Received: 23 June 2019 | Revised: 6 September 2019 | Accepted: 5 October 2019

DOI: 10.1002/nme.6248

RESEARCH ARTICLE

WILEY

Output-based mesh optimization for hybridized and embedded discontinuous Galerkin methods

Krzysztof J. Fidkowski[1] | **Guodong Chen**[1]

[1]Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan

Correspondence
Krzysztof J. Fidkowski, Department of Aerospace Engineering, University of Michigan, 1320 Beal Avenue, 3029 FXB, Ann Arbor, MI, 48188.
Email: kfid@umich.edu

Funding information
Boeing, 1200596; U.S. Department of Energy, DE-SC0010341

Summary

This paper presents a method for optimizing computational meshes for the prediction of scalar outputs when using hybridized and embedded discontinuous Galerkin (HDG/EDG) discretizations. Hybridization offers memory and computational time advantages compared to the standard discontinuous Galerkin (DG) method through a decoupling of elemental degrees of freedom and the introduction of 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 techniques 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 error sampling uses fine-space adjoint projections and does not require additional solves on subelements. Together with a degree-of-freedom cost model, the error models drive metric-based unstructured mesh optimization. Adaptive results for inviscid and viscous two-dimensional flow problems demonstrate (i) improvement of EDG mesh optimality when using error models that incorporate face errors, (ii) the relative insensitivity of HDG mesh optimality to the incorporation of face errors, and (iii) degree of freedom and computational-time benefits of hybridized methods, particularly EDG, relative to DG.

KEYWORDS

adjoints, embedded discontinuous Galerkin, error estimation, hybridized discontinuous Galerkin, mesh optimization, output-based adaptation

1 | INTRODUCTION

Although discontinuous Galerkin (DG) methods[1-6] 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 (i) modifying the discretization; and (ii) optimizing the computational mesh. In this work we pursue both approaches and compare their relative benefits.

Hybridization of DG[7-11] 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 $p^{\text{dim}-1}$ compared to the p^{dim} scaling for elements, where p is the approximation order and dim is the spatial dimension, HDG methods can be computationally cheaper and use less memory compared to DG. The embedded discontinuous Galerkin (EDG) method[10,12] 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,[13-16] semi-heuristic,[17-20] and more recently, rigorous[21] 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.[5,17,22-28]

This work introduces a mesh optimization approach for hybridized discretizations, effectively combining the two cost-reduction approaches. Novel contributions of this work include the development of two algorithms for incorporating face error contributions from hybridized methods into an element-based mesh optimization procedure, and a comparison of the three discretizations on meshes tailored to each discretization, using multiple cost measures. In addition to reducing computational costs, the resulting adaptive method improves (i) robustness of the solution through quantitative error estimates, and (ii) 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 2 presents the DG, HDG, and EDG discretizations. Section 3 derives adjoint-based error estimates, which drive discretization-specific mesh optimization techniques that are presented in Section 4. Section 5 demonstrates the adaptive method for selected two-dimensional flows, and Section 6 concludes with a summary and a discussion of future directions.

2 | DISCRETIZATION

We simulate the compressible Navier-Stokes equations,

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

where $\mathbf{u}(\vec{x}, t) \in \mathbb{R}^s = [\rho, \rho\vec{v}, \rho E]^T$ 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, $\frac{\partial \mathbf{u}}{\partial t} = \mathbf{0}$, although for clarity in exposition, we leave the unsteady term in the weak forms.

2.1 | Discontinuous Galerkin (DG)

Denote by T_h the set of N_{elem} elements in a nonoverlapping tessellation of the domain Ω. As shown in Figure 1B, 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, $\mathbf{u}_h \in \mathcal{V}_h = [\mathcal{V}_h]^s$, 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 (1) follows from multiplying the equation by test functions in the same approximation space, integrating by parts, and coupling elements via unique fluxes,

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

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 $\hat{\mathbf{u}}_h = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$.

