AIAA SciTech Forum
4-8 January 2016, San Diego, California, USA
54th AIAA Aerospace Sciences Meeting

10.2514/6.2016-0835

A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization

Krzysztof J. Fidkowski*
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

This work presents an output-based anisotropic mesh optimization algorithm that builds on previous work of Yano.[1] The distinguishing new feature is a simple error sampling approach for determining the convergence rate tensor of the error on a single element in a finite-element discretization. This approach does not require refinement of the element, nor any additional solves or residual evaluations. Instead, all calculations take place on the original mesh, using fine-space adjoint projections that reflect the extra resolution introduced by a sampled refinement option. The sampling is efficient and relatively simple to implement as it does not require mesh manipulation. Results for various cases demonstrate that the mesh optimization adds resolution and creates anisotropy as expected, and that it performs well compared to uniform refinement strategies and heuristic anisotropy detection methods.

I. Introduction

Unstructured meshes offer flexibility in mesh generation and in adaptation, since resolution can be placed only where necessary. *Resolution* refers to the size and shape of an element, as both of these affect the approximation power of the mesh. This information can be encoded in a metric field[2,3] over the computational domain. Our goal is to relate this metric field to a practical objective, such as minimizing error or computational cost.

Previous works have derived metric fields from consideration of such objectives through heuristic,[4–7] semi-heuristic,[8–11] and more recently, rigorous[1] ways. In this paper we consider a modified version of the existing rigorous output-based approach of Yano[1] for determining the metric that gives the best possible mesh. Our modification is in the form of an element-local error sampling technique that simplifies the implementation by not requiring mesh refinement, nor solves on refined subsets of the original mesh, nor information from neighbors.

In a solution-adaptive setting, we compare the performance of the proposed mesh optimization approach to uniform refinement and other adaptive approaches, including output-based methods with heuristic anisotropy detection. We show that the present method not only improves accuracy at fixed costs but also improves solver robustness.

The outline for the remainder of this paper is as follows. We briefly review our chosen discretization, a high-order discontinuous Galerkin finite-element method, in Section II. We then present output-error estimates in a variational setting in Section III. Section IV describes the iterative mesh optimization approach, which relies on error and cost models. Section V presents the new error sampling approach that is used in determining the error convergence rate tensor. We finish with results in Section VI and conclusions in Section VII.

*Associate Professor, AIAA Senior Member

II. Discretization

We present the mesh optimization method in the context of a discontinuous Galerkin (DG) finite element discretization.[12–14] We consider a system of partial differential equations in conservative form,

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

where $\mathbf{u} \in \mathbb{R}^s$ is the s-component state vector, $\vec{\mathbf{F}} \in \mathbb{R}^{d \times s}$ is the flux vector, d is the spatial dimension, and $\mathbf{S}$ is a source term, e.g. associated with turbulence modeling closure equations.

We assume a subdivision, T_h, of the computational domain, Ω, into N_e non-overlapping elements, Ω_e, and on each element we approximate the state by an order p polynomial. Specifically, the state approximation is $\mathbf{u}_h^p \in \boldsymbol{\mathcal{V}}_h^p = [\mathcal{V}_h^p]^s$, where

$$\mathcal{V}_h^p \equiv \left\{ u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \ \forall \Omega_e \in T_h \right\}. \tag{2}$$

The subscript h indicates a specific domain subdivision, i.e. the collective size/shape distribution of all of the elements, and $\mathcal{P}^p$ is the set of order-p polynomials [a].

Multiplying Eqn. 1 by test functions $\mathbf{v}_h^p \in \boldsymbol{\mathcal{V}}_h^p$, integrating by parts on each element, and using the Roe[15] convective flux and the second form of Bassi and Rebay (BR2)[16] for the viscous treatment, we obtain the following semilinear weak form:

$$\mathcal{R}_h^p(\mathbf{u}_h^p, \mathbf{v}_h^p) \quad = \quad 0. \tag{3}$$

Much of the DG discretization is standard; the sections below outline a few practical details.

III. Output Error Estimation

We use an adjoint-based output error estimate to drive the mesh optimization. Theoretical details can be found in previous works.[17–19] For a scalar output, $\mathcal{J}(\mathbf{u}_h^p)$, the discrete adjoint field $\boldsymbol{\psi}_h^p \in \boldsymbol{\mathcal{V}}_h^p$ is the linearized sensitivity of $\mathcal{J}$ to residual source perturbations, $\delta\mathcal{R}(\cdot) : \mathcal{V}_h^p \to \mathbb{R}$ added to the left-hand side of Eqn. 3,

$$\delta\mathcal{J} = \delta\mathcal{R}(\boldsymbol{\psi}_h^p), \tag{4}$$

where the equality assumes infinitesimal perturbations. We obtain the adjoint field by solving the following linear equation,

$$\mathcal{R}_h'^p[\mathbf{u}_h^p](\mathbf{v}_h^p, \boldsymbol{\psi}_h^p) + \mathcal{J}'[\mathbf{u}_h^p](\mathbf{v}_h^p) \quad = \quad 0, \quad \forall \mathbf{v}_h^p \in \boldsymbol{\mathcal{V}}_h^p, \tag{5}$$

where the prime denotes Fréchét linearization with respect to the argument in square brackets.

Output error estimation relies on a finer discretization space, which in this work is $\boldsymbol{\mathcal{V}}_h^{p+1}$: order $p+1$ approximation on elements of the same domain subdivision, T_h. The original (coarse) primal state $\mathbf{u}_h^p$ does not generally satisfy the fine-space weak form; instead it satisfies a perturbed weak form, with a residual source of $\delta\mathcal{R}(\cdot) = -\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \cdot)$. If we have a fine-space adjoint, $\boldsymbol{\psi}_h^{p+1}$, we can then estimate the error in $\mathcal{J}$ due to using the coarse primal instead of the (never calculated) fine-space primal,

$$\text{output error } = \delta\mathcal{J} \equiv \mathcal{J}(\mathbf{u}_h^p) - \mathcal{J}(\mathbf{u}_h^{p+1}) \approx -\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \boldsymbol{\psi}_h^{p+1}). \tag{6}$$

[a] The polynomials are defined in reference space and for curved elements they may not remain order p polynomials after the mapping to physical space.

The approximation sign indicates that the adjoint-weighted residual (last term) is an *estimate* of the error between the spaces when the equations or output are not linear, since we are using linear sensitivities with non-infinitesimal perturbations. The calculation in Eqn. 6 is also only an estimate of the true numerical error, i.e. compared to the exact solution, since it uses an approximate, finite-dimensional, fine space. The richer the fine space, the better the error estimate. We rely on this observation when formulating the element-local error sampling procedure in Section V. Finally, we note that Eqn. 6 can be localized to elemental contributions,

$$\delta\mathcal{J} \approx \sum_{e=1}^{N_e} -\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \boldsymbol{\psi}_h^{p+1}\big|_{\Omega_e}), \qquad \mathcal{E}_e \equiv |\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \boldsymbol{\psi}_h^{p+1}\big|_{\Omega_e})|. \tag{7}$$

$\mathcal{E}_e \geq 0$ in the above equation is the error indicator for element e.

IV. Mesh Optimization

Our goal is to optimize the computational mesh, T_h, in order to minimize the output error at a prescribed computational cost. We follow the approach introduced by Yano,[1] which iteratively determines the optimal change in the mesh metric field given a prescribed metric-cost relationship and a sampling-inferred metric-error relationship. In this section we briefly review the key elements of this method, and in Section V we present a modification to the error sampling procedure.

IV.A. Metric-Based Meshing

A Riemannian metric field, $\mathcal{M}(\vec{x})$, is a field of symmetric positive definite (SPD) tensors that can be used to encode information about the desired size and stretching of a computational mesh. At each point in physical space, $\vec{x}$, the metric tensor $\mathcal{M}(\vec{x})$ provides a "yardstick" for measuring the distance from $\vec{x}$ to another point infinitesimally far away, $\vec{x} + \delta\vec{x}$. This distance is

$$\delta\ell = \sqrt{\delta\vec{x}^T \mathcal{M} \delta\vec{x}}. \tag{8}$$

After choosing a Cartesian coordinate system and basis for physical space, $\mathcal{M}$ can be represented as a $d \times d$ SPD matrix. The set of points at unit metric distance from $\vec{x}$ is an ellipse: eigenvectors of $\mathcal{M}$ give directions along the principal axes, while the length of each axis (stretching) is the inverse square root of the corresponding eigenvalue.

A mesh that *conforms* to a metric field is one in which each edge has the same length, to some tolerance, when measured with the metric according to Eqn. 8, which can be integrated to obtain the distance between points that are not infinitesimally close together. An example of a two-dimensional metric-conforming mesher is the Bi-dimensional Anisotropic Mesh Generator (BAMG),[2] and this is used to obtain the results in the present work.

BAMG generates a mesh given a metric field, which is specified at nodes of a background mesh – the current mesh in an adaptive setting. The optimization will determine *changes* to the current, mesh-implied, metric, $\mathcal{M}_0(\vec{x})$, which is obtained on each simplex element of the background mesh by solving a linear system for the $d(d+1)/2$ independent entries of $\mathcal{M}_0(\vec{x})$; the equations in this system enforce that each of the $d(d+1)/2$ edges has unit metric measure. The element-based mesh-implied metrics are then averaged to the nodes using an affine-invariant[b] algorithm.[3]

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

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S}) \mathcal{M}_0^{\frac{1}{2}}. \tag{9}$$

[b]For example, just averaging matrix entries would be coordinate-system dependent and hence not affine invariant.

Note that $\mathcal{S} = 0$ leaves the metric unchanged, while diagonal values in $\mathcal{S}$ of $\pm 2 \log 2$ halve/double the metric stretching sizes.

IV.B. Error Convergence Model

The mesh optimization algorithm requires a model for how the error changes as the metric changes. We consider one element, Ω_e, with a current error $\mathcal{E}_{e0}$, the absolute value of the element's contribution to Eqn. 7, and a proposed metric step matrix of $\mathcal{S}_e$. What will the new error be over Ω_e following refinement with this step matrix? A typical a priori model may predict $\mathcal{E}_e/\mathcal{E}_{e0} = (h/h_0)^r = \exp[r \log(h/h_0)]$, where h/h_0 is some measure of element size change and r is the error convergence rate. A generalization to anisotropic metric tensors, for which the error may converge differently depending on the direction of change, is that S_e plays the role of $\log(h/h_0)$ and a symmetric *rate tensor* $\mathcal{R}_e$ replaces the scalar r:

$$\mathcal{E}_e = \mathcal{E}_{e0} \exp\left[\mathrm{tr}(\mathcal{R}_e \mathcal{S}_e)\right] \quad \Rightarrow \quad \frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e} = \mathcal{E}_e \mathcal{R}_e. \tag{10}$$

Note, we include a linearization with respect to $\mathcal{S}_e$ for use in Section IV.D. The total error over the mesh is the sum of the elemental errors, $\mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e$. During optimization we will want to keep $\mathcal{E}$ small, and we will be able to change the step matrices at the mesh vertices, $\mathcal{S}_v$. The rate tensor, $\mathcal{R}_e$, will be determined separately for each element through the sampling procedure described in Section V.

IV.C. Cost Model

To measure the cost of refinement, we use degrees of freedom, `dof`, which on each element just depends on the approximation order p, assumed constant over the elements. By Eqn. 9 and properties of the metric tensor, when the step matrix S_e is applied to the metric of element e, the area of the element decreases by $\exp\left[\frac{1}{2}\mathrm{tr}(S_e)\right]$. Equivalently, the number of new elements, and hence degrees of freedom, occupying the original area Ω_e *increases* by this factor. So the elemental cost model is

$$C_e = C_{e0} \exp\left[\frac{1}{2}\mathrm{tr}(\mathcal{S}_e)\right] \quad \Rightarrow \quad \frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e} = \mathcal{C}_e \frac{1}{2}\mathcal{I}, \tag{11}$$

where $C_{e0} = \mathtt{dof}_{e0}$ is the current number of degrees of freedom on element e, and $\mathcal{I}$ is the identity tensor. Again we include a linearization with respect to $\mathcal{S}_e$ for use in Section IV.D. The total cost over the mesh is the sum of the elemental costs, $\mathcal{C} = \sum_{e=1}^{N_e} C_e$.

IV.D. Metric Optimization Algorithm

Given a current mesh with its mesh-implied metric ($\mathcal{M}_0(\vec{x})$), elemental error indicators $\mathcal{E}_{e0}$, 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.

The step matrix field is approximated by values at the mesh vertices, $\mathcal{S}_v$, which are arithmetically-averaged to adjacent elements[c]:

$$\mathcal{S}_e = \frac{1}{|V_e|} \sum_{v \in V_e} \mathcal{S}_v, \tag{12}$$

where V_e is the set of vertices ($|V_e|$ is the number of them) adjacent to element e. The optimization problem is to determine $\mathcal{S}_v$ such that the total error $\mathcal{E}$ is minimized at a prescribed total cost $\mathcal{C}$.

