Output-Based Mesh Optimization for the Embedded Discontinuous Galerkin Method

Krzysztof J. Fidkowski[*]

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

This paper presents a method for optimizing computational meshes for the prediction of scalar output quantities when using the embedded discontinuous Galerkin (EDG) discretization. EDG offers memory and computational time advantages compared to the standard discontinuous Galerkin (DG) method through its decoupling of elemental degrees of freedom and the introduction of continuous face degrees of freedom that become the only globally-coupled unknowns. However, the additional equations of weak flux continuity on each interior face introduce new residuals that augment output error estimates and complicate existing element-centric mesh optimization methods. This work presents a technique for converting face-based error estimates to elements and sampling their reduction with refinement in order to determine element-specific anisotropic convergence rate tensors. The sampling uses fine-space adjoint projections and does not require any additional solves on subelements. Together with a degree-of-freedom cost model, this EDG error model drives metric-based unstructured mesh optimization. Adaptive results for inviscid and viscous two-dimensional flow problems demonstrate (1) improvement of EDG mesh optimality when using the new error model compared to one that does not incorporate face errors, and (2) degree of freedom and computational-time benefits of EDG relative to DG.

I. Introduction

Although discontinuous Galerkin (DG) methods[1,2] have enabled high-order accurate computational fluid dynamics simulations, their memory footprint and computational costs remain large. Two approaches for reducing the expense of DG are (1) modifying the discretization; and (2) optimizing the computational mesh. In this work we pursue both approaches and compare their relative benefits.

Hybridization of DG[3–7] is an approach that modifies the high-order discretization to reduce its expense for a given mesh. The high cost of DG arises from the large number of degrees of freedom required to approximate an element-wise discontinuous high-order polynomial solution. Furthermore, these degrees of freedom are globally-coupled, increasing the memory requirements for solvers that require storage of the residual Jacobian matrix, even with an element-compact stencil. Hybridized discontinuous Galerkin (HDG) methods reduce the number of globally-coupled degrees of freedom by decoupling element solution approximations and stitching them together through weak flux continuity enforcement. HDG methods introduce face unknowns that become the only globally-coupled degrees of freedom in the system. Since the number of face unknowns scales as $p^{\dim -1}$ compared to the $p^{\dim}$ scaling for elements, HDG methods can be computationally cheaper and use less memory compared to DG. The embedded discontinuous Galerkin (EDG)

[*]Associate Professor, AIAA Senior Member

method[6,8] is a particular type of HDG method in which the approximation space of face unknowns is continuous, further reducing the number of globally-coupled degrees of freedom.

Another approach to reducing the cost of high-order simulations is mesh optimization. In finite-element discretizations, the number of elements affects the cost and accuracy of the simulations. A mesh is considered optimal if it delivers the highest possible accuracy with the fewest possible elements. Much work has been done in this area, including heuristic,[9–12] semi-heuristic,[13–16] and more recently, rigorous[17] techniques. Of particular interest in engineering applications are methods which directly address accuracy of scalar outputs, and such output-based methods have also been extensively studied in the context of CFD.[13,18–25]

This work introduces a mesh optimization approach for EDG discretizations, effectively combining the two cost-reduction approaches. In addition to reducing computational costs, the resulting adaptive method improves (1) robustness of the solution through quantitative error estimates, and (2) robustness of the solver through a mesh size continuation approach in which the problem is solved on successively finer meshes.

The outline for the remainder of this paper is as follows. Section II presents the DG and EDG discretizations. Section III derives adjoint-based error estimates, which drive discretization-specific mesh optimization techniques that are presented in Section IV. Section V demonstrates the adaptive method for selected two-dimensional flows, and Section VI concludes with a summary and a discussion of future directions.

II. Discretization

We simulate the compressible Navier-Stokes equations,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \nabla \mathbf{u}) \quad = \quad \mathbf{0}, \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^s = [\rho, \rho\vec{v}, \rho E]^T$ is the conservative state vector of rank s, and $\vec{\mathbf{H}}(\mathbf{u}, \nabla\mathbf{u}) = \vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \nabla\mathbf{u})$ is the total flux, consisting of the convective and viscous components. The viscous flux is assumed linear in the state gradients, $\mathbf{G}_i(\mathbf{u}, \nabla\mathbf{u}) = -\mathbf{K}_{ij}(\mathbf{u})\,\partial_j \mathbf{u}$. Presently we consider the steady-state equations, $\frac{\partial \mathbf{u}}{\partial t} = \mathbf{0}$.

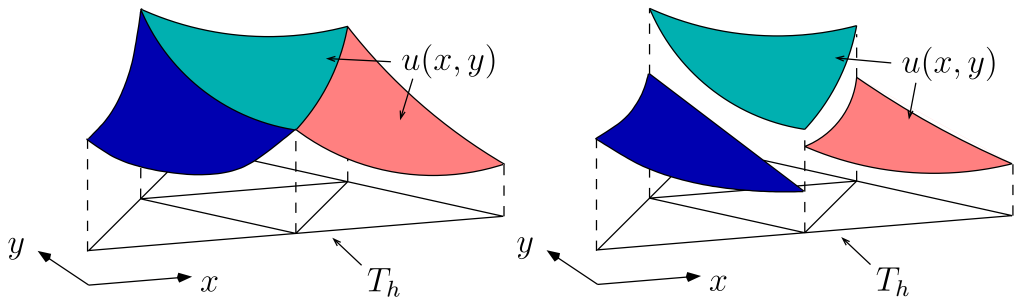
II.A. Discontinuous Galerkin (DG)

Denote by T_h the set of N_{elem} elements in a non-overlapping tessellation of the domain Ω. As shown in Figure 1(b), in DG, the state is approximated by polynomials of order p on each element, with no continuity constraints imposed on the approximations on adjacent elements. Formally, $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where $\mathcal{V}_h = \{u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \; \forall \Omega_e \in T_h\}$, and $\mathcal{P}^p$ denotes polynomials of order p on the reference space of element Ω_e. The weak form of Eqn. 1 follows from multiplying the equation by test functions in the same approximation space, integrating by parts, and coupling elements via unique fluxes,

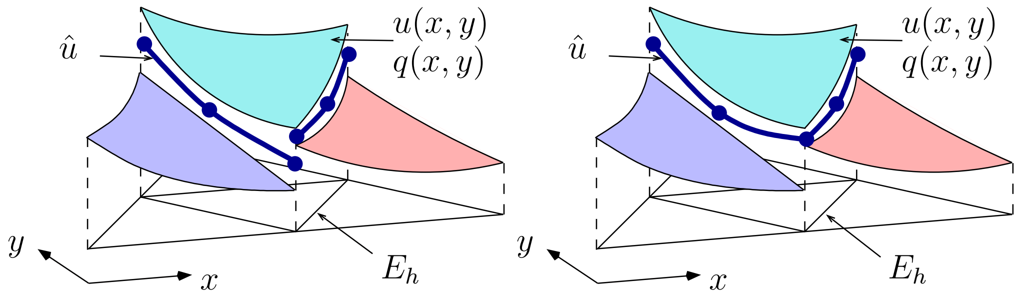
$$-\int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \vec{\mathbf{H}}\, d\Omega + \int_{\partial\Omega_e} \mathbf{w}_h^T \widehat{\mathbf{H}} \cdot \vec{n}\, ds - \int_{\partial\Omega_e} \partial_i \mathbf{w}_h^{+T} \mathbf{K}_{ij}^{+} \left(\mathbf{u}_h^{+} - \widehat{\mathbf{u}}_h\right) n_j\, ds \quad = \quad 0 \quad \forall \mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h, \tag{2}$$

where $(\cdot)^T$ denotes transpose, and on the element boundary $\partial\Omega_e$, $(\cdot)^+, (\cdot)^-$ denote quantities taken from the element or its neighbor, respectively. The last term symmetrizes the semilinear form for adjoint consistency. The unique state on an interior face is $\widehat{\mathbf{u}}_h = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$.

$\widehat{\mathbf{H}} \cdot \vec{n}$ denotes the unique normal flux on faces. We use the Roe approximate Riemann solver[26] for the convective flux, and the second form of Bassi and Rebay (BR2)[27] for the viscous flux. Choosing



(a) Continuous Galerkin (b) Discontinuous Galerkin



(c) Hybridized Discontinuous Galerkin (d) Embedded Discontinuous Galerkin

Figure 1. Schematic description of the solution approximation using various high-order methods.

a basis for the test and trial spaces yields a system of nonlinear equations,

$$\mathbf{R}(\mathbf{U}) \quad = \quad \mathbf{0}, \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{N_h}$ is the discrete state vector of basis function coefficients, N_h is the number of unknowns, and $\mathbf{R}$ is the discrete steady residual vector.