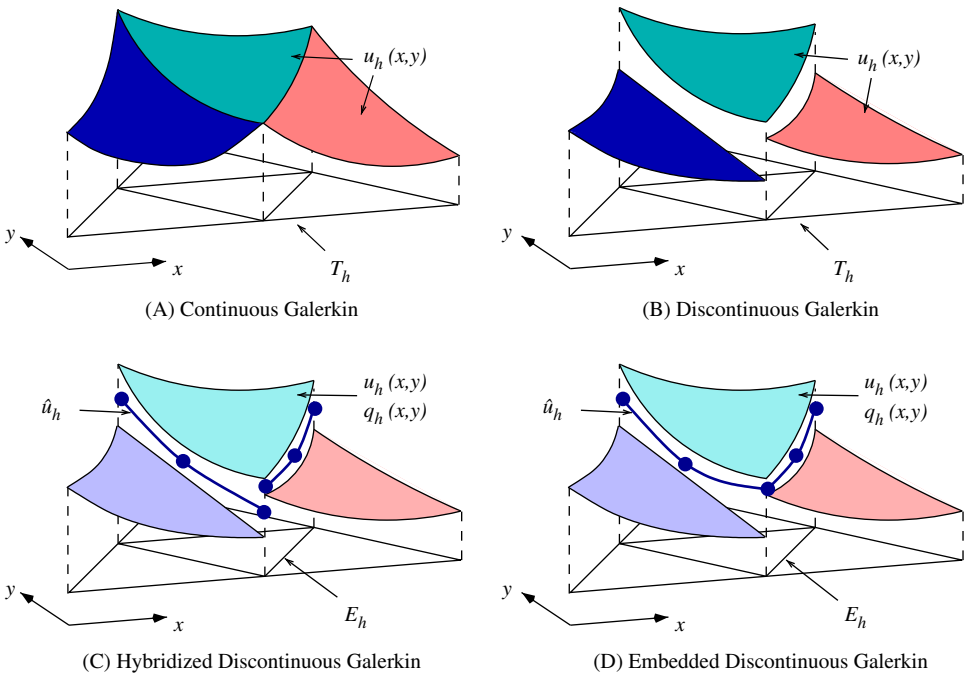


FIGURE 1 Schematic description of the solution approximation using various high-order methods [Color figure can be viewed at wileyonlinelibrary.com]

$\hat{\mathbf{H}} \cdot \vec{n}$ denotes the unique normal flux on faces, computed from the approximated solution and its gradient. We use the Roe approximate Riemann solver[29] for the convective flux, and the second form of Bassi and Rebay (BR2)[30] for the viscous flux. Choosing a basis for the test and trial spaces yields a system of nonlinear equations,

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

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

2.2 | Hybridized and embedded discontinuous Galerkin

The starting point for the HDG discretization is the conversion of (1) to a system of first-order equations,

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

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

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

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

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

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

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

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 $\mathcal{M}_h = \{u \in L_2(\sigma_f) : u|_{\sigma_f} \in \mathcal{P}^p \;\; \forall \sigma_f \in F_h\}$, 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 $\hat{\mathbf{u}}_h$, 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 VS HDG. On the other hand, in HDG, the face approximations are independent and generally discontinuous at nodes and edges in three dimensions. This increases the size of the global system but yields well-defined blocks in the Jacobian matrix that simplify preconditioning.

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

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

Note that τ consists of a convective stabilization computed about the Roe-average state, $\mathbf{u}_h^*$, and a BR2 viscous stabilization,[31] where η is set to the number of faces and $\vec{\boldsymbol{\delta}}_h$ is the BR2 auxiliary variable driven by the state jump $\mathbf{u}_h - \hat{\mathbf{u}}_h$.

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

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

with the Newton update system

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

where $\mathbf{Q}_h$, $\mathbf{U}_h$, and Λ_h are the discrete unknowns in the approximation of $\vec{\mathbf{q}}_h$, $\mathbf{u}_h$, and $\hat{\mathbf{u}}_h$, respectively. $[\mathbf{A}_h, \mathbf{B}_h; \mathbf{C}_h, \mathbf{D}_h]$ is the primal Jacobian matrix partitioned into element-interior and interface unknown blocks. Note that $\mathbf{A}_h$ and $\mathbf{B}_h$ contain both $\mathbf{Q}_h$ and $\mathbf{U}_h$ components. In addition, $\mathbf{A}_h$ 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,