[c]There is no need for an affine-invariant average because entries of $\mathcal{S}$ are coordinate system independent.

First-order optimality conditions require derivatives of the error and cost with respect to $\mathcal{S}_v$. From the equations in Sections IV.B and IV.C, and using Eqn. 12, these are

$$\frac{\partial \mathcal{E}}{\partial \mathcal{S}_v} = \sum_{e \in E_v} \underbrace{\frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e}}_{\text{Eqn. 10}} \underbrace{\frac{\partial \mathcal{S}_e}{\partial \mathcal{S}_v}}_{1/|V_e|}, \qquad \frac{\partial \mathcal{C}}{\partial \mathcal{S}_v} = \sum_{e \in E_v} \underbrace{\frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e}}_{\text{Eqn. 11}} \underbrace{\frac{\partial \mathcal{S}_e}{\partial \mathcal{S}_v}}_{1/|V_e|}, \tag{13}$$

where E_v is the set of elements adjacent to vertex v. We note that the cost only depends on the *trace* of the step matrix; i.e. the trace-free part of $\mathcal{S}_e$ stretches an element but does not alter its area. We therefore separate the vertex step matrices into trace ($s_v\mathcal{I}$) and trace-free ($\widetilde{\mathcal{S}}_v$) parts,

$$\mathcal{S}_v = s_v\mathcal{I} + \widetilde{\mathcal{S}}_v. \tag{14}$$

Derivatives of the error with respect to s_v and $\widetilde{\mathcal{S}}_v$ are

$$\frac{\partial \mathcal{E}}{\partial s_v} = \operatorname{tr}\left(\frac{\partial \mathcal{E}}{\partial \mathcal{S}_v}\right), \qquad \frac{\partial \mathcal{E}}{\partial \widetilde{\mathcal{S}}_v} = \frac{\partial \mathcal{E}}{\partial \mathcal{S}_v} - \frac{\partial \mathcal{E}}{\partial s_v}\frac{\mathcal{I}}{d} \tag{15}$$

The optimization algorithm is then the same as presented by Yano:[1]

1. Given a mesh, solution, and adjoint, calculate $\mathcal{E}_e, \mathcal{C}_e, \mathcal{R}_e$ for each element e.
2. Set $\delta s = \delta s_{\max}/n_{\text{step}}$, $\mathcal{S}_v = 0$.
3. Begin loop: $i = 1 \dots n_{\text{step}}$
   (a) Calculate $\mathcal{S}_e$ from Eqn. 12, $\frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e}$ from Eqn. 10, and $\frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e}$ from Eqn. 11.
   (b) Calculate derivatives of $\mathcal{E}$ and $\mathcal{C}$ with respect to s_v and $\widetilde{\mathcal{S}}_v$ using Eqn. 15.
   (c) At each vertex form the ratio $\lambda_v = \frac{\partial \mathcal{E}/\partial s_v}{\partial \mathcal{C}/\partial s_v}$ and
      - Refine the metric for 30% of the vertices with the largest $|\lambda_v|$: $\mathcal{S}_v = \mathcal{S}_v + \delta s\mathcal{I}$
      - Coarsen the metric for 30% of the vertices with the smallest $|\lambda_v|$: $\mathcal{S}_v = \mathcal{S}_v - \delta s\mathcal{I}$
   (d) Update the trace-free part of S_v to enforce stationarity with respect to shape changes at fixed area: $S_v = S_v + \delta s(\partial\mathcal{E}/\partial\widetilde{\mathcal{S}}_v)/(\partial\mathcal{E}/\partial s_v)$.
   (e) Rescale $S_v \to S_v + \beta\mathcal{I}$, where β is a global constant calculated from Eqn. 11 to constrain the total cost to the desired `dof` value: $\beta = \frac{2}{d}\log\frac{\mathcal{C}_{\text{target}}}{\mathcal{C}}$, where $\mathcal{C}_{\text{target}}$ is the target cost.

Note, λ_v is a Lagrange multiplier in the optimization. It is the ratio of the marginal error to marginal cost of a step matrix trace increase (i.e. mesh refinement). The above algorithm iteratively equidistributes λ_v globally so that, at optimum, all elements have the same marginal error to cost ratio. Constant values that work generally well in the above algorithm are $n_{\text{step}} = 20$ and $\delta s_{\max} = 2\log 2$.

In practice, the mesh optimization and flow/adjoint solution are performed several times at a given target cost, $\mathcal{C}_{\text{target}}$, until the error stops changing. Then the target cost is increased to reduce the error further if desired. Figure 1 illustrates this process. For reporting the final error and cost at each adaptive iteration, the values are averaged over the last few solution iterations at each target cost.

V. Element-Local Error Sampling

The mesh optimization algorithm requires that the error convergence rate tensor, $\mathcal{R}_e$ be known for each element e. We estimate this rate tensor a posteriori by *sampling* a small number of element refinements (cuts) for each element and performing a regression. This is also the approach taken by Yano,[1] and our work differs in that we do not solve any primal or adjoint problems on the refined

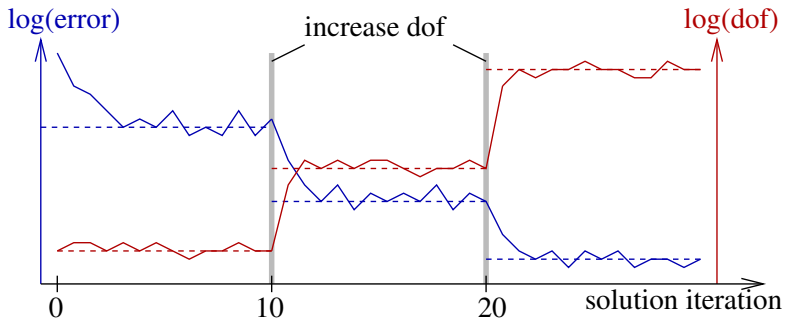


Figure 1. Illustration of the typical error (left axis) and cost/dof (right axis) history during mesh optimization. At each solution iteration, the primal and adjoint problems are solved and the mesh optimization algorithm is applied to determine the metric for the mesh for the next solution iteration. The target cost is held fixed for multiple (e.g. 10) solution iterations, during which the error eventually stabilizes as each mesh "hovers" about the optimum. Following an allowable cost increase, the error drops over a few solution/optimization iterations and settles to a lower value. Note that the primal and adjoint solutions can be transferred from one mesh to the next, making the solves at each iteration much cheaper compared to solving from scratch (e.g. using the freestream state).

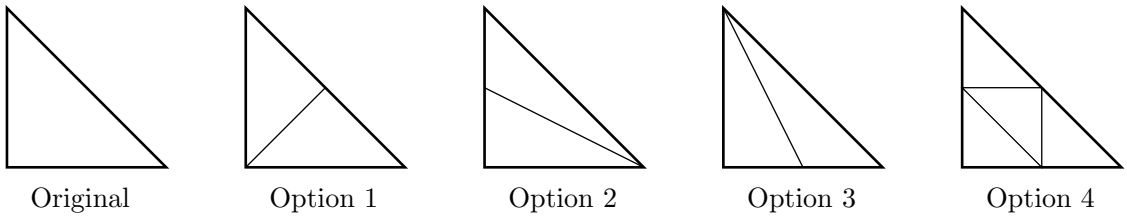


Figure 2. Four refinement options for a triangle. Each one is considered implicitly during error sampling, though the elements are never actually refined.

elements. Indeed, we never actually modify the mesh when considering the refinement samples, and this simplifies data structure handling in the implementation, especially in parallel.

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

Consider one element, Ω_e. The fine space adjoint $\boldsymbol{\psi}_h^{p+1}\big|_{\Omega_e}$ gives, via Eqn. 7, an estimate of the output error in the current order p solution, as measured relative to a "truth" order $p+1$ solution: this is $\mathcal{E}_{e0}$. If the fine-space adjoint were of order $p+2$, we would have an estimate of the error relative to an even finer space. Now, suppose that we are looking at refinement option i in Figure 2. Refining an element with such an option creates a solution space that is finer than the original, though (we assume) not as fine as increasing the order to $p+1$ – we still use $p+1$ as the "truth" space. Suppose we have an order p adjoint on this refined space; call it $\boldsymbol{\psi}_{hi}^{p}\big|_{\Omega_e}$, where the i indicates that we are considering refinement option i. With this adjoint we can compute an error indicator $\Delta\mathcal{E}_{ei}$: this estimates the error between the coarse solution and that on refinement option i. The

remaining error associated with refinement option i is then given by the difference,

$$\mathcal{E}_{ei} \equiv \mathcal{E}_{e0} - \Delta\mathcal{E}_{ei}. \tag{16}$$

Calculating $\Delta\mathcal{E}_{ei}$ requires an adjoint-weighted residual evaluation on the element refined under option i. We bypass this complication with an additional simplification: we project $\boldsymbol{\psi}^p_{hi}\big|_{\Omega_e}$ into the $p+1$ space on the original element and evaluate the adjoint weighted residual there. That is, we perform

$$\Delta\mathcal{E}_{ei} \equiv |\mathcal{R}^{p+1}_h(\mathbf{u}^p_h, \widetilde{\boldsymbol{\psi}}^p_{hi}\big|_{\Omega_e})|, \tag{17}$$

where $\widetilde{\boldsymbol{\psi}}^p_{hi}$ is $\boldsymbol{\psi}^p_{hi}$ projected from order p on refinement option i into order $p+1$ on the original element. The final simplification is to not actually calculate $\boldsymbol{\psi}^p_{hi}\big|_{\Omega_e}$ for refinement option i. Instead, we simply project the known fine-space order $p+1$ adjoint onto order p in refinement option i.

In summary, the error uncovered by refinement option i, $\Delta\mathcal{E}_{ei}$, is estimated by the adjoint-weighted residual in Eqn. 17, with all calculations occurring at order $p+1$ on the original element. Figure 3 breaks down the key computation of $\widetilde{\boldsymbol{\psi}}^p_{hi}$ into individual projections. Using least-squares

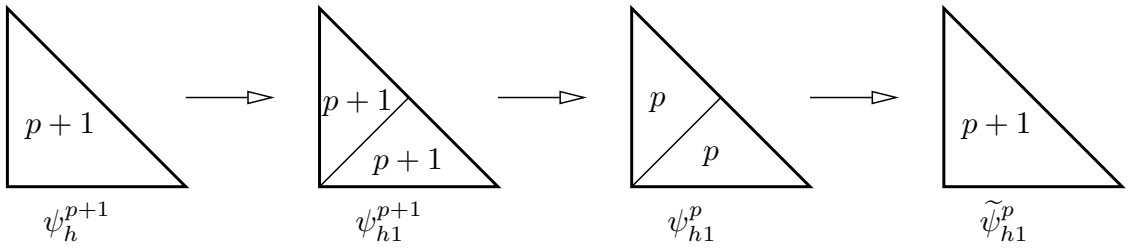


Figure 3. Projection of the fine-space adjoint, ψ_h^{p+1}, down to the space created by refinement option $i = 1$, and then back up to order $p+1$ on the original element. This sequence of projections is encapsulated in a single transfer matrix in Eqn. 18.

projections in reference space, the combination of the steps depicted in Figure 3 can be encapsulated into one transfer matrix that converts $\boldsymbol{\psi}^{p+1}_h$ into $\widetilde{\boldsymbol{\psi}}^p_{hi}$, both represented in the order $p+1$ space on the original element:

$$\widetilde{\boldsymbol{\psi}}^p_{hi} = T_i \boldsymbol{\psi}^{p+1}_h, \tag{18}$$

$$T_i = \left[M_0(\phi_0^{p+1}, \phi_0^{p+1})\right]^{-1} \sum_{k=1}^{n_i} T_{ik}, \tag{19}$$

$$T_{ik} = M_k(\phi_0^{p+1}, \phi_k^p) \left[M_k(\phi_k^p, \phi_k^p)\right]^{-1} M_k(\phi_k^p, \phi_k^{p+1}) \left[M_k(\phi_k^{p+1}, \phi_k^{p+1})\right]^{-1} M_k(\phi_k^{p+1}, \phi_0^{p+1}). \tag{20}$$

In these equations, n_i is the number of sub-elements in refinement option i, k is an index over these sub-elements, ϕ_k^p, ϕ_k^{p+1} are order p and $p+1$ basis functions on sub-element k, ϕ_0^p, ϕ_0^{p+1} are order p and $p+1$ basis functions on the original element, and components of the mass-like matrices are defined as

$$M_k(\phi_l, \phi_m) = \int_{\Omega_k} \phi_l \phi_m d\Omega, \qquad M_0(\phi_l, \phi_m) = \int_{\Omega_0} \phi_l \phi_m d\Omega, \tag{21}$$

where Ω_k is sub-element k and Ω_0 is the original element. Note that the transfer matrix T_i can be calculated for each refinement option i once in reference space and then used for all elements, so that the calculation of $\Delta\mathcal{E}_{ei}$ consumes minimal additional cost – and most importantly, no solves

or residual evaluations are needed on the refined element, as these generally require cumbersome data management and transfer.