II.B. Embedded Discontinuous Galerkin (EDG)

The starting point for the EDG discretization is the conversion of Eqn. 1 to a system of first-order equations,

$$\vec{\mathbf{q}} - \nabla \mathbf{u} \quad = \quad \vec{\mathbf{0}}, \tag{4}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \vec{\mathbf{q}}) \quad = \quad \mathbf{0}, \tag{5}$$

where $\vec{\mathbf{q}}$ is an approximation of the state gradient. Multiplying these two equations by test functions $\vec{\mathbf{v}} \in [\boldsymbol{\mathcal{V}}_h]^{\mathrm{dim}}, \mathbf{w} \in \boldsymbol{\mathcal{V}}_h$ and integrating by parts over an element Ω_e yields the weak form: we seek $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h$, and $\vec{\mathbf{q}} \in [\boldsymbol{\mathcal{V}}_h]^{\mathrm{dim}}$, such that

$$\int_{\Omega_e} \vec{\mathbf{v}}_h^T \cdot \vec{\mathbf{q}}_h \, d\Omega + \int_{\Omega_e} \nabla \cdot \vec{\mathbf{v}}_h^T \mathbf{u}_h \, d\Omega - \int_{\partial\Omega_e} \vec{\mathbf{v}}_h^T \cdot \vec{n} \; \widehat{\mathbf{u}}_h \, ds \quad = \quad 0 \quad \forall \vec{\mathbf{v}}_h \in [\boldsymbol{\mathcal{V}}_h]^{\mathrm{dim}}, \tag{6}$$

$$\int_{\Omega_e} \mathbf{w}_h^T \frac{\partial \mathbf{u}_h}{\partial t} \, d\Omega - \int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \vec{\mathbf{H}} \, d\Omega + \int_{\partial\Omega_e} \mathbf{w}_h^T \widehat{\mathbf{H}} \cdot \vec{n} \, ds \quad = \quad 0 \quad \forall \mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h, \tag{7}$$

where $\widehat{\mathbf{u}}$ is a new independent unknown: the state approximated on faces of the mesh. Note that through Eqn. 7, element degrees of freedom are coupled to the face degrees of freedom, but not to each other. The additional unknowns call for additional equations, which arise from weak enforcement of flux continuity across faces,

$$\int_{\sigma_f} \boldsymbol{\mu}^T \left\{ \widehat{\mathbf{H}} \cdot \vec{n}\big|_L + \widehat{\mathbf{H}} \cdot \vec{n}\big|_R \right\} ds \quad = \quad 0 \qquad \forall \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}_h. \tag{8}$$

In this equation, $\boldsymbol{\mathcal{M}}_h$ denotes the order-p approximation space on the faces $\sigma_f \in F_h$ of the mesh: $\boldsymbol{\mathcal{M}}_h = [\mathcal{M}_h]^s$, where $\mathcal{M}_h = \left\{ u \in L_2(\sigma_f) : u|_{\sigma_f} \in \mathcal{P}^p \; \forall \sigma_f \in F_h \right\}$, and the subscripts L and R refer to the left and right sides of a face. As shown in Figure 1, both HDG and EDG introduce $\widehat{\mathbf{u}}_h$, with the key difference that in EDG, the approximation space $\boldsymbol{\mathcal{M}}_h$ is continuous at mesh nodes (and edges in three dimensions). This leads to a large reduction in the number of degrees of freedom for face approximations in EDG versus HDG.

The fluxes in Eqn. 7 are one-sided, meaning that they depend only on the state and gradient inside the element, and the face state,

$$\widehat{\mathbf{H}} \cdot \vec{n} = \vec{\mathbf{H}}(\hat{\mathbf{u}}, \vec{\mathbf{q}}) \cdot \vec{n} + \boldsymbol{\tau}(\hat{\mathbf{u}}, \mathbf{u}, \vec{n}), \qquad \boldsymbol{\tau} = \left| \frac{\partial}{\partial \mathbf{u}} (\widehat{\mathbf{F}} \cdot \vec{n}) \right| (\mathbf{u} - \hat{\mathbf{u}}) + \eta \vec{\boldsymbol{\delta}} \cdot \vec{n}. \tag{9}$$

Note that $\boldsymbol{\tau}$ consists of a Roe-like convective stabilization and a BR2 viscous stabilization,[28] where η is set to the number of faces and $\vec{\boldsymbol{\delta}}$ is the BR2 auxiliary variable driven by the state jump $\mathbf{u} - \hat{\mathbf{u}}$.

Choosing bases for the trial/test spaces in Eqns. 6, 7, 8 gives a nonlinear system of equations,

$$\mathbf{R}^Q = \mathbf{0}, \quad \mathbf{R}^U = \mathbf{0}, \quad \mathbf{R}^\Lambda = \mathbf{0}, \tag{10}$$

with the Newton update system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{Q} \\ \Delta \mathbf{U} \\ \Delta \boldsymbol{\Lambda} \end{bmatrix} + \begin{bmatrix} \mathbf{R}^Q \\ \mathbf{R}^U \\ \mathbf{R}^\Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{11}$$

where $\mathbf{Q}$, $\mathbf{U}$, and $\boldsymbol{\Lambda}$ are the discrete unknowns in the approximation of $\vec{\mathbf{q}}$, $\mathbf{u}$, and $\hat{\mathbf{u}}$, respectively. $[\mathbf{A}, \mathbf{B}; \mathbf{C}, \mathbf{D}]$ is the primal Jacobian matrix partitioned into element-interior and interface unknown blocks. Note that $\mathbf{A}$ and $\mathbf{B}$ contain both $\mathbf{Q}$ and $\mathbf{U}$ components. In addition, $\mathbf{A}$ is element-wise block diagonal, and hence easily invertible using element-local operations.

Statically condensing out the element-interior states gives a smaller system for the face degrees of freedom,

$$\underbrace{\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)}_{\boldsymbol{\mathcal{K}}} \Delta \boldsymbol{\Lambda} + \left(\mathbf{R}^\Lambda - \mathbf{C}\mathbf{A}^{-1} \left[\mathbf{R}^Q; \mathbf{R}^U\right]\right) = \mathbf{0}. \tag{12}$$

Solving this set of equations constitutes the global solve of the problem. Following the global solve for $\Delta \boldsymbol{\Lambda}$, an element-local back-solve yields the updates to $\mathbf{Q}$ and $\mathbf{U}$,

$$\begin{bmatrix} \Delta \mathbf{Q} \\ \Delta \mathbf{U} \end{bmatrix} = -\mathbf{A}^{-1} \left(\begin{bmatrix} \mathbf{R}^Q \\ \mathbf{R}^U \end{bmatrix} + \mathbf{B} \Delta \boldsymbol{\Lambda} \right).$$

II.C. Adjoint Discretization

For a scalar output J, the discrete adjoint $\boldsymbol{\Psi}$ is a vector of sensitivities of J to residual source perturbations. For DG, these perturbations refer to Eqn. 3, and the associated adjoint equation is

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \right)^T \boldsymbol{\Psi} + \left(\frac{\partial J}{\partial \mathbf{U}} \right)^T = \mathbf{0}. \tag{13}$$

For EDG, residual perturbations refer to Eqn. 10, and with three sets of residuals, the analog of Eqn. 13 is

$$\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}^Q \\ \mathbf{\Psi}^U \\ \mathbf{\Psi}^\Lambda \end{bmatrix} + \begin{bmatrix} (\partial J/\partial \mathbf{Q})^T \\ (\partial J/\partial \mathbf{U})^T \\ (\partial J/\partial \mathbf{\Lambda})^T \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{14}$$

Statically condensing out the element-interior adjoints gives a smaller system for the face adjoints,

$$\underbrace{\left(\mathbf{D}^T - \mathbf{B}^T\mathbf{A}^{-T}\mathbf{C}^T\right)}_{\boldsymbol{\mathcal{K}}^T} \mathbf{\Psi}^\Lambda + \left(\frac{\partial J}{\partial \mathbf{\Lambda}}^T - \mathbf{B}^T\mathbf{A}^{-T} \left[\frac{\partial J}{\partial \mathbf{Q}}^T ; \frac{\partial J}{\partial \mathbf{U}}^T \right] \right) = \mathbf{0}. \tag{15}$$

Note that the operator appearing in this equation is the transpose of the primal operator in Eqn. 12. After solving this global system for $\mathbf{\Psi}^\Lambda$, $\mathbf{\Psi}^Q$ and $\mathbf{\Psi}^U$ follow from an element-local back-solve.

II.D. Degrees of Freedom and Matrix Sparsity

On a given mesh, the DG, HDG, and EDG discretizations will have different degree of freedom counts and residual Jacobian sparsity patterns. Figure 2 presents an example of the degree of freedom placement for $p = 2$ approximation on a ten-element mesh of triangles.

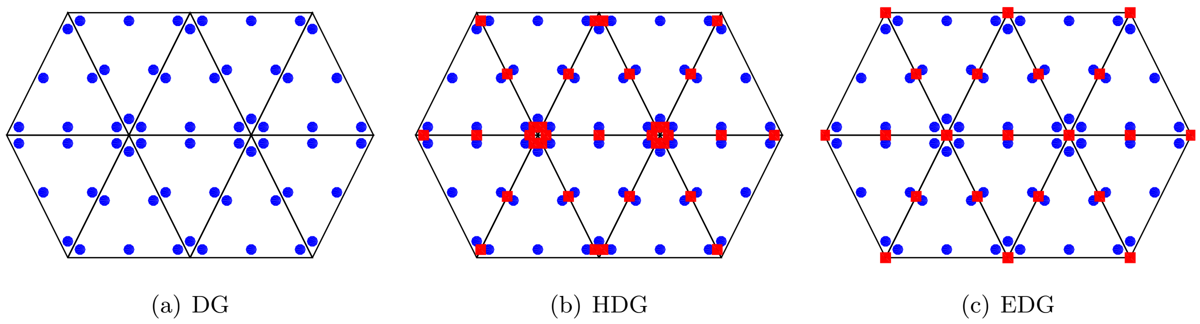


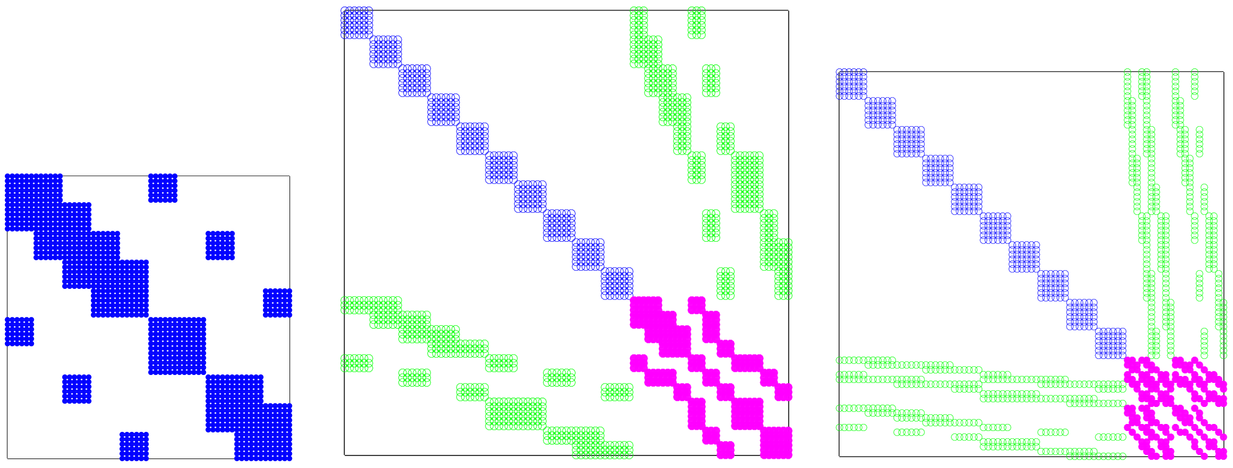
Figure 2. Element (blue) and face (red) degree of freedom placement for a sample mesh, using various discretizations.