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

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

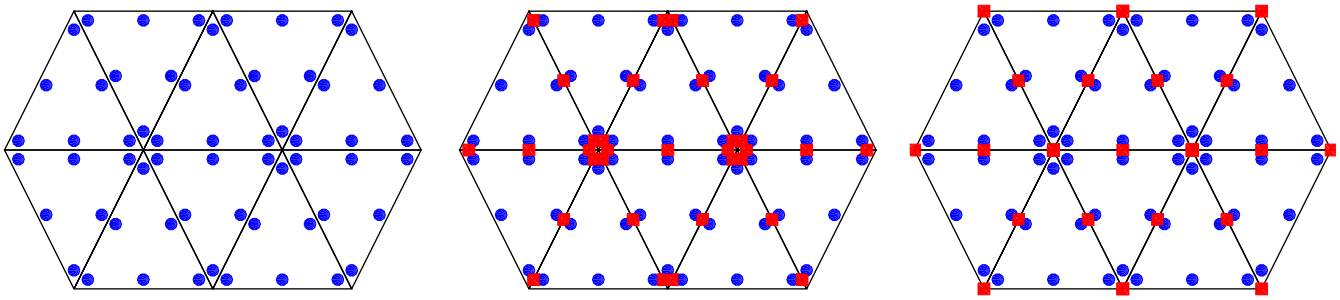
$$\begin{bmatrix} \Delta \mathbf{Q}_h \\ \Delta \mathbf{U}_h \end{bmatrix} = -\mathbf{A}_h^{-1} \left(\begin{bmatrix} \mathbf{R}_h^Q \\ \mathbf{R}_h^U \end{bmatrix} + \mathbf{B}_h \Delta \boldsymbol{\Lambda}_h \right). \tag{13}$$

2.3 | 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. Note that in HDG and EDG, we do not introduce the trace variable, $\hat{\mathbf{u}}_h$, on boundary faces, as the flux there is computed in the same way as in DG.

Table 1 shows average degree of freedom counts of DG, HDG, and EDG, for regular simplex-element meshes. Since different discretizations associate degrees of freedom with different structures of the mesh (elements, faces, edges, nodes), the results in the table normalize the total degrees of freedom by the nodes in the mesh, under assumptions of mesh regularity and ignoring boundaries. As expected, for all orders in two and three dimensions, EDG uses the fewest degrees of freedom out of the three discretizations. The relative benefit of EDG is greatest at the lower approximation orders. Note that HDG on tetrahedra actually consumes more degrees of freedom than DG for lower orders, due to the large number of faces relative to elements.

In addition to degrees of freedom, the number of nonzeros and the sparsity pattern of the Jacobian matrix also affect the computational cost. For the example mesh in Figures 2 and 3 shows the sparsity patterns of the resulting residual Jacobian matrices, with static condensation applied to HDG and EDG. The number of nonzeros refers to the globally

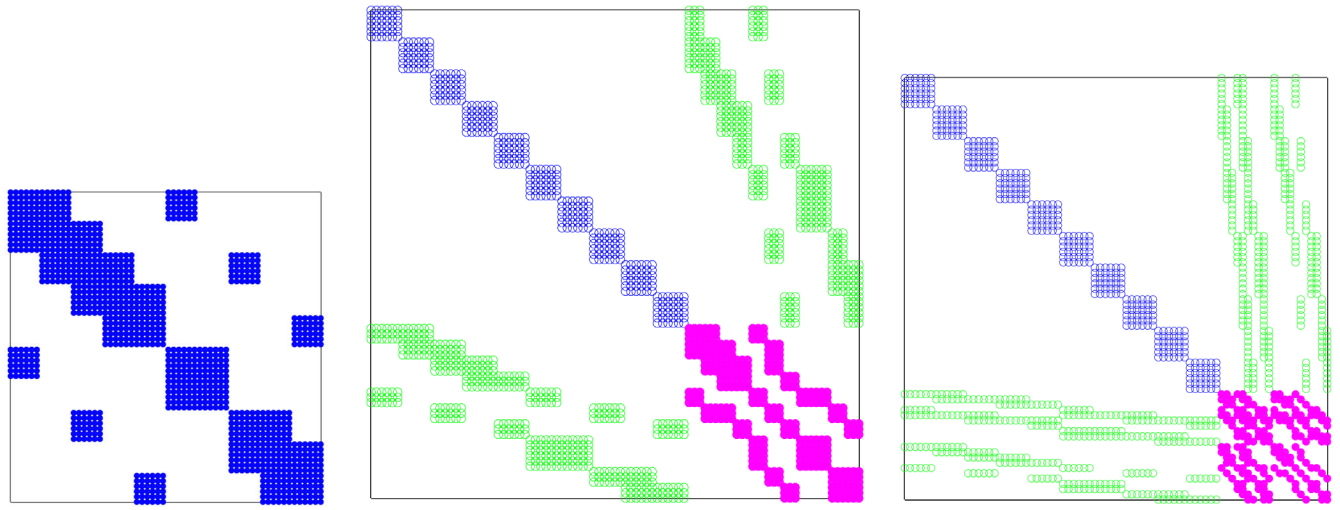


(A) DG (B) HDG (C) EDG

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

TABLE 1 Degree of freedom counts per vertex of regular simplex-element meshes in two and three dimensions