Finally, after calculating $\mathcal{E}_{ei}$, the errors remaining after each refinement option i according to Eqn. 16, we use least-squares regression to estimate the rate tensor $\mathcal{R}_e$. Note that for triangles, we have 4 refinement options and 3 independent entries in the symmetric $\mathcal{R}_e$ tensor. Using Eqn. 10, we formulate the regression to minimize the following error, summed over refinement options,

$$\sum_i \left[\log \frac{\mathcal{E}_{ei}}{\mathcal{E}_{e0}} - \mathrm{tr}(\mathcal{R}_e \mathcal{S}_{ei}) \right]^2 . \tag{22}$$

In this equation, $\mathcal{S}_{ei}$ is the step matrix associated with refinement option i, given by (from Eqn. 9),

$$\mathcal{S}_{ei} = \log \left(\mathcal{M}_0^{-\frac{1}{2}} \mathcal{M}_i \mathcal{M}_0^{-\frac{1}{2}} \right), \tag{23}$$

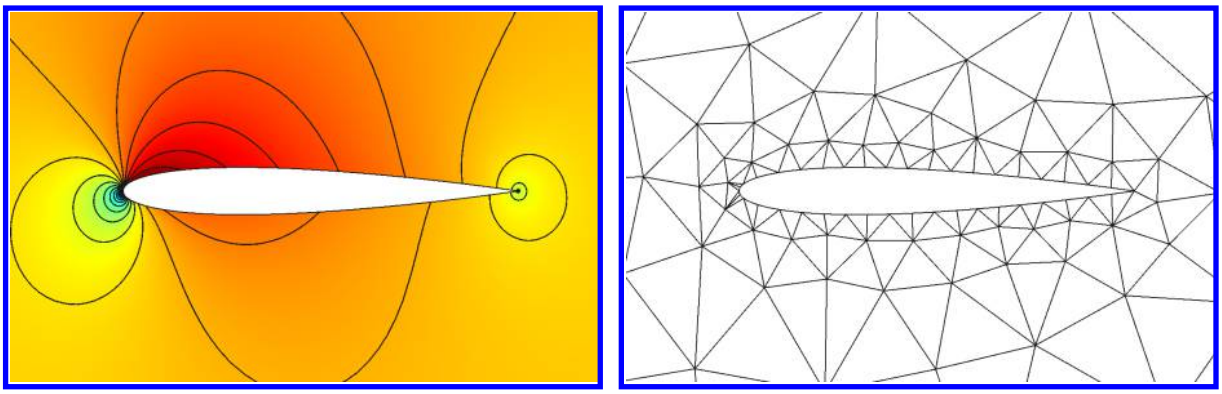
where $\mathcal{M}_i$ is the affine-invariant metric average of the mesh-implied metrics of all sub-elements in refinement option i. Differentiating Eqn. 22 with respect to the independent components of $\mathcal{R}_e$ yields a linear system for these components.

VI. Results

We demonstrate the mesh optimization algorithm with the new local error sampling technique on four test cases, using the compressible Euler, Navier-Stokes, and Reynolds-averaged Navier-Stokes equations.

VI.A. NACA 0012 Airfoil in Inviscid Flow

We first consider inviscid flow over a NACA 0012 airfoil at $M_\infty = 0.5$ and $\alpha = 2°$. The output of interest is the drag coefficient. Figure 4 shows the Mach number contours and the initial coarse mesh for this case. Curved elements of geometry order $q = 4$ are used adjacent to the airfoil to represent the geometry, and the farfield boundary is over 2000 chord-lengths away.



(a) Mach number contours (b) Initial mesh (356 triangles)

Figure 4. NACA 0012, $M_\infty = 0.5$, $\alpha = 2°$, inviscid: Mach number contours and the initial coarse mesh for metric-based mesh optimization.

Starting from the coarse mesh, adaptive runs were performed using approximation orders $p = 1, 2, 3$, and at four `dof` targets: 2000, 4000, 8000, and 16000. Note that for a given `dof` target, we expect coarser meshes (fewer elements) at higher p, when each element consumes more `dof`. At each `dof` target, ten solution iterations were performed before moving to the next `dof` target. Figure 5

shows a sample run using $p = 2$. Note how quickly the error drops in the first ~ 3 iterations after increasing the `dof` target and after starting from the coarse mesh. Since the 356-element initial mesh has 2136 `dof` at $p = 2$, the `dof` plot shows little change in the first ten iterations, when the target is 2000 `dof`, even though the error drops as the ~ 350 elements are resized and repositioned to an optimal configuration for drag prediction.

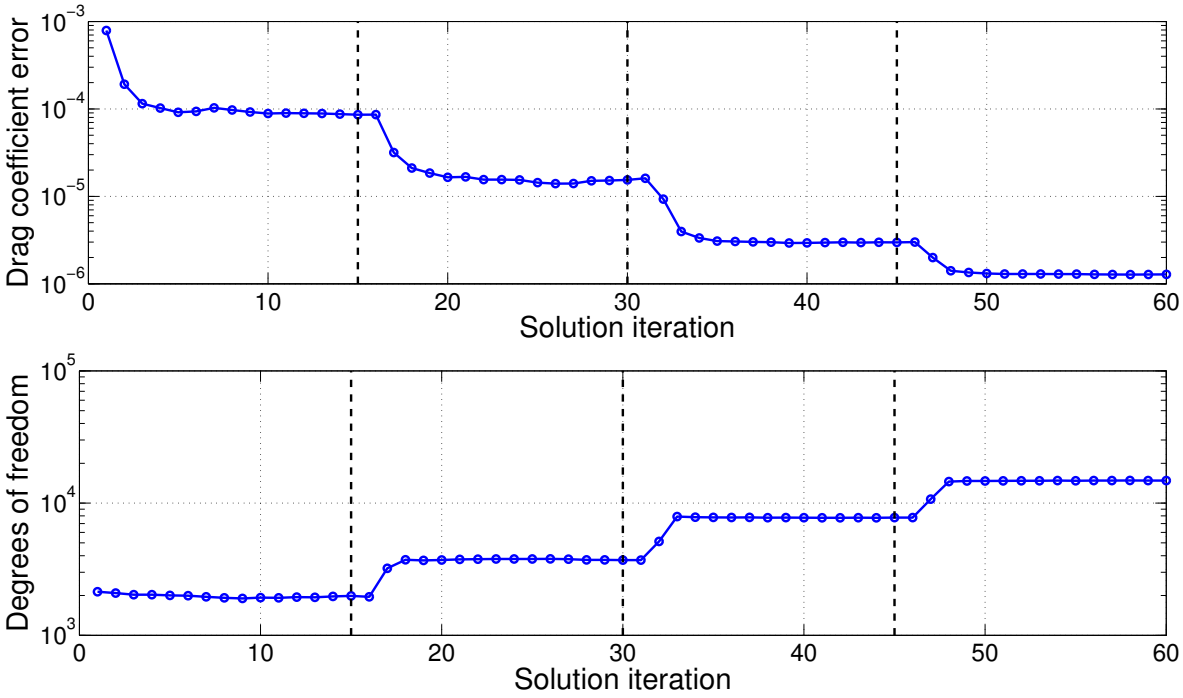


Figure 5. NACA 0012, $M_\infty = 0.5$, $\alpha = 2°$, inviscid: output error and degrees of freedom histories for mesh optimization with $p = 2$ approximation and 15 solution iterations at each target `dof` value.

Figure 6 compares the convergence of the error in the drag coefficient with a measure of the mesh size, $\texttt{dof}^{-1/2}$, when using the presented mesh optimization algorithm and when using uniform refinement of the initial mesh. The uniform refinement results exhibit stagnating convergence, even at high order, due to the presence of singular solution features (e.g. at the trailing edge) which limit the attainable rate. Indeed there is little benefit in using $p = 3$ compared to $p = 2$ on the uniform refinement sequence. On the other hand, the adaptive results with the presented optimization algorithm show greatly-improved convergence. Orders of magnitude error reductions are achieved compared to uniform refinement, and there is a clear benefit of higher-order approximation. Note that Figure 6 presents data points for every solution iteration in the mesh optimization, with larger symbols denoting the output and `dof` averaged over the five iterations at each `dof` target. As shown, the 2000 initial `dof` target required refinement of the initial mesh for $p = 1$ and coarsening of the initial mesh at $p = 3$; at $p = 2$ the total number of elements remained virtually unchanged at this target.

Figures 7, 8, 9 show the adapted meshes for $p = 1, 2, 3$ approximation, at the four target `dof` values. The differences in the number of elements among the various meshes are evident: the $p = 1$ adapted meshes are the finest, whereas the $p = 3$ ones are the coarsest. The adaptation adds mesh resolution to the most interesting areas in the flowfield: the leading and trailing edges of the airfoil. Especially at the trailing edge, small elements persist even at $p = 3$, in order to resolve the strong singularity.

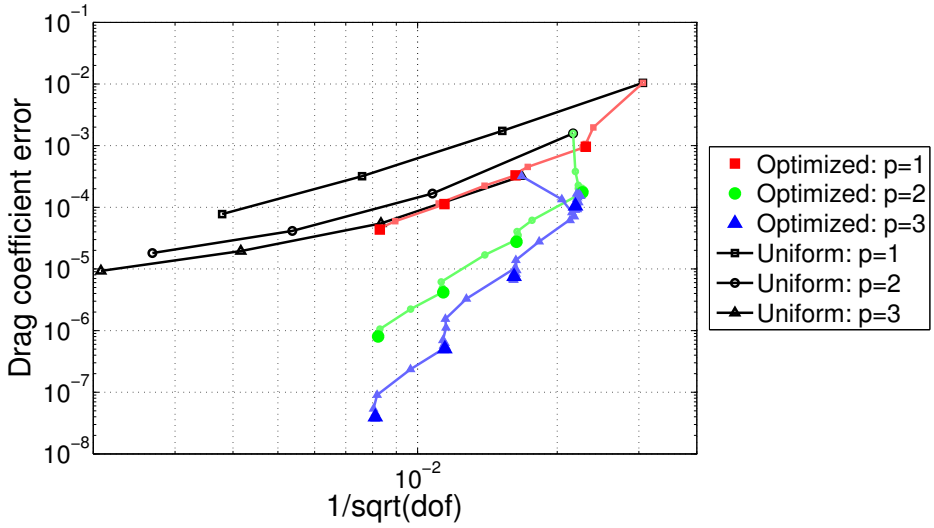
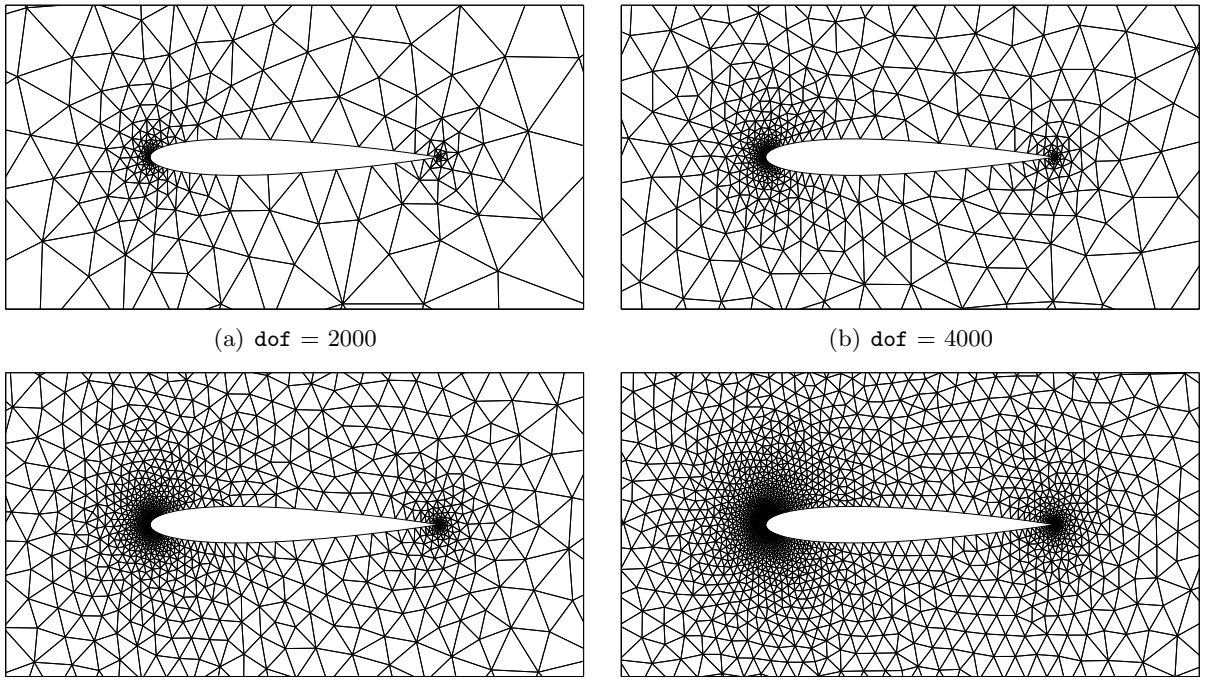