In HDG and EDG, we do not introduce $\hat{\mathbf{u}}$ on boundary faces, as the flux there is computed in the same way as in DG. Figure 3 shows the resulting residual Jacobian matrices, with the static condensation applied to HDG and EDG. The number of nonzeros refers to the globally-coupled (condensed) matrices. Note that the number of matrix nonzeros for EDG is about a factor of 6 smaller than for DG.

III. Output Error Estimation

III.A. The Adjoint-Weighted Residual

An adjoint solution can be used to estimate the numerical error in the corresponding output of interest, J, through the adjoint-weighted residual.[19,22] Let H denote a coarse/current discretization space, and h a fine one, e.g. obtained by increasing the approximation order by one, $p \to p + 1$. Denote by $\mathbf{U}_h^H$ the state injected from the coarse to the fine space, and similarly for $\mathbf{Q}_h^H$ and $\mathbf{\Lambda}_h^H$ in EDG. Computing the fine-space residuals with these injected states and weighting them by the



(a) DG, nnz=1080 (b) HDG, nnz=330 (c) EDG, nnz=177

Figure 3. Matrix sparsity patterns for the ten-element mesh in Figure 2. Matrix sizes are shown to scale.

fine-space adjoint gives an estimate of the output error between the coarse and fine spaces,

$$\text{DG:} \qquad J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx -\delta\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) \tag{16}$$

$$\text{EDG:} \qquad J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx \underbrace{-(\delta\mathbf{\Psi}_h^Q)^T\mathbf{R}_h^Q}_{\delta J^Q} \underbrace{-(\delta\mathbf{\Psi}_h^U)^T\mathbf{R}_h^U}_{\delta J^U} \underbrace{-(\delta\mathbf{\Psi}_h^\Lambda)^T\mathbf{R}_h^\Lambda}_{\delta J^\Lambda}, \tag{17}$$

where all of the residuals are evaluated using the coarse state injected into the fine space, including $\mathbf{Q}_h^H$ and $\mathbf{\Lambda}_h^H$ for EDG. For the fine space, we increment the approximation order by one on each element and face and obtain the fine-space adjoint by solving exactly on this fine space. We obtain $\delta\mathbf{\Psi}_h$ for use in the error estimates by subtracting from the fine-space adjoint an injection of the coarse-space adjoint. Note that Eqn. 17 separates the error estimate into three components, one for each residual.

III.B. Error Localization

The error estimates involving element residuals can be localized to element (e) contributions, resulting in the error indicators

$$\text{DG:} \qquad \mathcal{E}_e \equiv \left|\delta\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)\right|, \tag{18}$$

$$\text{EDG:} \qquad \mathcal{E}_e^Q \equiv \left|\delta\mathbf{\Psi}_{h,e}^{Q\,T}\mathbf{R}_{h,e}^Q\right|, \quad \mathcal{E}_e^U \equiv \left|\delta\mathbf{\Psi}_{h,e}^{U\,T}\mathbf{R}_{h,e}^U\right|. \tag{19}$$

On the other hand, the error contribution δJ^Λ is associated with an inner product over faces, in the space $\boldsymbol{\mathcal{M}}_h$. For HDG, this error could be localized to faces, but for EDG, the localization is not as simple due to the continuous approximation space $\boldsymbol{\mathcal{M}}_h$.

III.C. EDG Face Error Treatment

The mesh optimization algorithm used in this study works with an element-based error estimate, as the model for the error is based on the size of the elements. We therefore convert the EDG face output error contribution, δJ^Λ, to elements. The starting point for this conversion is writing the absolute value of δJ^Λ as an adjoint-weighted residual integrated over the faces, in variational form,

$$|\delta J^\Lambda| = \Big| \sum_f \int_{\sigma_f} \boldsymbol{\psi}_h^{\Lambda\,T} \left\{ \widehat{\mathbf{H}} \cdot \vec{n}\big|_L + \widehat{\mathbf{H}} \cdot \vec{n}\big|_R \right\} ds \Big|. \tag{20}$$

Defining $\delta \widehat{\mathbf{H}} \cdot \vec{n}\big|_L \equiv \widehat{\mathbf{H}} \cdot \vec{n}\big|_L - \widehat{\mathbf{H}} \cdot \vec{n}\big|_{\text{exact}}$, where "exact" refers to the flux computed from the exact (no numerical error) solution, and similarly for $\delta \widehat{\mathbf{H}} \cdot \vec{n}\big|_R$, we can bound $|\delta J|^\Lambda$ as

$$|\delta J^\Lambda| \le \Big| \sum_f \int_{\sigma_f} \boldsymbol{\psi}_h^{\Lambda\,T} \delta \widehat{\mathbf{H}} \cdot \vec{n}\big|_L \Big| + \Big| \sum_f \int_{\sigma_f} \boldsymbol{\psi}_h^{\Lambda\,T} \delta \widehat{\mathbf{H}} \cdot \vec{n}\big|_R \Big| \le \sum_e \int_{\partial\Omega_e} \big| \boldsymbol{\psi}_h^{\Lambda\,T} \delta \widehat{\mathbf{H}} \cdot \vec{n} \big| \, ds. \tag{21}$$

Using the divergence theorem, we obtain integrals over element interiors, from which we can define an element indicator,

$$|\delta J^\Lambda| \le \sum_e \underbrace{\int_{\Omega_e} \left| \nabla \cdot (\boldsymbol{\psi}_h^{\Lambda\,T} \delta \vec{\mathbf{H}}) \, d\Omega \right|}_{\mathcal{E}_e^\Lambda}. \tag{22}$$

IV. Adaptation

Estimates of the output error not only provide information about the accuracy of a solution, but can also drive mesh adaptation. A fair comparison of DG and EDG requires optimal meshes for each discretization. In previous work, we presented an output-based mesh optimization algorithm for DG,[29] which built on earlier work of Yano.[17] This section describes an extension of this algorithm, Mesh Optimization through Error Sampling and Synthesis (MOESS), to EDG.

IV.A. Mesh Metrics

A Riemannian metric field, $\mathcal{M}(\vec{x}) \in \mathbb{R}^{\text{dim} \times \text{dim}}$, can be used to encode information about the size and stretching of elements in a mesh. 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. The Bi-dimensional Anisotropic Mesh Generator (BAMG)[30] supports metric-based re-meshing and is used to obtain the results in the present work.

The optimization algorithm determines changes to the current, mesh-implied, metric, $\mathcal{M}_0(\vec{x})$, which is calculated for each simplex element by requiring unit measure of its edges under the metric. The element metrics are then averaged to the nodes using an affine-invariant algorithm.[31] Changes to the metric are introduced using a symmetric step matrix, $\mathcal{S} \in \mathbb{R}^{\text{dim} \times \text{dim}}$, according to

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S}) \mathcal{M}_0^{\frac{1}{2}}. \tag{23}$$

IV.B. Error Convergence Models

MOESS requires a model for how the error changes as the metric changes. We use an element-based model that relates the error indicator on element e to the step matrix $\mathcal{S}_e$. For DG, this is[17]

$$\text{DG:} \quad \mathcal{E}_e = \mathcal{E}_{e0} \exp\left[\text{tr}(\mathcal{R}_e \mathcal{S}_e) \right], \tag{24}$$

where $\mathcal{R}_e$ is an element-specific error rate tensor determined through a sampling procedure, as described in Section IV.E. The total error over the mesh is the sum of the elemental errors, $\mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e$.

For EDG, each element contributes to the output error in three ways: through the Q, U, and Λ equations. We define separate models for how the associated three error indicators change with $\mathcal{S}_e$,

$$\text{EDG:}\quad \mathcal{E}_e^U = \mathcal{E}_{e0}^U e^{\mathrm{tr}(R_e^U \mathcal{S}_e)}, \quad \mathcal{E}_e^Q = \mathcal{E}_{e0}^Q e^{\mathrm{tr}(R_e^Q \mathcal{S}_e)}, \quad \mathcal{E}_e^\Lambda = \mathcal{E}_{e0}^\Lambda e^{\mathrm{tr}(R_e^\Lambda \mathcal{S}_e)}, \tag{25}$$

where $\mathcal{R}_e^U$, $\mathcal{R}_e^Q$, and $\mathcal{R}_e^\Lambda$ are element-specific error rate tensors, also identified through sampling. The total error indicator on element e is $\mathcal{E}_e = \mathcal{E}_e^U + \mathcal{E}_e^Q + \mathcal{E}_e^\Lambda$.