	Triangles				Tetrahedra			
Method	$p=1$	$p=2$	$p=3$	$p=4$	$p=1$	$p=2$	$p=3$	$p=4$
DG	6	12	20	30	24	60	120	210
HDG	6	9	12	15	36	72	120	180
EDG	1	4	7	10	1	8.2	27.4	58.6



(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 [Color figure can be viewed at wileyonlinelibrary.com]

coupled (condensed) matrices. Note that the number of matrix nonzeros for EDG is about a factor of 6 smaller than for DG. However, a caveat is that the arrangement of nonzeros in blocks in DG and HDG can be favorable for preconditioner effectivity.

2.4 | Adjoint discretization

For a scalar output J_h, computed from the discrete solution $\mathbf{U}_h$, the discrete adjoint $\mathbf{\Psi}_h$ is a vector of sensitivities of J_h to residual source perturbations. For DG, these perturbations refer to (3), and the associated adjoint equation is

$$\left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right)^T \mathbf{\Psi}_h + \left(\frac{\partial J_h}{\partial \mathbf{U}_h}\right)^T = \mathbf{0}. \tag{14}$$

For HDG and EDG, residual perturbations refer to (10), and with three sets of residuals, the analog of (14) is

$$\begin{bmatrix} \mathbf{A}_h^T & \mathbf{C}_h^T \\ \mathbf{B}_h^T & \mathbf{D}_h^T \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}_h^Q \\ \mathbf{\Psi}_h^U \\ \mathbf{\Psi}_h^\Lambda \end{bmatrix} + \begin{bmatrix} (\partial J_h/\partial \mathbf{Q}_h)^T \\ (\partial J_h/\partial \mathbf{U}_h)^T \\ (\partial J_h/\partial \mathbf{\Lambda}_h)^T \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{15}$$

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

$$\underbrace{\left(\mathbf{D}_h^T - \mathbf{B}_h^T \mathbf{A}_h^{-T} \mathbf{C}_h^T\right)}_{\boldsymbol{\kappa}_h^T} \mathbf{\Psi}_h^\Lambda + \left(\frac{\partial J_h}{\partial \mathbf{\Lambda}_h}^T - \mathbf{B}_h^T \mathbf{A}_h^{-T} \left[\frac{\partial J_h}{\partial \mathbf{Q}_h}^T ; \frac{\partial J_h}{\partial \mathbf{U}_h}^T \right] \right) = \mathbf{0}. \tag{16}$$

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

3 | OUTPUT ERROR ESTIMATION

3.1 | 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.[23,25] Let H denote a coarse/current discretization space, and h a fine one, for example, 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 HDG/EDG. Computing the fine-space residuals with these injected states and weighting them by the fine-space adjoint gives an estimate of the output error between the coarse and fine spaces,

$$\text{DG:} \qquad J_h\left(\mathbf{U}_h^H\right) - J_h\left(\mathbf{U}_h\right) \approx -\delta\mathbf{\Psi}_h^T \mathbf{R}_h\left(\mathbf{U}_h^H\right) \tag{17}$$

$$\text{HDG/EDG:} \qquad J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx \underbrace{-(\delta\mathbf{\Psi}_h^Q)^T \mathbf{R}_h^Q}_{\delta J^Q} \underbrace{-(\delta\mathbf{\Psi}_h^U)^T \mathbf{R}_h^U}_{\delta J^U} \underbrace{-(\delta\mathbf{\Psi}_h^\Lambda)^T \mathbf{R}_h^\Lambda}_{\delta J^\Lambda}, \tag{18}$$

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 HDG/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 (18) separates the error estimate into three components, one for each residual.