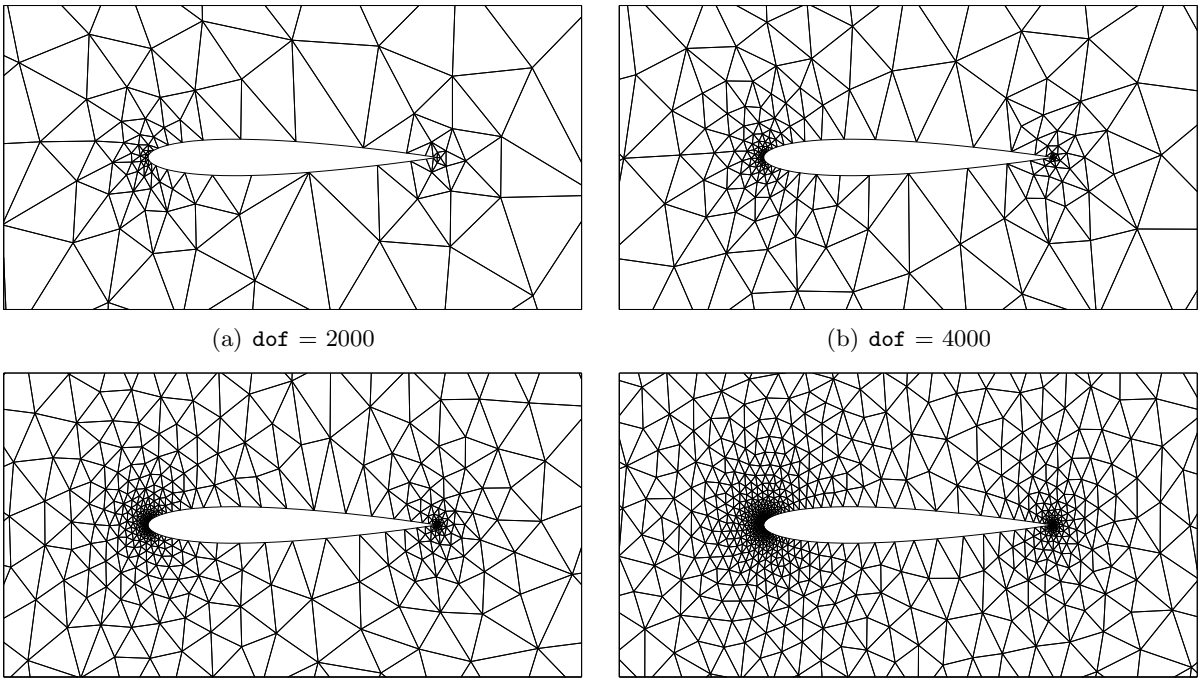
Figure 6. NACA 0012, $M_\infty = 0.5$, $\alpha = 2°$, inviscid: comparison of output error convergence with `dof` for uniform refinement and mesh optimization at orders $p = 1, 2, 3$. In the optimization results, the smaller symbols denote the output at each solution iteration, whereas the larger symbols show the average over the last five iterations at each target `dof`.



(a) `dof` = 2000 (b) `dof` = 4000

(c) `dof` = 8000 (d) `dof` = 16000

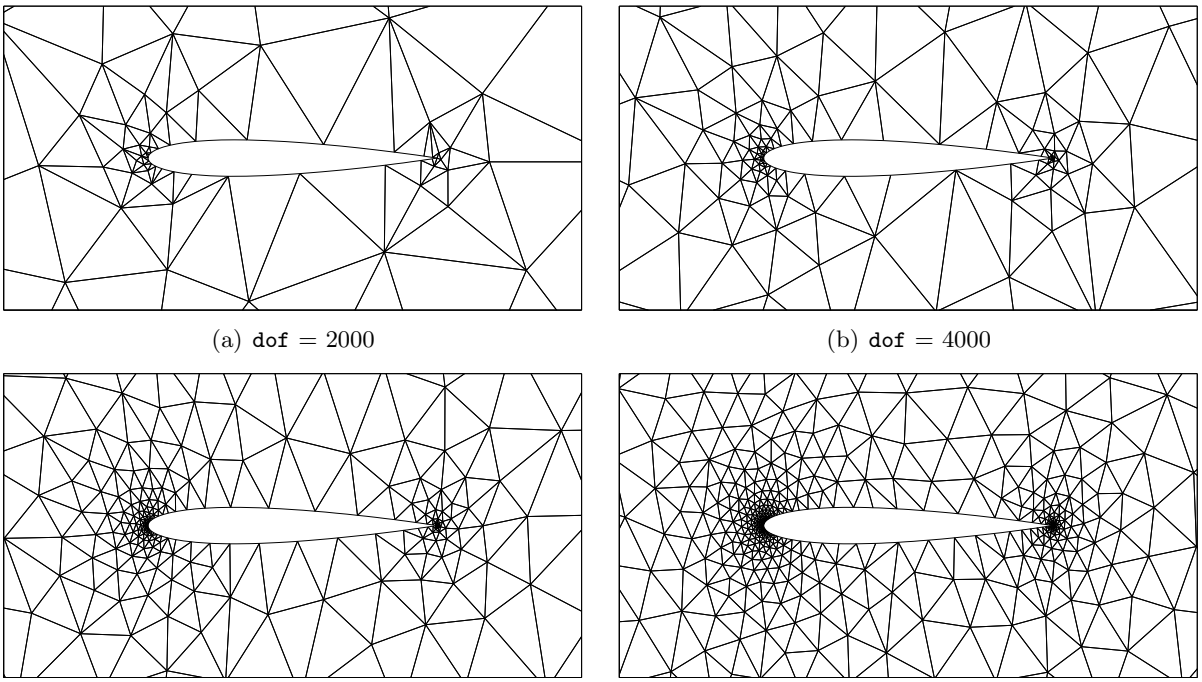
Figure 7. NACA 0012, $M_\infty = 0.5$, $\alpha = 2°$, inviscid: optimized meshes (last in sequence for each target `dof`) for $p = 1$.



(a) `dof` = 2000 (b) `dof` = 4000

(c) `dof` = 8000 (d) `dof` = 16000

Figure 8. NACA 0012, $M_\infty = 0.5$, $\alpha = 2^\circ$, inviscid: optimized meshes (last in sequence for each target `dof`) for $p = 2$.



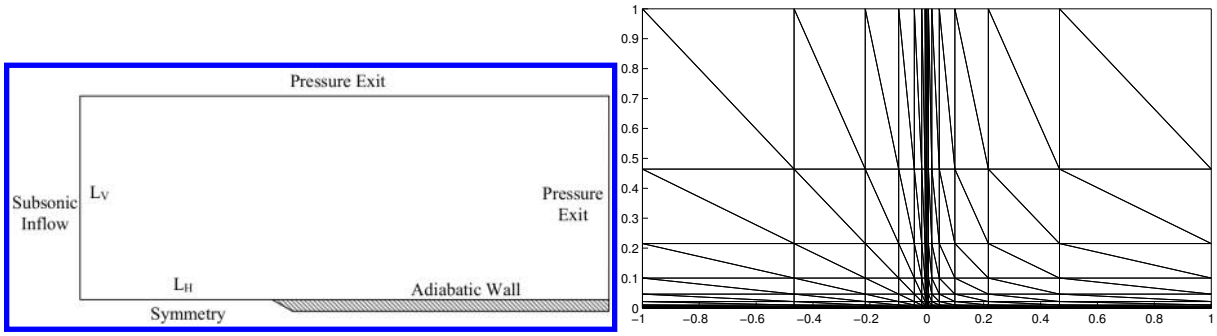
(a) `dof` = 2000 (b) `dof` = 4000

(c) `dof` = 8000 (d) `dof` = 16000

Figure 9. NACA 0012, $M_\infty = 0.5$, $\alpha = 2^\circ$, inviscid: optimized meshes (last in sequence for each target `dof`) for $p = 3$.

VI.B. Flat Plate in Laminar Flow at $Re = 10^6$

The case of interest is laminar flow over a flat plate at $M_\infty = 0.5, \alpha = 0, Re_L = 10^6$. This case was considered as part of the most recent workshop on high-order methods in CFD, held at the AIAA 2015 SciTech conference. The compressible Navier-Stokes equations are solved over the domain shown in Figure 10(a), where $L_H = L_V = L = 1$. The Prandtl number is $Pr = 0.72$ and the ratio of specific heats is $\gamma = 1.4$. The output of interest is the drag coefficient on the flat plate.



(a) Domain and boundary conditions

(b) Initial mesh: 336 triangles

Figure 10. Laminar flat-plate boundary layer: case setup and initial mesh for uniform refinement and optimization runs.

Figure 10(b) shows the initial mesh used for uniform refinement and mesh optimization runs. This mesh was obtained from subdivision of elements in an initially structured quadrilateral mesh that was generated with extra refinement in the boundary layer and leading-edge of the flat plate. In the uniform refinement runs, each triangular element was subdivided isotropically into four new elements to obtain the next mes. In addition, "quasi-uniform" refinement was considered in which a hand-generated fine mesh, also with stretched spacing, was coarsened by removal of every other grid line. The resulting mesh sequence was no longer nested but did a better job at resolving the boundary layer and leading-edge regions.

Figure 11 shows the results of the uniform refinement and adaptive runs for approximation orders $p = 1, 2, 3$. Note that among the uniform refinement runs, increasing the order does decrease the error, but that no higher-order convergence rate is observed due to insufficient resolution of the singular region of the flow (i.e. the leading edge of the flat plate). The quasi-uniform meshes show better results because they are not nested and the finer meshes intentionally have high resolution at the leading edge and in the boundary layer. The meshes optimized using the proposed method show, for a given order, the best results. Note that in this case four cost (`dof`) values were considered: $2000, 4000, 8000, 16000$. For each cost value, fifteen iterations of mesh optimization were performed, and errors from the last five were averaged to produce the large symbols shown in Figure 11. The stall in convergence around error values of 10^{-10} is likely due to the residual tolerance used (eight orders of magnitude drop).

Figures 12 and 13 show the optimized meshes for $p2$ and $p = 3$, respectively. The meshes are shown over the entire computational domain and zoomed-in with a stretched vertical axis to see refinement in the boundary layer. Comparing $p = 2$ and $p = 3$, we see coarser meshes for $p = 3$ since the number of degrees of freedom is the same for both orders, and $p = 3$ consumes, with full-order approximation, 10 degrees of freedom compared to 6 for $p = 2$.

From the un-zoomed figures, we see that the optimized meshes show no sign of the initial-mesh refinement pattern: not much resolution is required away from the boundary layer. In the zoomed-in figures we see very high resolution in the leading-edge region, and in an arc-shaped region extending from the leading edge to the trailing edge. The boundary layer of course does not

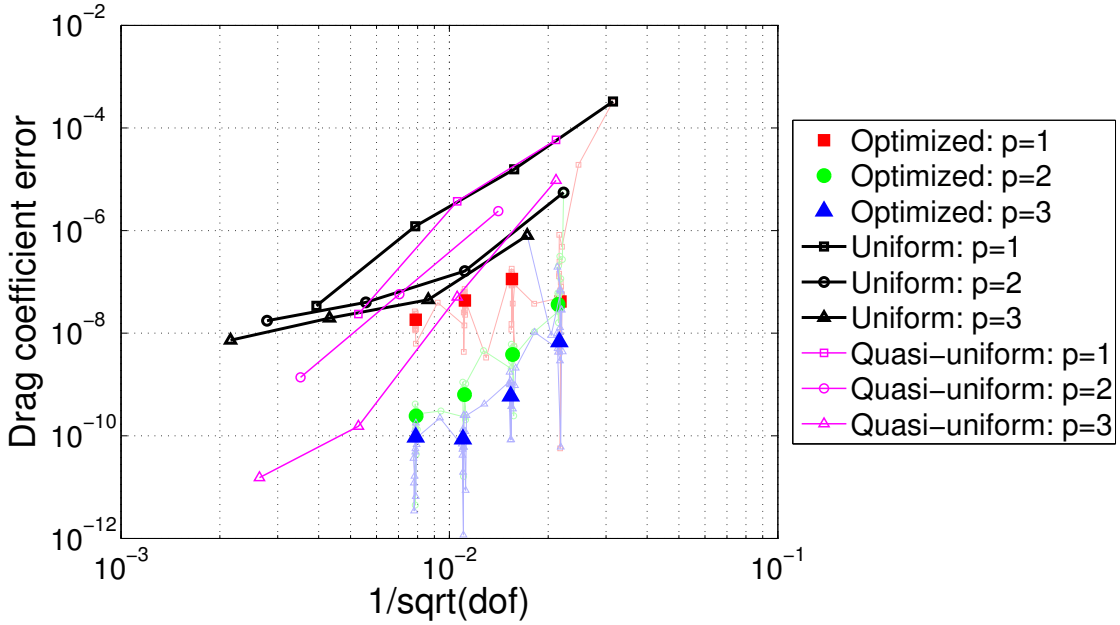


Figure 11. Laminar flat-plate boundary layer: drag error convergence histories for uniform refinement and mesh optimization. The light lines in the optimized mesh plots show the drag error for each iteration of the optimization at a fixed cost.

disappear to zero height at the trailing edge, but errors made away from the wall this far down the plate have little impact on the drag. The adaptive algorithm picks up on this. In addition, we see reasonably-high anisotropy of the elements in the boundary layer region – note that the ratio of horizontal to vertical axis scaling is 100-to-1 in the zoomed-in figures. This anisotropy is detected automatically by the error sampling procedure that yields the rate tensor for each element, as described in Section V.