IV.C. Cost Model

Mesh refinement reduces error but increases cost, measured by degrees of freedom. These can be the globally-coupled degrees of freedom, which are element-specific for DG and face/edge/node-specific for EDG. Elemental degrees of freedom can also be included in the case of EDG, potentially with a weighting factor, to account for the cost of static condensation and back-solves. In all of these cases, assuming a uniform order p and constant factor relationships between the number of elements and nodes/edges/faces, the total cost is directly proportional to the number of elements, $\mathcal{C} = N_e \mathcal{C}_0$, where $\mathcal{C}_0$ is the cost per element.

When the step matrix S_e is applied to the metric of element e, the area of the element decreases by $\exp\left[\frac{1}{2}\mathrm{tr}(S_e)\right]$. As the number of new occupying the original area Ω_e increases by this factor, the elemental cost model is

$$C_e = C_0 \exp\left[\frac{1}{2}\mathrm{tr}(\mathcal{S}_e)\right]. \tag{26}$$

Note that this cost model remains the same between DG and EDG, with the only difference in the definition of C_0.

IV.D. Metric Optimization Algorithm

The goal of mesh optimization is to determine the step matrix field, $\mathcal{S}(\vec{x})$, that minimizes the error at a given cost. The step matrix field is approximated by values at the mesh vertices, $\mathcal{S}_v$, which are arithmetically-averaged to adjacent elements. The cost only depends on the trace of the step matrix, and we therefore separate the vertex step matrices into trace ($s_v\mathcal{I}$) and trace-free ($\widetilde{\mathcal{S}}_v$) parts, $\mathcal{S}_v = s_v\mathcal{I} + \widetilde{\mathcal{S}}_v$.

The optimization algorithm then consists of the following steps:[17, 29]

1. Given a mesh, solution, and adjoint, calculate the error indicator(s) and rate tensor(s) 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 $\mathcal{S}_e$, $\frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e}$, and $\frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e}$.
   (b) Calculate derivatives of $\mathcal{E}$ and $\mathcal{C}$ with respect to s_v and $\widetilde{\mathcal{S}}_v$.
   (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. 26 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.

This algorithm iteratively equidistributes λ_v globally so that, at optimum, all elements have the same marginal error to cost ratio. User-defined 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.

IV.E. Error Sampling

The error convergence models in Section IV.B rely on convergence tensors, e.g. $\mathcal{R}_e$ for DG, for each element e. We estimate this rate tensor a posteriori by *sampling* a small number of refinements for each element, as shown in Figure 4, and performing a regression. However, as described next and first introduced in our previous work,[29] we never actually modify the mesh when considering the refinement samples, which greatly simplifies the algorithm.

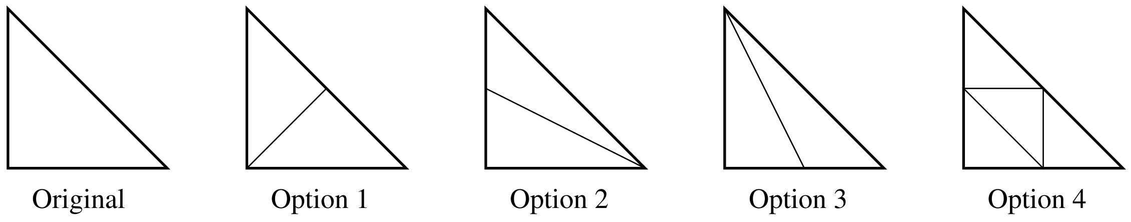


Figure 4. Four refinement options for a triangle, with face refinements highlighted. Each one is considered implicitly during error sampling, though the elements are never actually refined.

Each element refinement is also associated with the refinement of a certain number of adjacent faces. For EDG, these face refinements reduce the error contributions of those faces to the element error indicator $\mathcal{E}_e^{\Lambda}$, per Eqn. 21. Therefore, the same refinements shown in Figure 4 can be used to sample all three error contributions in EDG. To determine how much the error(s) decrease(s) for each refinement option, we use element and face-local projections of the fine-space adjoints to semi-refined spaces associated with each element/face refinement option.

Discontinuous Galerkin: Consider first one element, Ω_e, in a DG discretization. The fine space adjoint, $\mathbf{\Psi}_{h,e}$, provides an estimate of the output error in the current order p solution, as measured relative to the $p+1$ solution: this is $\mathcal{E}_{e0}$. Now, suppose that we are looking at refinement option i in Figure 4: this creates a solution space that is finer than the original, though we assume not as fine as increasing the order to $p+1$. If we have an order p adjoint on this refined space, $\mathbf{\Psi}_{Hi}$, where the i indicates that we are considering refinement option i, we can compute an error indicator $\Delta\mathcal{E}_{ei}$, which 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,

$$\mathcal{E}_{ei} \equiv \mathcal{E}_{e0} - \Delta\mathcal{E}_{ei}. \tag{27}$$

Calculating $\Delta\mathcal{E}_{ei}$ requires an adjoint-weighted residual evaluation on the element refined under option i. To simplify this calculation we project $\mathbf{\Psi}_{Hi}$ back into the $p+1$ space on the original element and evaluate the adjoint weighted residual there. That is, we perform

$$\Delta\mathcal{E}_{ei} \equiv \mathbf{\Psi}_{h,e}^{Hi\,T}\mathcal{R}_{h,e}(\mathbf{U}_h^H), \tag{28}$$

where $\mathbf{\Psi}_h^{Hi}$ is $\mathbf{\Psi}_{Hi}$ projected from order p on refinement option i into order $p+1$ on the original element. The final simplification is that we do not solve for $\mathbf{\Psi}_{Hi}$ but instead project the fine-space $(p+1)$ adjoint to order p under 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. 28, with all calculations occurring at order $p+1$ on the original element.

Using least-squares projections in reference space, the combination of projections can be encapsulated into one transfer matrix that converts $\boldsymbol{\Psi}_h$ into $\boldsymbol{\Psi}_h^{Hi}$, both represented in the order $p+1$ space on the original element:

$$\boldsymbol{\Psi}_h^{Hi} = \mathbf{T}_i \boldsymbol{\Psi}_h, \tag{29}$$

$$\mathbf{T}_i = \left[\mathbf{M}_0(\phi_0^{p+1}, \phi_0^{p+1})\right]^{-1} \sum_{k=1}^{n_i} \mathbf{T}_{ik}, \tag{30}$$

$$\mathbf{T}_{ik} = \mathbf{M}_k(\phi_0^{p+1}, \phi_k^p) \left[\mathbf{M}_k(\phi_k^p, \phi_k^p)\right]^{-1} \mathbf{M}_k(\phi_k^p, \phi_k^{p+1}) \left[\mathbf{M}_k(\phi_k^{p+1}, \phi_k^{p+1})\right]^{-1} \mathbf{M}_k(\phi_k^{p+1}, \phi_0^{p+1}). \tag{31}$$

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

$$\mathbf{M}_k(\phi_l, \phi_m) = \int_{\Omega_k} \phi_l \phi_m d\Omega, \qquad \mathbf{M}_0(\phi_l, \phi_m) = \int_{\Omega_0} \phi_l \phi_m d\Omega, \tag{32}$$

where Ω_k is sub-element k and Ω_0 is the original element. Note that the transfer matrix $\mathbf{T}_i$ 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.

Embedded Discontinuous Galerkin: In EDG, the sampling of error for the calculation of rate tensors for $\mathcal{E}_e^U$ and $\mathcal{E}_e^Q$ proceeds as outlined in the preceding description for DG. For $\mathcal{E}_e^\Lambda$, we refer to the expressions in Eqn. 21 and Eqn. 22. The elemental error indicator $\mathcal{E}_e^\Lambda$ is the sum of adjoint-weighted flux residuals integrated over the faces of Ω_e. Let $\mathcal{E}_f$ be the error indicator associated with one face σ_f, which from Eqn. 20 is

$$\mathcal{E}_f \equiv \Big| \int_{\sigma_f} \psi_h^{\Lambda\,T} \left\{ \widehat{\mathbf{H}} \cdot \vec{n}\big|_L + \widehat{\mathbf{H}} \cdot \vec{n}\big|_R \right\} ds \Big|.$$

Just as for elements, we consider all available refinement options j for a face. For each of these refinement options, we compute $\mathcal{E}_{fj}$, the remaining error associated with refinement option j of face f, using the same adjoint-projection procedure as presented for elements. This requires the calculation of face adjoint transfer matrices, $\mathbf{T}_j^f$, which again is performed in reference space.

By nature of the continuous trial and test space over faces, $\boldsymbol{\mathcal{M}}_h$, the fine-space residuals that are available from the Λ error estimate are not tied to single faces. Rather, they are associated with nodes, edges, or face interiors. In order to leverage during adaptation the fine-space residuals available from error estimation, we distribute these residuals to elements as shown in Figure 5.

Performing such a residual distribution for the unrefined element e yields the baseline error indicator, $\mathcal{E}_{e0}^\Lambda$. Then, for each *element* refinement option i, the errors of refined faces are reduced to $\mathcal{E}_{fj}$ using the results of the face sampling procedure. The residual distribution is then performed again, yielding $\mathcal{E}_{ei}^\Lambda$.

