems," *Applied Mathematics and Computation*, Vol. 10, 1982, pp. 1–30. doi:10.1016/0096-3003(82)90185-0

[19] Hawken, D., Gottlier, J. J., and Hansen, J., "Review of Some Adaptive Node-Movement Techniques in Finite-Element and Finite-Difference Solutions of Partial Differential Equations," *Journal of Computational Physics*, Vol. 95, No. 2, 1991, pp. 254–302. doi:10.1016/0021-9991(91)90277-R

[20] Lang, J., Cao, W., Huang, W., and Russell, R. D., "A Two-Dimensional Moving Finite Element Method with Local Refinement Based on a Posteriori Error Estimates," *Applied Numerical Mathematics*, Vol. 46, No. 1, 2003, pp. 75–94. doi:10.1016/S0168-9274(03)00013-8

[21] Sastry, S. P., and Kirby, R. M., "On Interpolation Errors Over Quadratic Nodal Triangular Finite Elements," *Proceedings of the 22nd International Meshing Roundtable*, Springer, New York, 2014, pp. 349–366.

[22] Reed, W., and Hill, T., "Triangular Mesh Methods for the Neutron Transport Equation," Los Alamos Scientific Lab. TR LA-UR-73-479, Los Alamos, NM, 1973.

[23] Cockburn, B., and Shu, C.-W., "Runge-Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems," *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 173–261. doi:10.1023/A:1012873910884

[24] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372. doi:10.1016/0021-9991(81)90128-5

[25] Bassi, F., and Rebay, S., "GMRES Discontinuous Galerkin Solution of the Compressible Navier–Stokes Equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by Cockburn, K., and Shu, C.-W., Springer, Berlin, 2000, pp. 197–208.

[26] Bassi, F., and Rebay, S., "High-Order Accurate Discontinuous Finite Element Solution of the 2-D Euler Equations," *Journal of Computational Physics*, Vol. 138, No. 2, 1997, pp. 251–285. doi:10.1006/jcph.1997.5454

[27] Collins, E. M., and Luke, E. A., "Evaluation of Curved Element Discontinuous Galerkin Meshes," AIAA Paper 2008-5254, 2008.

[28] Toulorge, T., and Desmet, W., "Curved Boundary Treatments for the Discontinuous Galerkin Method Applied to Aeroacoustic Propagation," AIAA Paper 2009-3176, 2009.

[29] Hesthaven, J. S., and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Springer Science and Business Media, New York, 2007, pp. 479–489.

[30] Persson, P.-O., and Peraire, J., "Curved Mesh Generation and Mesh Refinement Using Lagrangian Solid Mechanics," AIAA Paper 2009-0949, 2009.

[31] Truong, A. H., Oldfield, C. A., and Zingg, D. W., "Mesh Movement for a Discrete-Adjoint Newton-Krylov Algorithm for Aerodynamic Optimization," *AIAA Journal*, Vol. 46, No. 7, 2008, pp. 1695–1704. doi:10.2514/1.33836

[32] Warburton, T., "A Low-Storage Curvilinear Discontinuous Galerkin Method for Wave Problems," *SIAM Journal on Scientific Computing*, Vol. 35, No. 4, 2013, pp. A1987–A2012. doi:10.1137/120899662

[33] Botti, L., "Influence of Reference-to-Physical Frame Mappings on Approximation Properties of Discontinuous Piecewise Polynomial Spaces," *Journal of Scientific Computing*, Vol. 52, No. 3, 2012, pp. 675–703. doi:10.1007/s10915-011-9566-3

[34] Rannacher, R., "Adaptive Galerkin Finite Element Methods for Partial Differential Equations," *Journal of Computational and Applied Mathematics*, Vol. 128, Nos. 1–2, 2001, pp. 205–233. doi:10.1016/S0377-0427(00)00513-6

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

[36] Lu, J., "An a Posteriori Error Control Framework for Adaptive Precision Optimization Using Discontinuous Galerkin Finite Element Method," Ph.D. Dissertation, Massachusetts Inst. of Technology, Cambridge, MA, 2005.

[37] Oliver, T. A., "A High-Order, Adaptive, Discontinuous Galerkin Finite Elemenet Method for the Reynolds-Averaged Navier–Stokes Equations," Ph.D. Dissertation, Massachusetts Inst. of Technology, Cambridge, MA, 2008.

[38] Johnen, A., Remacle, J.-F., and Geuzaine, C., "Geometrical Validity of Curvilinear Finite Elements," *Journal of Computational Physics*, Vol. 233, Jan. 2013, pp. 359–372. doi:10.1016/j.jcp.2012.08.051

[39] Dennis, J. E., Jr., and More, J. J., "Quasi-Newton Methods, Motivation and Theory," *Society for Industrial and Applied Mathematics Review*, Vol. 19, No. 1, 1977, pp. 359–372.

[40] Allmaras, S., Johnson, F., and Spalart, P., "Modifications and Clarifications for the Implementation of the Spalart–Allmaras Turbulence Model," *7th International Conference on Computational Fluid Dynamics (ICCFD7)*, 2012, Paper 1902.

H. Blackburn
Associate Editor