VI.C. NACA 0012 Airfoil in Laminar Flow

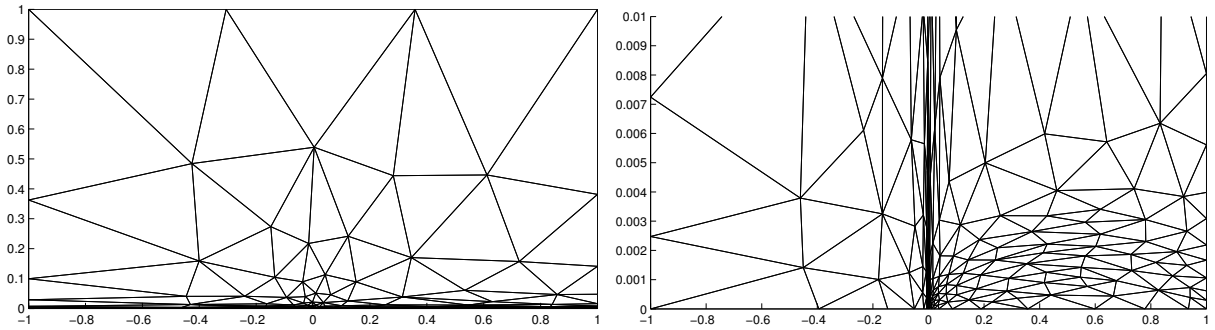
In this case we consider viscous flow over a NACA 0012 airfoil at $M_\infty = 0.5$, $\alpha = 2°$, and $Re = 5000$. The output of interest is the drag coefficient. Figure 14 shows the Mach number contours and the initial coarse mesh for this case. Curved elements of geometry order $q = 4$ are used adjacent to the airfoil to represent the geometry, and the farfield boundary is over 2000 chord-lengths away.

Starting from the coarse mesh, adaptive runs were performed using approximation orders $p = 1, 2, 3$ and four `dof` targets: 2000, 4000, 8000, and 16000. Figure 15 shows drag error and `dof` histories for a sample run using $p = 2$. Compared to the inviscid case the error does not drop as quickly after each `dof` target increase, but it does plateau after 10-15 solution iterations. The output and `dof` values reported are again averages over the last 5 solution iterations at each target `dof`.

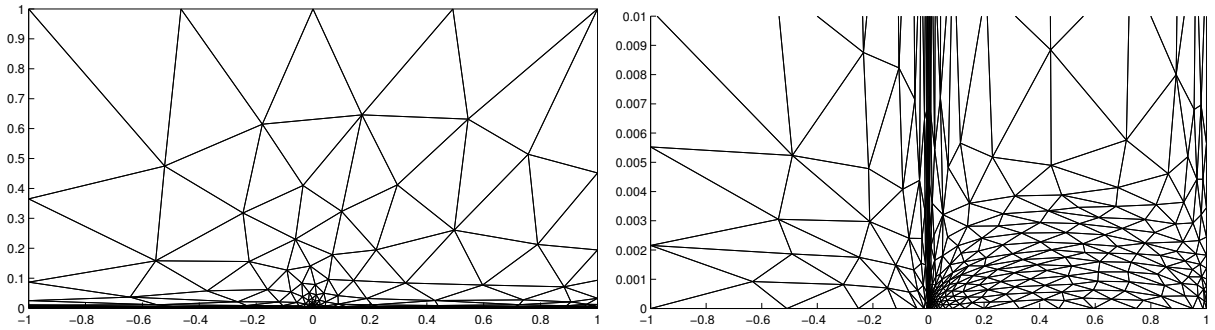
Figure 16 compares convergence of the error in the drag coefficient with a measure of the mesh size, $\texttt{dof}^{-1/2}$, when using the presented mesh optimization algorithm, when using uniform refinement of the initial mesh, and when using an alternative adaptation algorithm: one in which the output error indicator is used to set the mesh size while the anisotropy comes from the Hessian matrix associated with the Mach number.[8,10] As in the inviscid case, the uniform refinement results exhibit stagnating convergence, even at high order, due to the presence of singular solution features – there is little benefit to high order in this scenario. On the other hand, the adaptive results show greatly-improved convergence, especially with the presented optimization algorithm. Using

the Mach number Hessian in lieu of the metric optimization yields reasonable results for $p = 1$ and $p = 2$, but stalls out for $p = 3$. This should not come as a surprise, since: (1) the Hessian matrix approach was derived for linear ($p = 1$) solution approximation; and (2) the use of the Mach number Hessian is a heuristic that is not guaranteed to provide optimal meshes for the prediction of an output.

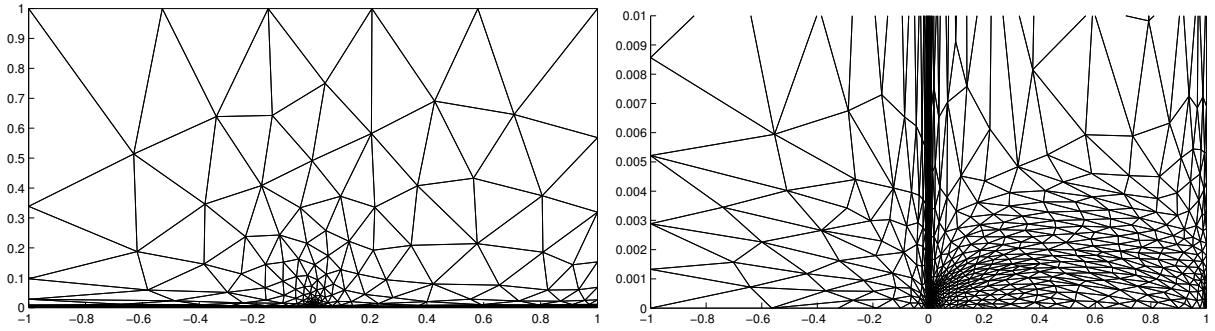
Figures 17 and 18 show the adapted meshes for $p = 2$ and $p = 3$ approximation, respectively, at the four target `dof` values. The adaptation algorithm adds resolution in the boundary layer, wake, and in the vicinity of the leading-edge stagnation streamline. The two sets of meshes, one optimized using the error sampling procedure and the other generated using the Mach number Hessian, show resolution of similar regions but with different anisotropy. In the boundary layer and in the edges of the wake, using the Mach number Hessian leads to elements with somewhat larger anisotropy. However, on the leading-edge stagnation streamline, the sampling-based meshes exhibit flow-aligned anisotropy, a result of features in the adjoint solution that the Mach Hessian cannot identify. In general, the Mach number Hessian is a reasonable heuristic for identifying anisotropy, but it is not perfect, as the output convergence results show that the sampling-based optimized meshes are more efficient.



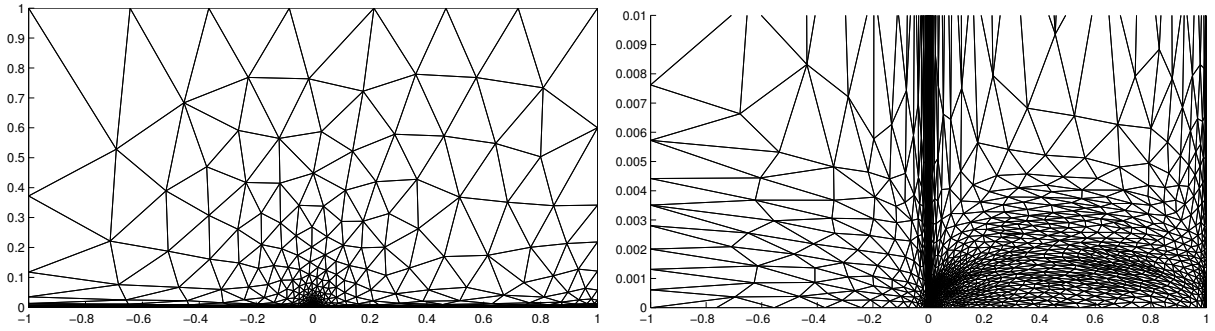
(a) `dof` = 2000 (b) `dof` = 2000 (Zoom)



(c) `dof` = 4000 (d) `dof` = 4000 (Zoom)

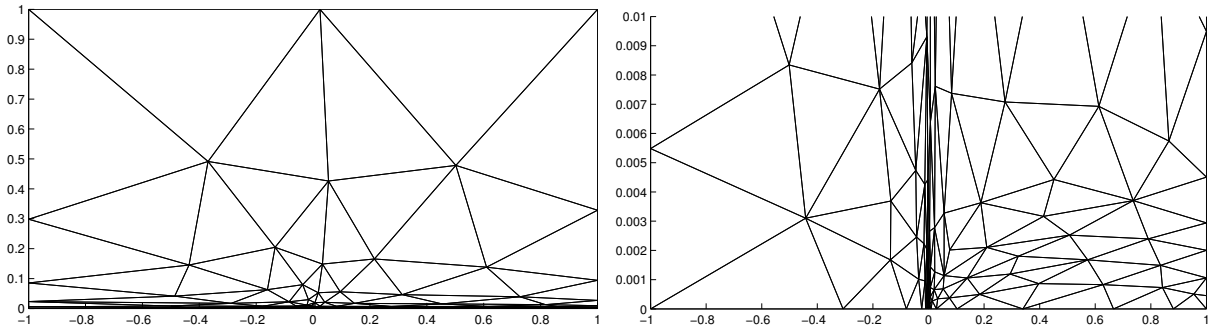


(e) `dof` = 8000 (f) `dof` = 8000 (Zoom)



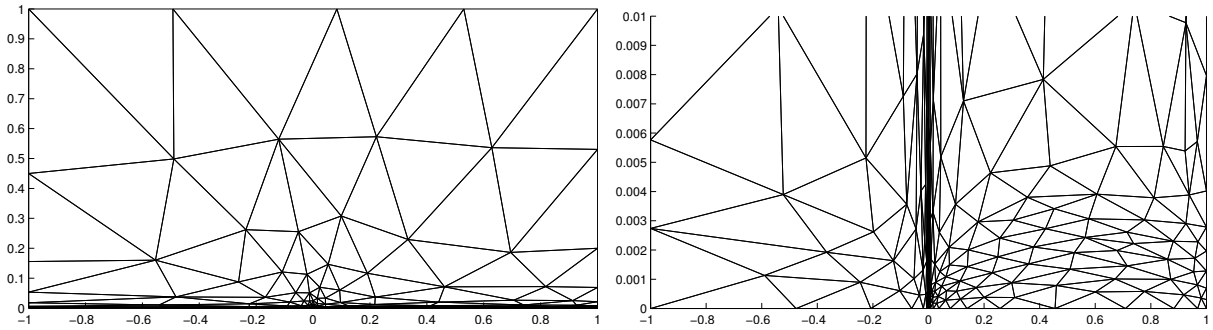
(g) `dof` = 16000 (h) `dof` = 16000 (Zoom)

Figure 12. Laminar flat-plate boundary layer: optimized meshes for $p = 2$ solution approximation. Note the unequal axis scaling in the zoomed figures.