Regression: After calculating $\mathcal{E}_{ei}$, the errors remaining after each refinement option i according to Eqn. 27, we use least-squares regression to estimate the rate tensor $\mathcal{R}_e$. Note that for triangles,

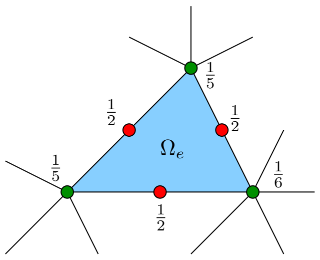


Figure 5. Weights for distributing residuals, and adjoint-weighted residuals from globally-coupled EDG degrees of freedom to elements. Face weights are $\frac{1}{2}$ and node weights are the inverse of the node cardinality.

we have 4 refinement options and 3 independent entries in the symmetric $\mathcal{R}_e$ tensor. Using Eqn. 24, we formulate the regression to minimize the following error, summed over refinement options,

$$\sum_i \left[\log\frac{\mathcal{E}_{ei}}{\mathcal{E}_{e0}} - \mathrm{tr}(\mathcal{R}_e\mathcal{S}_{ei})\right]^2. \tag{33}$$

In this equation, $\mathcal{S}_{ei}$ is the step matrix associated with refinement option i, given by (from Eqn. 23),

$$\mathcal{S}_{ei} = \log\left(\mathcal{M}_0^{-\frac{1}{2}}\mathcal{M}_i\mathcal{M}_0^{-\frac{1}{2}}\right), \tag{34}$$

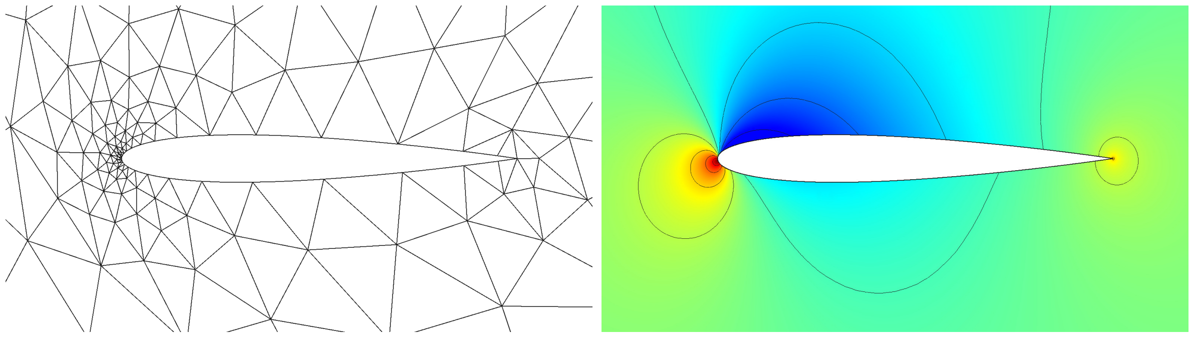
where $\mathcal{M}_i$ is the affine-invariant metric average of the mesh-implied metrics of all sub-elements in refinement option i. Differentiating Eqn. 33 with respect to the independent components of $\mathcal{R}_e$ yields a linear system for these components.

V. Results

This section presents results that compare the performance of the DG and EDG discretizations in an output-based mesh-optimization framework. MOESS adaptations are performed at various orders using a target number of element-interior (U) degrees of freedom.

V.A. Invsicid flow

The first test case consists of inviscid flow over a NACA 0012 airfoil at $\alpha = 2°, M = 0.5$. The output of interest is the drag coefficient. Figure 6 shows the initial mesh and Mach number contours. The mesh is curved to the geometry using a cubic mapping from reference space.



(a) Initial mesh, 326 elements (b) Mach number contours

Figure 6. Inviscid flow test: initial mesh and solution Mach contours.

Figure 7 shows the adaptive convergence of the output with number of elements for the two discretizations. Two sets of EDG results are shown: one driven by the $\mathcal{E}_e^U$ error indicator only, and one driven by the complete, edge-based (including $\mathcal{E}_e^\Lambda$) error indicator (note, "face-based" becomes "edge-based" in 2D). The mesh optimizations in each case were run for 20 adaptive iterations, starting with the initial mesh, for each target number of elements. The data points shown are the average errors and costs over the last six optimization iterations.

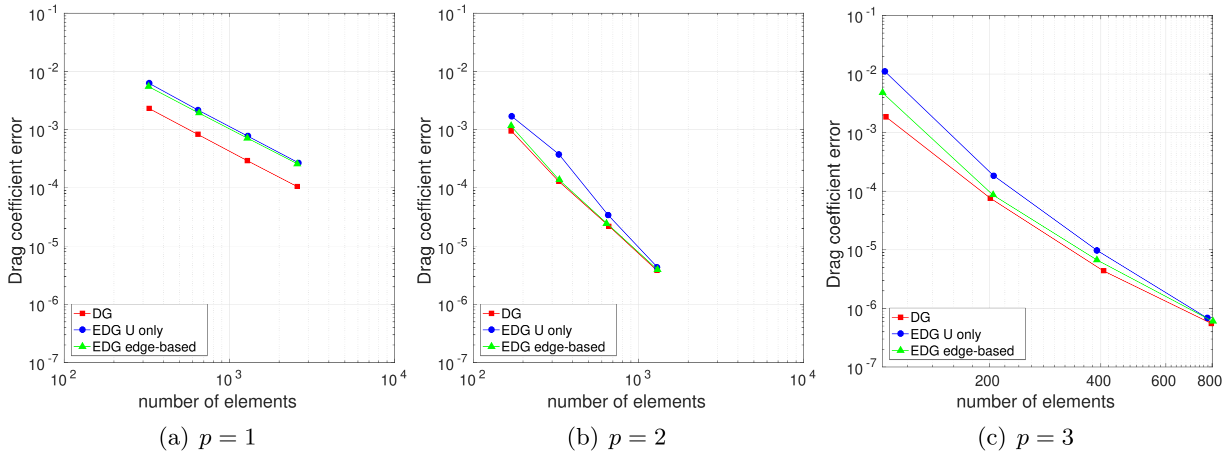


Figure 7. Inviscid flow test: output convergence results versus number of elements.

For $p = 1$, DG is consistently more accurate than EDG for a given number of elements. In addition, the difference between the two EDG adaptive indicators is small: both yield errors that are approximately three times larger than those of DG. At $p = 2$, the edge-based EDG results close the gap relative to DG, and the adaptive results are nearly identical. However, the U-only EDG adaptive indicator lags behind and has larger errors. $p = 3$ yields similar conclusions, with a slightly higher gap between edge-based EDG and DG, and differences that diminish for finer meshes.

The comparison versus number of elements does not accurately convey the cost of EDG relative to DG. The most expensive part of the solver is the inversion of the global Jacobian matrix, and here EDG has an advantage over DG, as it has fewer globally-coupled degrees of freedom for a given number of elements. Figure 8 presents the same adaptive convergence data in terms of globally-coupled degrees of freedom. For a given error level, the EDG simulations are cheaper than DG by approximately a factor of three in degrees of freedom.

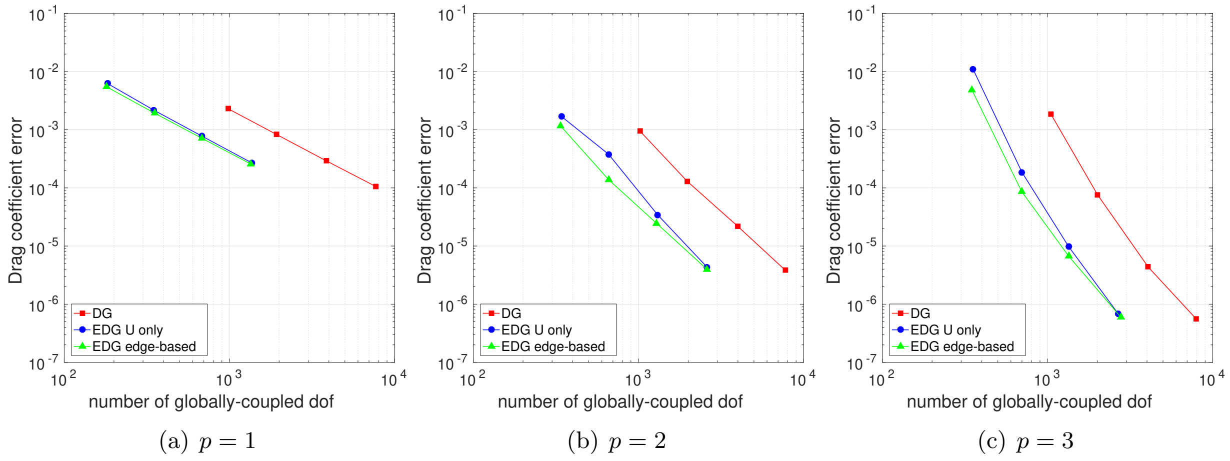
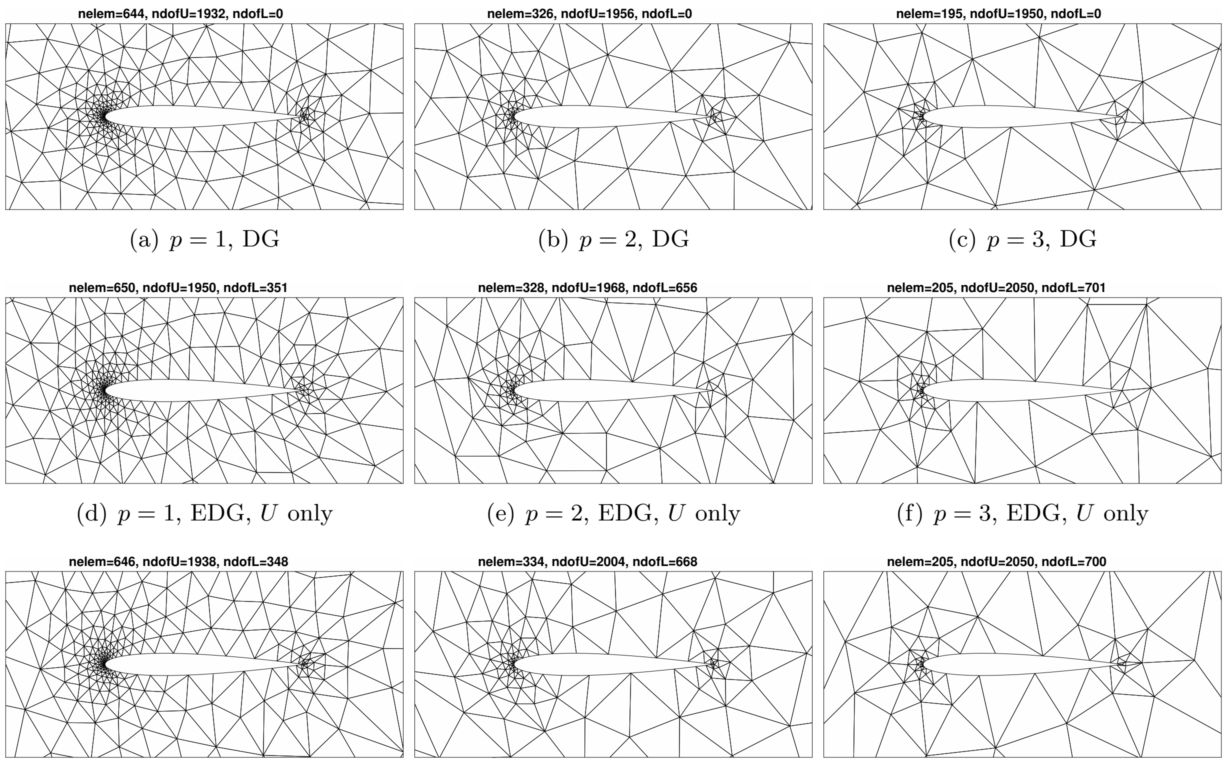