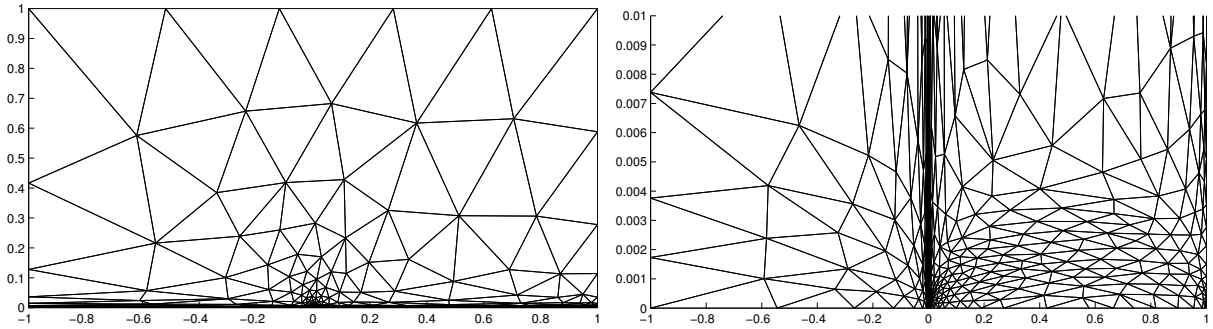
(a) `dof` = 2000

(b) `dof` = 2000 (Zoom)



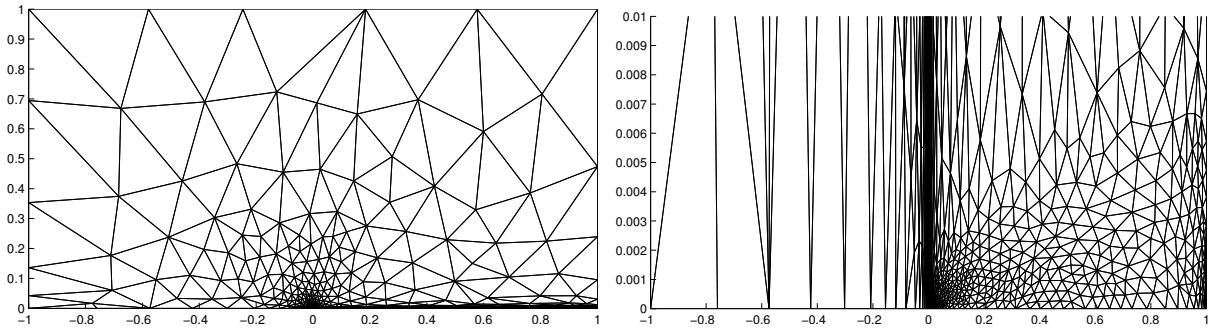
(c) `dof` = 4000

(d) `dof` = 4000 (Zoom)



(e) `dof` = 8000

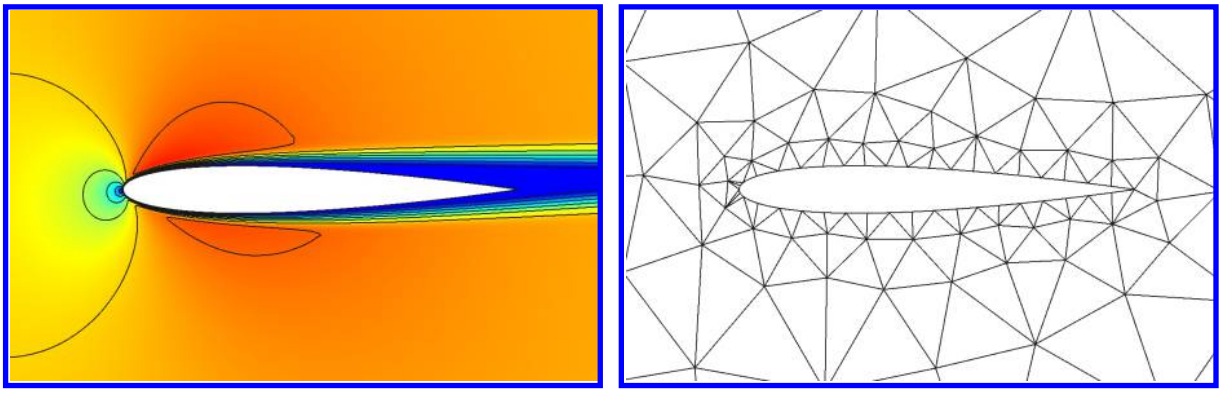
(f) `dof` = 8000 (Zoom)



(g) `dof` = 16000

(h) `dof` = 16000 (Zoom)

Figure 13. Laminar boundary layer: optimized meshes for $p = 3$ solution approximation. Note the unequal axis scaling in the zoomed figures.



(a) Mach number contours (b) Initial mesh (356 triangles)

Figure 14. NACA 0012, $M_\infty = 0.5$, $\alpha = 2°$, $Re = 5000$: Mach number contours and the initial coarse mesh for metric-based mesh optimization.

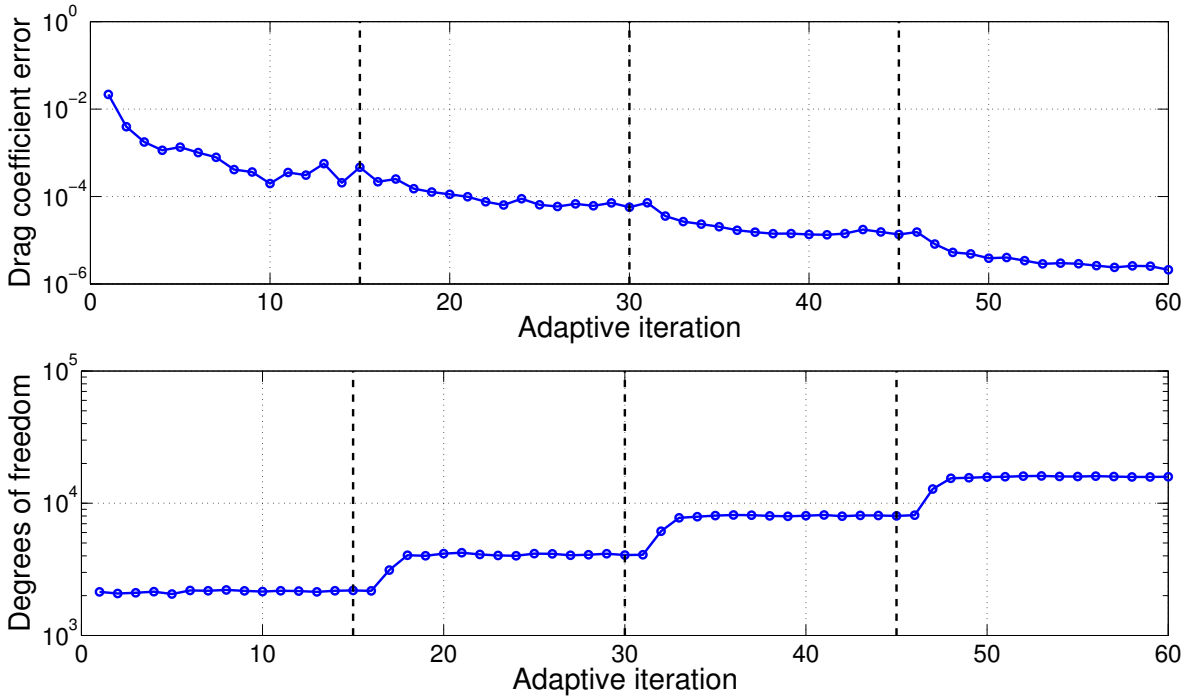


Figure 15. NACA 0012, $M_\infty = 0.5$, $\alpha = 2°$, $Re = 5000$: output error and degrees of freedom histories for mesh optimization with $p = 2$ approximation and 15 solution iterations at each target `dof` value.

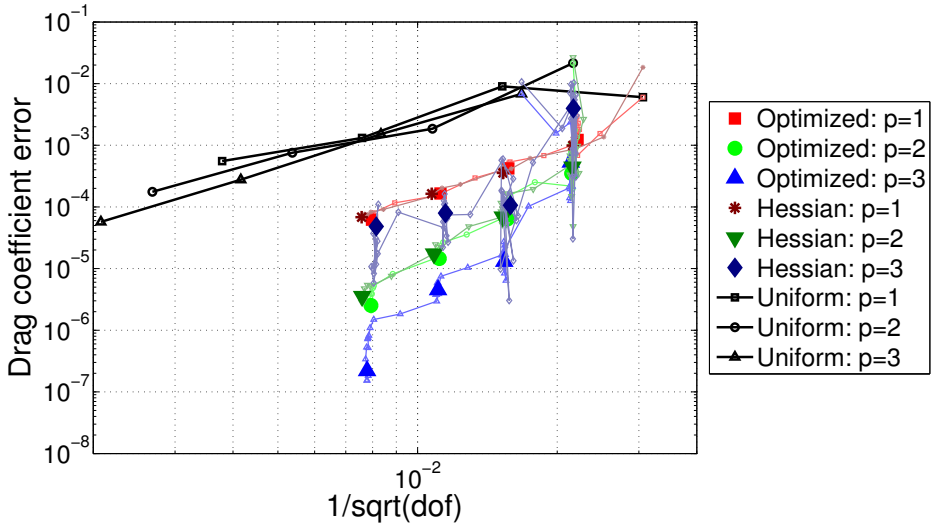
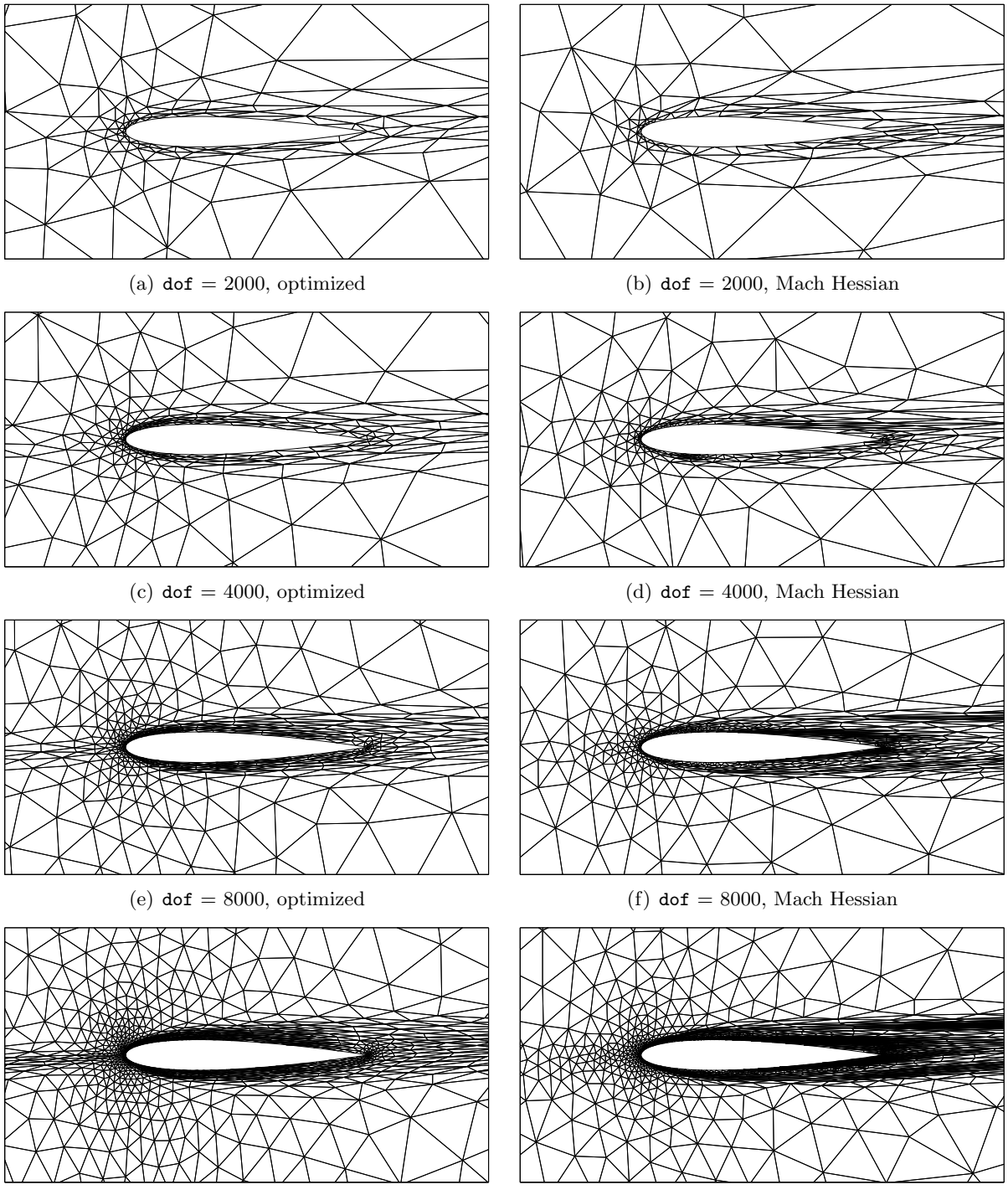


Figure 16. NACA 0012, $M_\infty = 0.5$, $\alpha = 2°$, $Re = 5000$: comparison of output error convergence with `dof` for uniform refinement, adaptation using the Mach number Hessian to determine anisotropy, and adaptation using the mesh optimization algorithm described in this section. The approximation orders are $p = 1, 2, 3$. In the adaptive results, the smaller symbols denote the output at each solution iteration, whereas the larger symbols show the average over the last five iterations at each target `dof`.