Figure 8. Inviscid flow test: output convergence results versus globally coupled degrees of freedom.

Figure 9 shows the final adapted meshes for the various discretizations and orders run. Areas targeted for refinement consist primarily of the leading and trailing edges in this relatively benign test case. The meshes are fairly similar, with the exception of the trailing edge refinement. The EDG U-only error estimate results show less refinement of the trailing edge compared to DG an EDG with the complete error estimate. This lower degree of refinement is likely responsible for the larger drag errors on these meshes. On the other hand, incorporating the Λ error estimate into the mesh optimization error model leads to more refinement at the trailing edge and lower errors.


nelem=644, ndofU=1932, ndofL=0
nelem=326, ndofU=1956, ndofL=0
nelem=195, ndofU=1950, ndofL=0
nelem=650, ndofU=1950, ndofL=351
nelem=328, ndofU=1968, ndofL=656
nelem=205, ndofU=2050, ndofL=701
nelem=646, ndofU=1938, ndofL=348
nelem=334, ndofU=2004, ndofL=668
nelem=205, ndofU=2050, ndofL=700

(a) $p = 1$, DG (b) $p = 2$, DG (c) $p = 3$, DG

(d) $p = 1$, EDG, U only (e) $p = 2$, EDG, U only (f) $p = 3$, EDG, U only

(g) $p = 1$, EDG, complete (h) $p = 2$, EDG, complete (i) $p = 3$, EDG, complete

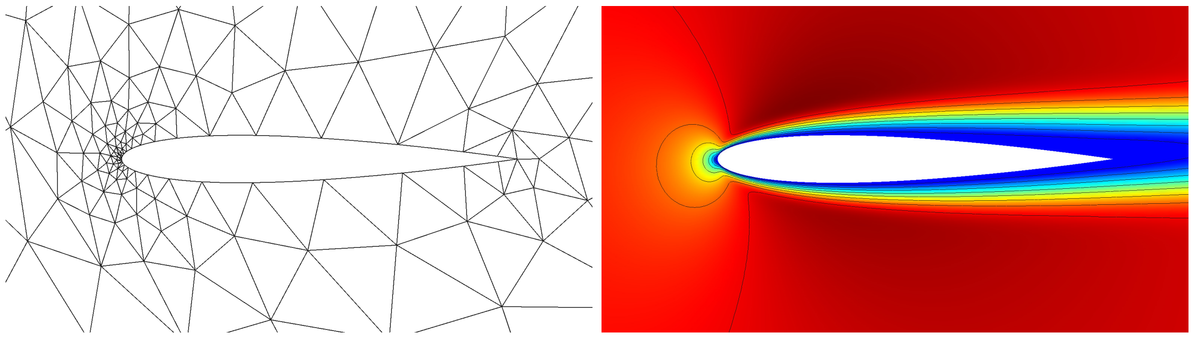
Figure 9. Inviscid flow test: adapted meshes for approximately 2000 interior (U) degrees of freedom.

V.B. Laminar viscous flow

The second test case is viscous flow over a NACA 0012 airfoil at $\alpha = 2°, M = 0.5, Re = 5000$. The output of interest is again the drag coefficient. Figure 10 shows the initial mesh and Mach number contours. The mesh is curved to the geometry using a cubic mapping from reference space.

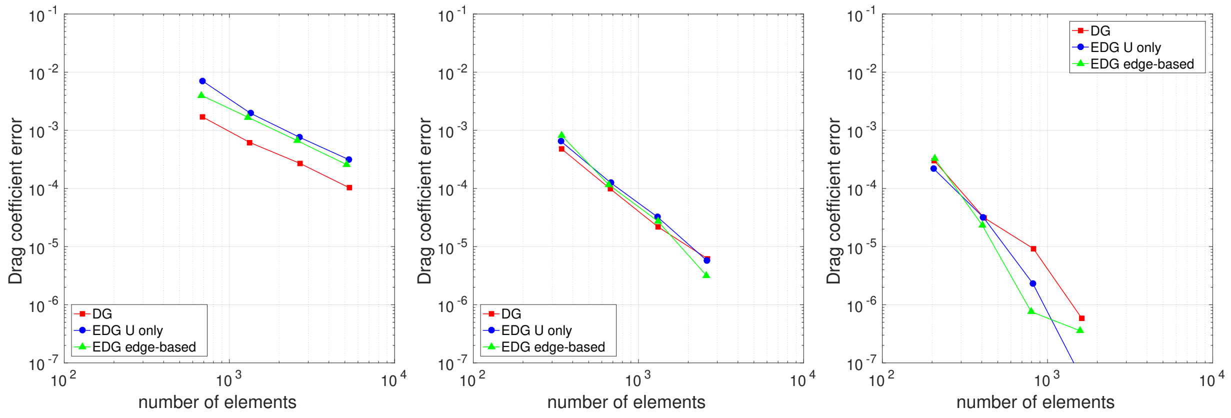
Figure 11 shows the convergence of the output with number of elements for the two discretizations, and again two sets of EDG results are shown: one driven by the element-only ($\mathcal{E}_e^U$ and $\mathcal{E}_e^Q$) error indicator only, and one driven by the complete (including $\mathcal{E}_e^\Lambda$) error indicator. As in the previous case, mesh optimizations in each case were run for 20 adaptive iterations, starting with the initial mesh, for each target number of elements. The data points shown are the average errors and costs over the last six optimization iterations.

As in the inviscid case, for $p = 1$, DG is more accurate than EDG for a given number of elements. The output error for a given number of elements is approximately three times larger for EDG than DG. In addition, edge-based EDG again performs better than U-only EDG. This benefit diminishes at higher orders, but in general, incorporating the Λ and error estimates reduces the output errors in the optimized meshes. For $p > 1$, the EDG results are on par with the DG results



(a) Initial mesh, 326 elements (b) Mach number contours

Figure 10. Viscous flow test: initial mesh and solution Mach contours.

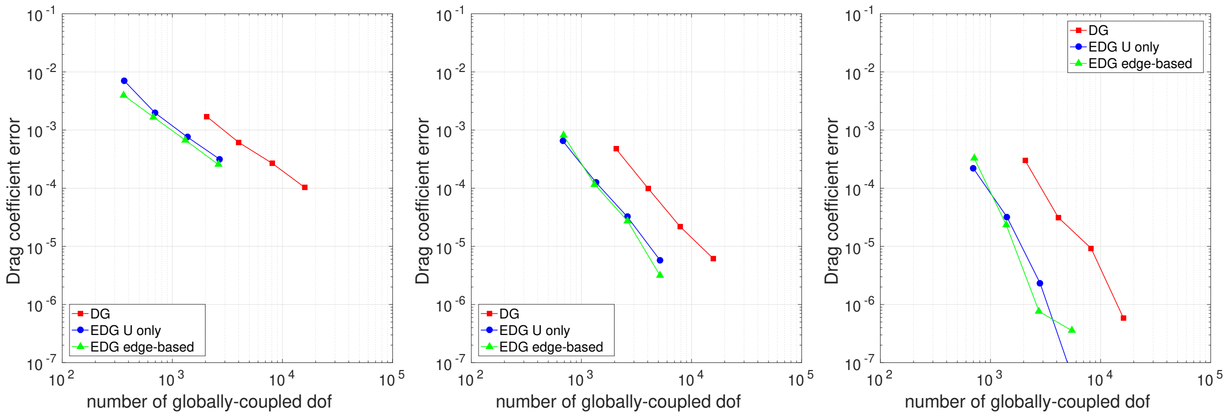


(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Figure 11. Viscous flow test: output convergence results versus number of elements.

for a given number of elements.

Figure 12 presents the same data versus number of globally-coupled degrees of freedom. We see that EDG is again advantageous compared to DG, especially at $p = 2$ and $p = 3$, where the error levels are already comparable for the same number of elements. The benefit of EDG versus DG in terms of globally-coupled of freedom grows with order, as expected since the ratio of element-interior to edge degrees of freedom scales linearly with p. By $p = 3$, on the finer meshes, the degree of freedom advantage is a factor of 4-5 in favor of EDG.