(a) `dof` = 2000, optimized

(b) `dof` = 2000, Mach Hessian

(c) `dof` = 4000, optimized

(d) `dof` = 4000, Mach Hessian

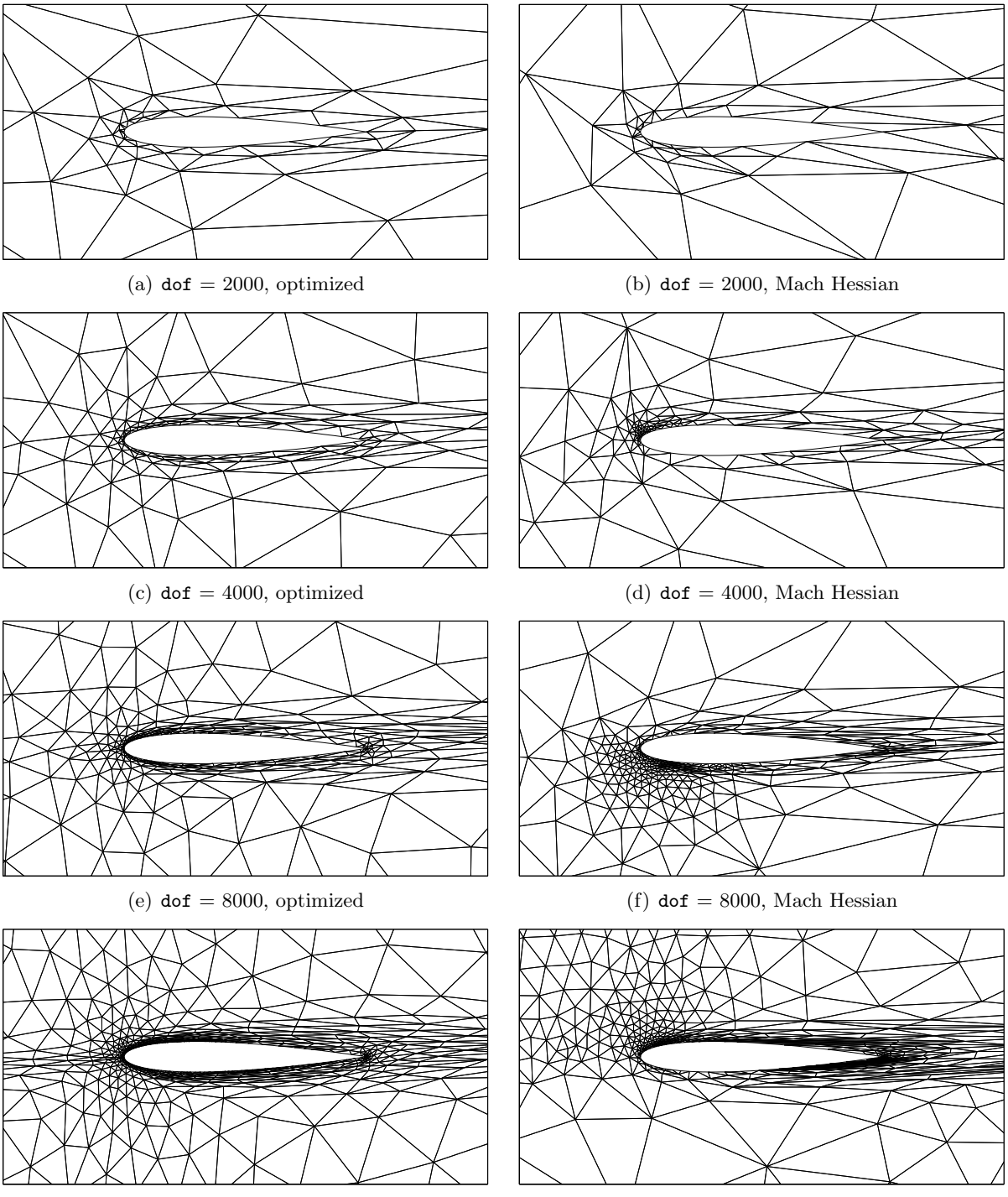
(e) `dof` = 8000, optimized

(f) `dof` = 8000, Mach Hessian

(g) `dof` = 16000, optimized

(h) `dof` = 16000, Mach Hessian

Figure 17. NACA 0012, $M_\infty = 0.5$, $\alpha = 2^\circ$, $Re = 5000$: adapted meshes (last in sequence for each target `dof`) for $p = 2$ using mesh optimization and, for comparison, the Mach number Hessian to set anisotropy.



(a) `dof` = 2000, optimized

(b) `dof` = 2000, Mach Hessian

(c) `dof` = 4000, optimized

(d) `dof` = 4000, Mach Hessian

(e) `dof` = 8000, optimized

(f) `dof` = 8000, Mach Hessian

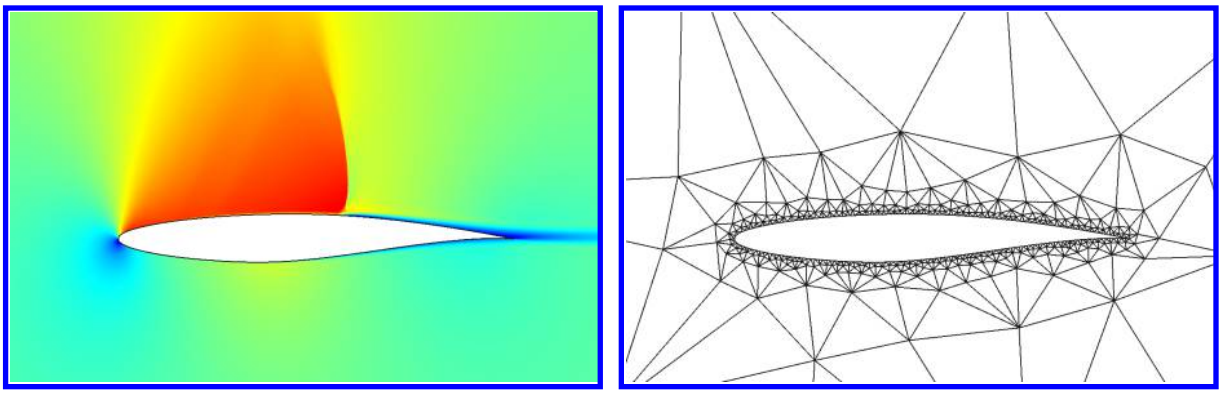
(g) `dof` = 16000, optimized

(h) `dof` = 16000, Mach Hessian

Figure 18. NACA 0012, $M_\infty = 0.5$, $\alpha = 2^\circ$, $Re = 5000$: adapted meshes (last in sequence for each target `dof`) for $p = 3$ using mesh optimization and, for comparison, the Mach number Hessian to set anisotropy.

VI.D. RAE 2822 Airfoil in Transonic Turbulent Flow

The final case is an RAE 2822 airfoil in transonic turbulent flow: $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$. The Spalart-Allmaras[20] closure is used to model the turbulent flow, and element-constant artificial viscosity[21] is used to capture shocks. The output of interest is the drag coefficient. Figure 19 shows the Mach number contours and the initial coarse mesh for this case. Note that the initial mesh is much coarser in the boundary layer than required for accuracy. Curved elements of geometry order $q = 4$ are used adjacent to the airfoil to represent the geometry, and the farfield boundary is over 500 chord-lengths away. A linear elasticity analogy is used to deform anisotropic linear meshes to curve them to the $q = 4$ boundary geometry approximation.



(a) Mach number contours (b) Initial mesh (758 triangles)

Figure 19. RAE 2822, $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$: Mach number contours and the initial coarse mesh for metric-based mesh optimization.

Starting from the coarse mesh, adaptive runs were performed using approximation orders $p = 1, 2, 3$, and at four `dof` targets: 5000, 10000, 20000, and 40000. At each `dof` target, fifteen solution iterations were performed before moving to the next `dof` target. Figure 20 shows a sample run using $p = 2$. The error drops more slowly in this case compared to the inviscid and laminar cases, and the behavior of the error from mesh to mesh is more oscillatory. This is likely due to the highly nonlinear nature of the RANS equations and the sensitivity of the output to anisotropy in the boundary layer.

Figure 21 shows the convergence of the drag coefficient error with respect to the mesh size, $\texttt{dof}^{-1/2}$, when using the presented mesh optimization algorithm, and when using uniform refinement. Here we see a dramatic benefit from the optimized meshes. Uniform refinement of the coarse mesh does not appear to consistently reduce the error at any of the orders – the refined meshes are still not fine enough to resolve the boundary layer. On the other hand, the adapted meshes show excellent error reduction: almost four orders of magnitude lower error on the finest meshes compared to the error on the initial mesh. The differences in the adapted results among the three orders are minor, with slight efficiency gains for $p = 2$ and $p = 3$ at the lower error levels.

Finally, Figures 22, 23, 24 show the adapted meshes for the different orders and `dof` targets. These meshes are much different from the initial one: we observe high levels of anisotropy in the thin boundary layer adjacent to the airfoil, and in the wake and leading-edge stagnation streamline resolution. We also see some refinement in the foot of the shock on the upper surface, and resolution of the "λ" feature of the adjoint solution above the airfoil. Note that resolution of the entire shock is not needed for accurate drag calculation: the important regions are the foot of the shock and the characteristics that connect to the shock-boundary layer junction.

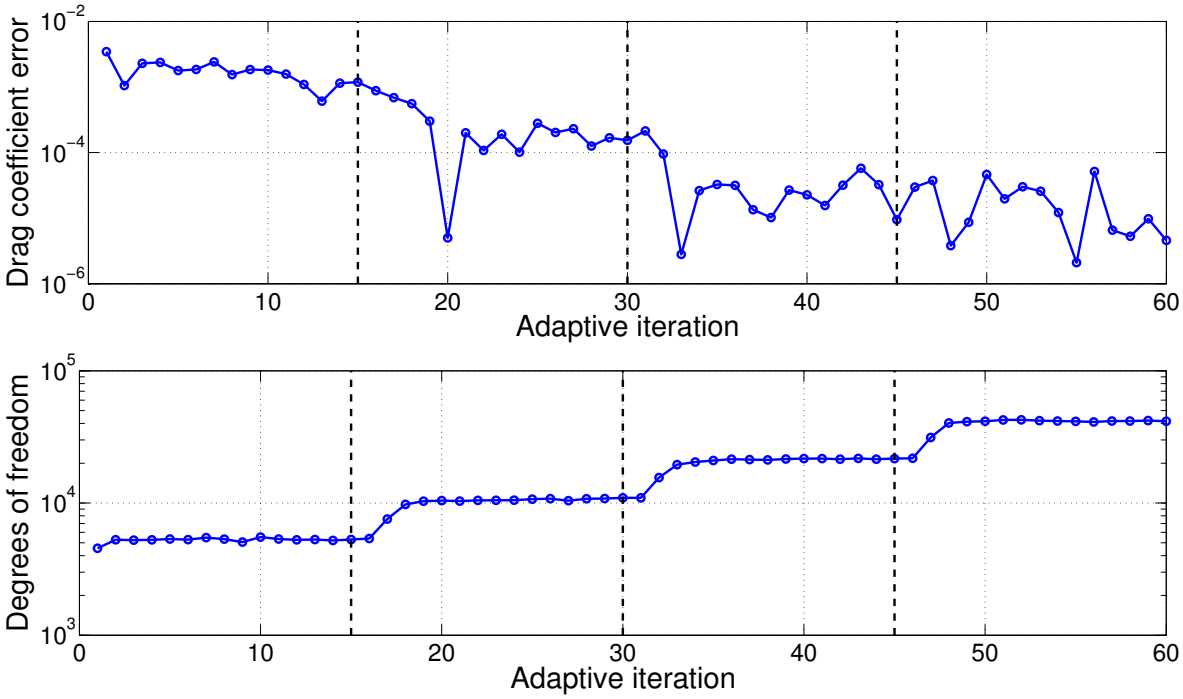


Figure 20. RAE 2822, $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$: output error and degrees of freedom histories for mesh optimization with $p = 2$ approximation and 15 solution iterations at each target `dof` value.