(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Figure 12. Viscous flow test: output convergence results versus number of globally-coupled degrees of freedom.

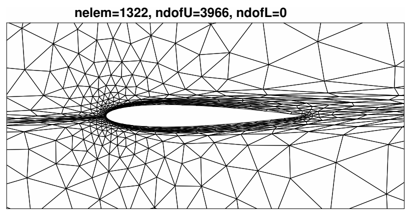
Figure 13 shows the final adapted meshes for the various discretizations and orders run. Areas targeted for refinement consist primarily of the leading edge, boundary layer, and stagnation streamline in front of the airfoil. The meshes are visually similar in this case, though the two sets of EDG results do show differences in the output. This suggests that optimal meshes may not be straightforward to identify purely from a visual analysis.

V.C. RANS flow

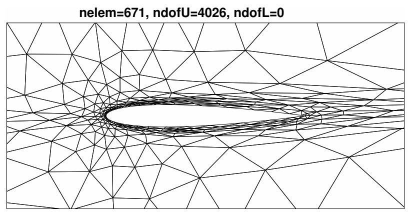
The third test case is Reynolds-averaged turbulent flow over a flat plate at $M = 0.5$, $\alpha = 0°$, and $Re = 10^6$. The flat plate has unit length, and the computational domain extends two units ahead of and behind the flat plate. A symmetry boundary condition is applied on these boundaries, as well as on the top boundary, which is two length units above the plate. Stagnation quantities are prescribed on the left boundary, and static pressure is specified on the right boundary. The RANS-SA equations, with negative turbulent viscosity modification,[32,33] are used for these runs. To aid solver convergence, the RANS equations and working variable are scaled by the square root of the Reynolds number.[34] The output of interest is the drag coefficient on the flat plate. Figure 10 shows the initial mesh.

Figure 15 shows the convergence of the output with number of elements for the two discretizations, and again two sets of EDG results are shown: one driven by the element error indicator only, and one driven by the complete (including $\mathcal{E}_e^{\Lambda}$) error indicator. As in the previous cases, mesh optimizations were run for 20 adaptive iterations, starting with the initial mesh, for each target number of elements. The data points shown are the average errors over the last six optimization iterations.

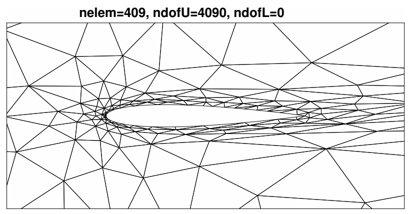
For this case, the $p = 1$ results show a substantial difference in errors between the two EDG adaptive methods. Not incorporating the edge-based error estimates into the adaptive indicator results in meshes for which the EDG drag coefficient error is over an order of magnitude larger than when the edge-based contribution is included. This difference diminishes significantly for $p = 2$ and



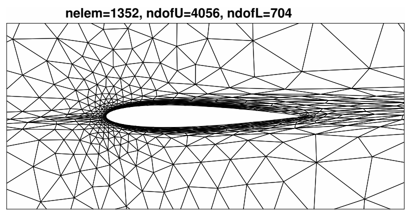
(a) $p = 1$, DG



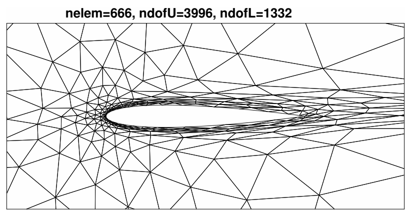
(b) $p = 2$, DG



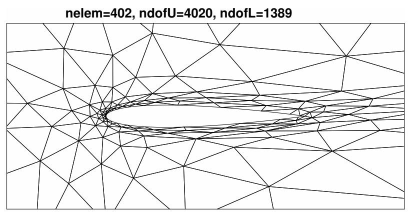
(c) $p = 3$, DG



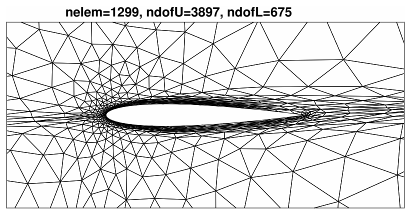
(d) $p = 1$, EDG, U only



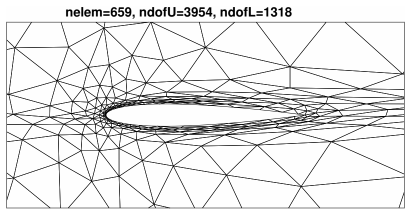
(e) $p = 2$, EDG, U only



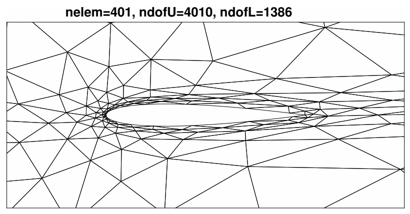
(f) $p = 3$, EDG, U only



(g) $p = 1$, EDG, complete

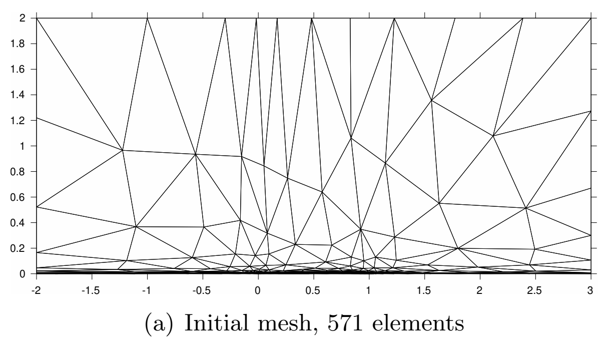


(h) $p = 2$, EDG, complete

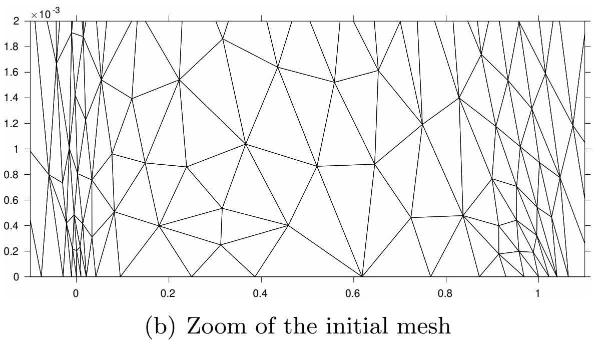


(i) $p = 3$, EDG, complete

Figure 13. Viscous flow test: adapted meshes for approximately 4000 interior (U) degrees of freedom.

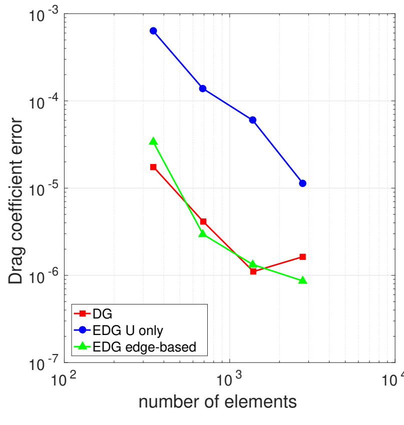


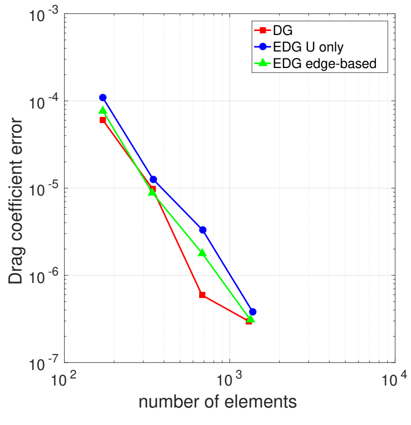
(a) Initial mesh, 571 elements



(b) Zoom of the initial mesh

Figure 14. RANS flow test: initial mesh, with a zoomed-in view of the boundary layer.


(a) $p = 1$


(b) $p = 2$

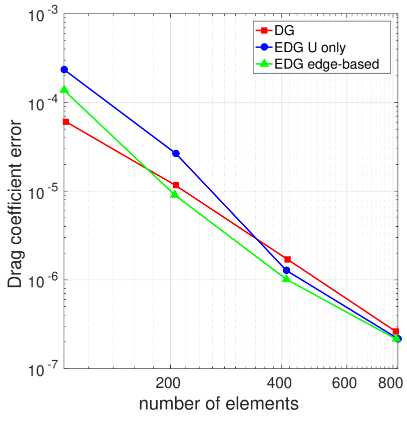
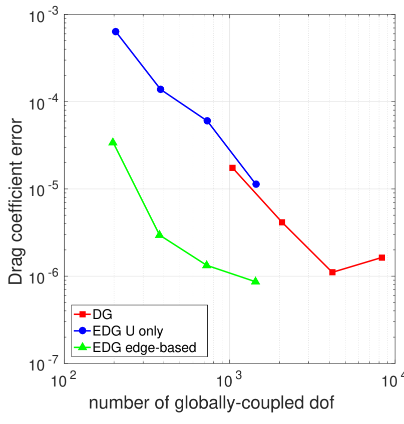
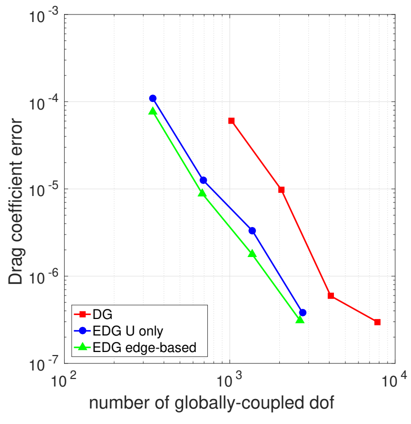

(c) $p = 3$

Figure 15. RANS flow test: output convergence results versus number of elements.

$p = 3$, where the two sets of EDG results are close. In all cases, the results of the EDG edge-based adaptation are close to those of the DG adaptation.

Figure 16 presents the convergence data versus number of globally-coupled degrees of freedom. We see that EDG is again advantageous compared to DG, especially at $p = 2$ and $p = 3$. For $p = 1$, the poorly-performing U-only EDG method is actually still on par with DG, and the edge-based EDG results are approximately a factor of 5 cheaper.


(a) $p = 1$


(b) $p = 2$

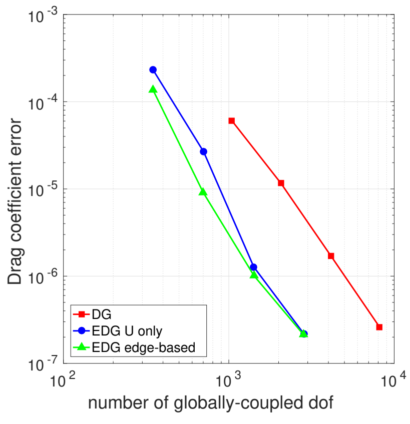
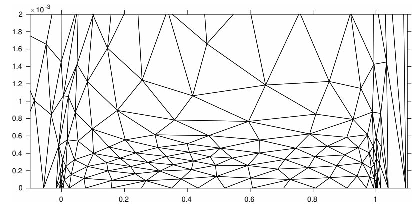
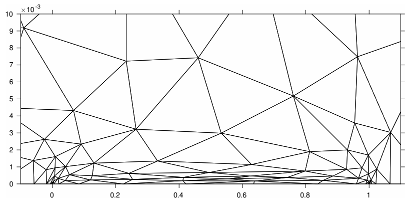
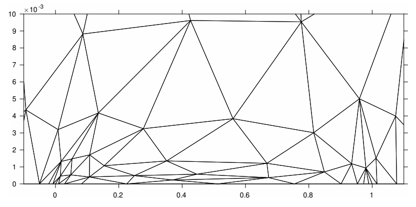

(c) $p = 3$

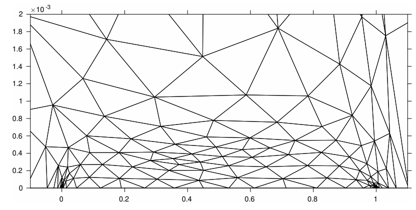
Figure 16. RANS flow test: output convergence results versus number of globally-coupled degrees of freedom.

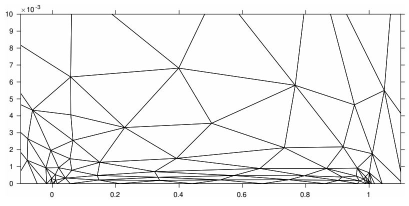
Figure 17 shows the final adapted meshes for the various discretizations and orders run. Small elements are needed to resolve the singularities at the leading and trailing edges of the flat plate. In addition, anisotropic elements are used to efficiently resolve the flow over the flat plate. Note that the figures use vastly different scales in the horizontal and vertical directions, so that the anisotropy is much larger than that apparent from the figures. As expected, for a constant number of element degrees of freedom, the higher-order meshes become coarser. However, the anisotropy at higher orders does not diminish and even increases (e.g. $p = 1$ to $p = 2$). At $p = 1$, a noticeable difference between the U-only EDG mesh and the other two meshes is the resolution near the leading and trailing edges. The U-only EDG mesh does target these areas, but not to the same extent vertically away from the flat plate. This lack of resolution slightly above the leading and trailing edges is likely responsible for the large difference in the drag coefficient errors on the U-only EDG meshes compared to the results on the meshes generated by the other two adaptive indicators.

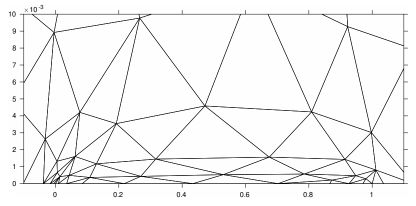

(a) $p = 1$, DG

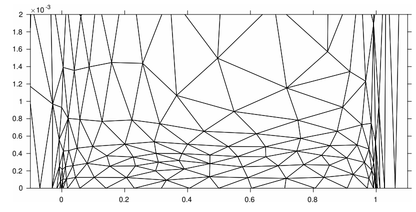

(b) $p = 2$, DG

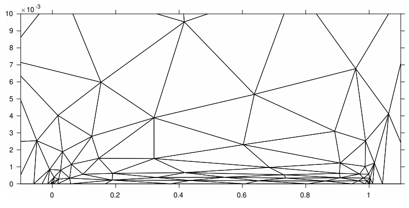

(c) $p = 3$, DG


(d) $p = 1$, EDG, U only


(e) $p = 2$, EDG, U only


(f) $p = 3$, EDG, U only


(g) $p = 1$, EDG, complete


(h) $p = 2$, EDG, complete

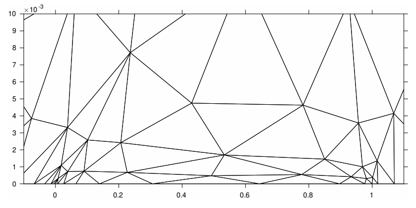

(i) $p = 3$, EDG, complete

Figure 17. RANS flow test: adapted meshes for approximately 1000 interior (U) degrees of freedom.

VI. Conclusions

We present an approach for optimizing meshes using an engineering output as the target and a measure of cost as the constraint. Both discontinuous Galerkin (DG) and embedded discontinuous Galerkin (EDG) discretizations are considered. Whereas mesh optimization has been studied previously for DG, this work extends these ideas to EDG. A particular contribution of this work is the modification of the element-based error model, required for mesh optimization, to take into account error contributions from faces of elements. By including all contributions into the error model and sampling their decay with refinement separately, we obtain a more accurate prediction of the behavior of the error with mesh refinement. This in turns creates more efficient meshes, as shown in the results for the inviscid and viscous cases considered.

Acknowledgments

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

References

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

[2]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.

[3]Nguyen, N., Peraire, J., and Cockburn, B., “An Implicit High-Order Hybridizable Discontinuous Galerkin, Method for Linear Convection-Diffusion Equations,” *Journal of Computational Physics*, Vol. 228, 2009, pp. 3232–3254.

[4]Cockburn, B., Gopalakrishnan, J., and Lazarov, R., “Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems,” *SIAM Journal on Numerical Analysis*, Vol. 47, No. 2, 2009, pp. 1319–1365.

[5]Nguyen, N. C., Peraire, J., and Cockburn, B., “Hybridizable Discontinuous Galerkin Methods,” *Spectral and High Order Methods for Partial Differential Equations*, edited by J. S. Hesthaven, E. M. Rnquist, B. T. J., M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick, Vol. 76 of *Lecture Notes in Computational Science and Engineering*, Springer Berlin Heidelberg, 2011, pp. 63–84, 10.1007/978-3-642-15337-2_4.

[6]Peraire, J., Nguyen, N. C., and Cockburn, B., “An Embedded Discontinuous Galerkin Method for the Compressible Euler and Navier-Stokes Equations,” AIAA Paper 2011-3228, 2011.

[7]Rhebergen, S. and Cockburn, B., “Space-Time Hybridizable Discontinuous Galerkin Method for the AdvectionDiffusion Equation on Moving and Deforming Meshes,” *The CourantFriedrichsLewy (CFL) Condition*, edited by C. A. de Moura and C. S. Kubrusly, Birkhäuser Boston, 2013, pp. 45–63.

[8]Fernandez, P., Nguyen, N., and Peraire, J., “The hybridized Discontinuous Galerkin method for Implicit Large-Eddy Simulation of transitional turbulent flows,” *Journal of Computational Physics*, Vol. 336, 2017, pp. 308–329.

[9]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.

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

[11]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.

[12]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.

[13]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.

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

[15]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.

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

[17]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.

[18]Pierce, N. A. and Giles, M. B., “Adjoint recovery of superconvergent functionals from PDE approximations,” *SIAM Review*, Vol. 42, No. 2, 2000, pp. 247–264.

[19]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.

[20]Hartmann, R. and Houston, P., “Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations,” *Journal of Computational Physics*, Vol. 183, No. 2, 2002, pp. 508–532.

[21]Nemec, M. and Aftosmis, M. J., “Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes,” AIAA Paper 2007-4187, 2007.

[22]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.

[23]Fidkowski, K., “High-Order Output-Based Adaptive Methods for Steady and Unsteady Aerodynamics,” *37th 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.

[24]Dahm, J. P. and Fidkowski, K. J., “Error Estimation and Adaptation in Hybridized Discontinous Galerkin Methods,” AIAA Paper 2014–0078, 2014.

[25]Woopen, M., Balan, A., May, G., and Schütz, J., “A comparison of hybridized and standard DG methods for target-based hp-adaptive simulation of compressible flow,” *Computers & Fluids*, Vol. 98, 2014, pp. 3–16.

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

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

[28]Fidkowski, K. J., “A Hybridized Discontinuous Galerkin Method on Mapped Deforming Domains,” *Computers and Fluids*, Vol. 139, No. 5, November 2016, pp. 80–91.

[29]Fidkowski, K. J., “A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization,” AIAA Paper 2016–0835, 2016.

[30]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.

[31]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.

[32]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.

[33]Ceze, M. A. and Fidkowski, K. J., “Drag Prediction Using Adaptive Discontinuous Finite Elements,” *AIAA Journal of Aircraft*, Vol. 51, No. 4, 2014, pp. 1284–1294.

[34]Ceze, M. A. and Fidkowski, K. J., “Constrained pseudo-transient continuation,” *International Journal for Numerical Methods in Engineering*, Vol. 102, 2015, pp. 1683–1703.