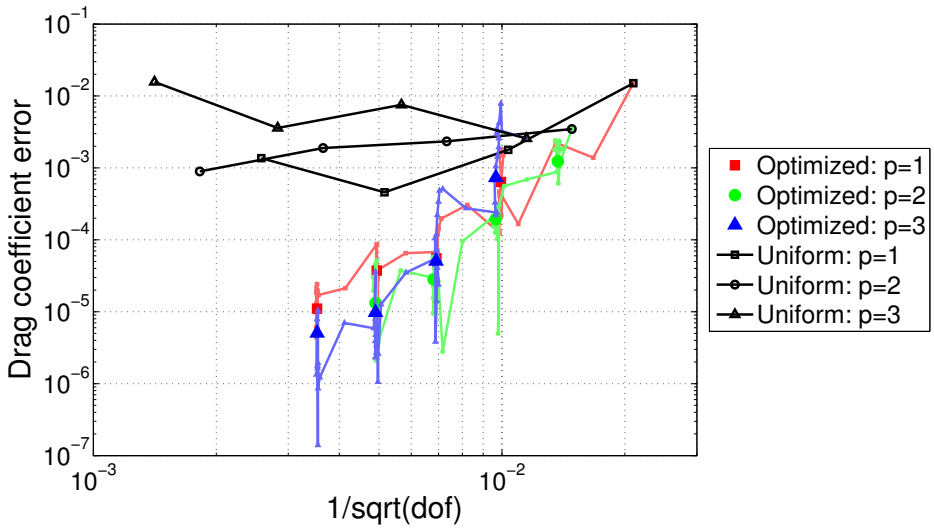
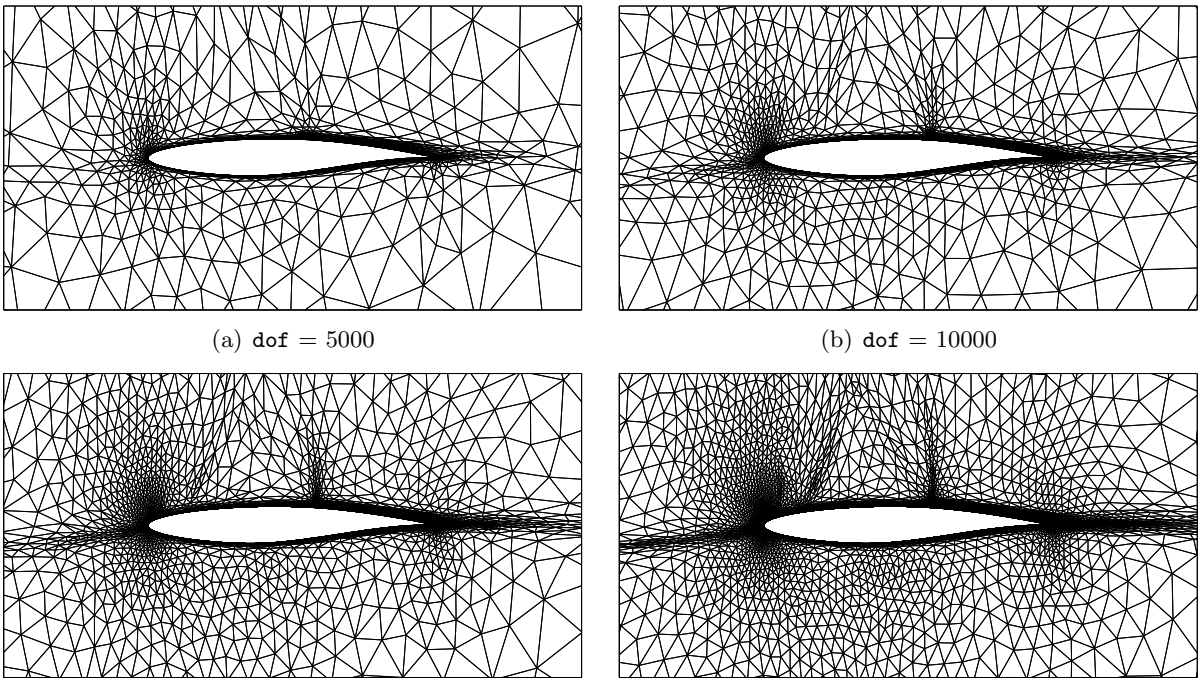


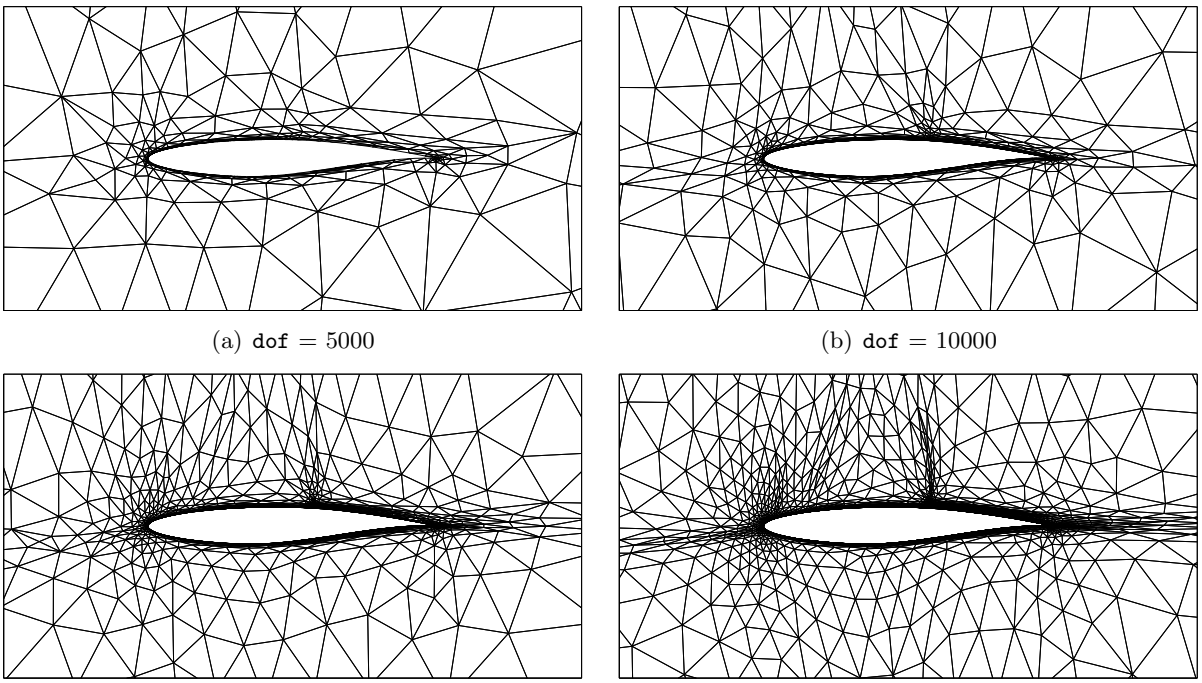
Figure 21. RAE 2822, $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$: comparison of output error convergence with `dof` for uniform refinement and adaptation using the mesh optimization algorithm described in this section. The approximation orders are $p = 1, 2, 3$. In the adaptive results, the smaller symbols denote the output at each solution iteration, whereas the larger symbols show the average over the last five iterations at each target `dof`.



(a) dof = 5000 (b) dof = 10000

(c) dof = 20000 (d) dof = 40000

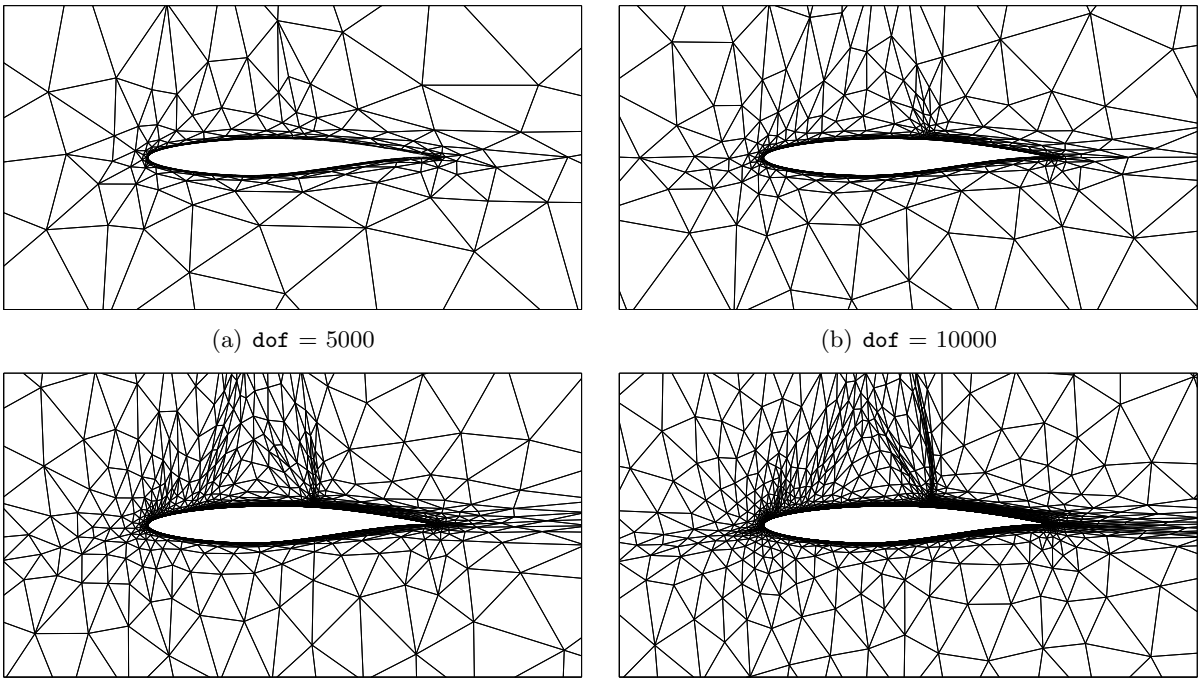
Figure 22. RAE 2822, $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$: optimized meshes (last in sequence for each target dof) for $p = 1$.



(a) dof = 5000 (b) dof = 10000

(c) dof = 20000 (d) dof = 40000

Figure 23. RAE 2822, $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$: optimized meshes (last in sequence for each target dof) for $p = 2$.



(a) `dof` = 5000 (b) `dof` = 10000

(c) `dof` = 20000 (d) `dof` = 40000

Figure 24. RAE 2822, $M_\infty = 0.734$, $\alpha = 2.79°$, $Re = 6.5 \times 10^6$: optimized meshes (last in sequence for each target `dof`) for $p = 3$.

VII. Conclusions

This paper presents a mesh optimization algorithm that builds on the work of Yano.[1] The new contribution is a local adjoint projection approach to sampling errors associated with different element refinement options. The advantage of this approach is that the mesh does not actually need to be refined in order to obtain an estimate of the error after refinement. That is, no primal solves, adjoint solves, or residual evaluations are required on refined meshes. Instead, all calculations take place on the original mesh, in the $p+1$ fine space. A single projection operator for each refinement option maps the fine-space adjoint to an adjoint that reflects the extra resolution introduced by a refinement option. This improves efficiency and simplifies the implementation and mesh/data handling. Results for a variety of test cases demonstrate that the mesh optimization performs well qualitatively, in creating resolution and anisotropy where expected, and quantitatively, in comparison to uniform refinement and Hessian-based strategies.

Acknowledgments

The author acknowledges support from the Department of Energy under grant DE-FG02-13ER26146/DE-SC0010341.

References

[1]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.

[2]Borouchaki, H., George, P., Hecht, F., Laug, P., and Saltel, E., "Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes," INRIA-Rocquencourt, France. Tech Report No. 2741, 1995.

[3]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.

[4]Castro-Diaz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic unstructured mesh adaptation for flow simulations," *International Journal for Numerical Methods in Fluids*, Vol. 25, 1997, pp. 475–491.

[5]Baker, T. J., "Mesh Adaptation Strategies for Problems in Fluid Dynamics," *Finite Elements in Analysis and Design*, Vol. 25, 1997, pp. 243–273.

[6]Buscaglia, G. C. and Dari, E. A., "Anisotropic mesh optimization and its application in adaptivity," *International Journal for Numerical Methods in Engineering*, Vol. 40, No. 22, November 1997, pp. 4119–4136.

[7]Habashi, W. G., Dompierre, J., Bourgault, Y., Ait-Ali-Yahia, D., Fortin, M., and Vallet, M.-G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles," *International Journal for Numerical Methods in Fluids*, Vol. 32, 2000, pp. 725–744.

[8]Venditti, D. A. and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22–46.

[9]Park, M. A., "Three–Dimensional Turbulent RANS Adjoint–Based Error Correction," AIAA Paper 2003-3849, 2003.

[10]Fidkowski, K. J. and Darmofal, D. L., "A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 225, 2007, pp. 1653–1672.

[11]Yano, M., Modisette, J., and Darmofal, D., "The Importance of mesh adaptation for higher-order discretizations of aerodynamics flows," AIAA Paper 2011-3852, 2011.

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

[13]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.

[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, 2005, pp. 92–113.

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

[16]Bassi, F. and Rebay, S., "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 197–207.

[17]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.

[18]Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *American Institute of Aeronautics and Astronautics Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[19]Fidkowski, K., "High-Order Output-Based Adaptive Methods for Steady and Unsteady Aerodynamics," *37^{th} Advanced CFD Lectures series; Von Karman Institute for Fluid Dynamics (December 9–12, 2013)*, edited by H. Deconinck and R. Abgrall, von Karman Institute for Fluid Dynamics, 2013.

[20]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.

[21]Persson, P.-O. and Peraire., J., "Sub-cell shock capturing for discontinuous Galerkin methods," AIAA Paper 2006-112, 2006.