3.2 | Error localization

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

$$\text{DG:} \qquad \mathcal{E}_e \equiv \left| \delta\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H) \right|, \tag{19}$$

$$\text{HDG/EDG:} \qquad \mathcal{E}_e^Q \equiv \left| \delta\mathbf{\Psi}_{h,e}^{QT} \mathbf{R}_{h,e}^Q \right|, \quad \mathcal{E}_e^U \equiv \left| \delta\mathbf{\Psi}_{h,e}^{UT} \mathbf{R}_{h,e}^U \right|. \tag{20}$$

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 $\boldsymbol{\mathcal{M}}_h$.

3.3 | HDG and 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 must convert the HDG and EDG face output error contribution,

δJ^Λ, to contributions associated with elements. In this work, we present two methods for such a conversion. We assume two spatial dimensions, so the notation of "faces" switches to "edges."

3.3.1 | Edge-based averaging

The variational form of the face contribution to the output error is

$$\delta J^\Lambda = -\sum_f \int_{\sigma_f} (\delta \boldsymbol{\psi}_h^\Lambda)^T \left\{ \hat{\mathbf{H}} \cdot \vec{n}|_L + \hat{\mathbf{H}} \cdot \vec{n}|_R \right\} \, ds, \tag{21}$$

where $\boldsymbol{\psi}_h^\Lambda \in \boldsymbol{\mathcal{M}}_h$ is the edge-based adjoint approximation, and the term in curly brackets is the strong-form residual of the flux-continuity Equation (8). Note that this residual is computed on the fine-space, using the injected coarse-space solution. Define $\delta\hat{\mathbf{H}} \cdot \vec{n}|_L \equiv \hat{\mathbf{H}} \cdot \vec{n}|_L - \hat{\mathbf{H}} \cdot \vec{n}|_{\text{exact}}$, where "exact" refers to the flux computed from the exact (no numerical error) solution, and similarly for $\delta\hat{\mathbf{H}} \cdot \vec{n}|_R$. The edge output error estimate can then be written as

$$\begin{aligned}
\delta J^\Lambda &= -\sum_f \int_{\sigma_f} (\delta \boldsymbol{\psi}_h^\Lambda)^T \{ \hat{\mathbf{H}} \cdot \vec{n}|_L + \hat{\mathbf{H}} \cdot \vec{n}|_R \} \, ds \\
&= -\sum_f \int_{\sigma_f} (\delta \boldsymbol{\psi}_h^\Lambda)^T \{ \delta\hat{\mathbf{H}} \cdot \vec{n}|_L + \delta\hat{\mathbf{H}} \cdot \vec{n}|_R \} \, ds \\
&= -\sum_e \int_{\partial\Omega_e} (\delta \boldsymbol{\psi}_h^\Lambda)^T \delta\hat{\mathbf{H}} \cdot \vec{n} \, ds \\
&= -\sum_e \underbrace{\int_{\Omega_e} \nabla \cdot [(\delta \boldsymbol{\psi}_h^\Lambda)^T \delta\vec{\mathbf{H}}] \, d\Omega}_{\mathcal{E}_e^\Lambda}
\end{aligned} \tag{22}$$

The quantity $\mathcal{E}_e^\Lambda$ is an element-based contribution, calculated in practice by equally distributing edge contributions to elements, as shown in Figure 4. In HDG, this means that each element picks up half of the output error from the adjacent edges. In EDG, the discrete adjoint-weighted residuals from the Λ error estimate are not tied to single edges, but are instead associated with nodes or edges. The weight for the division is therefore the inverse of the number of elements adjacent to the structure. Finally, the motivation for writing δJ^Λ as in (22), using the divergence theorem, is to allow the formulation of an element-based error model during element subdivision sampling for mesh optimization.

3.3.2 | Residual lumping

An alternative method for converting edge errors to elements is based on rewriting the edge adjoint in terms of the element adjoint and lumping residuals from edges to elements via the reverse of static condensation. This method relies on the assumption that the output J does not depend on the edge degrees of freedom, $\boldsymbol{\Lambda}$, which is the case for any practical

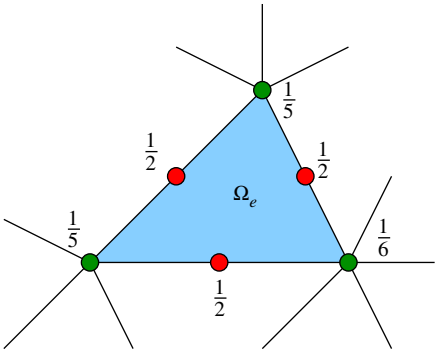


FIGURE 4 Weights for distributing residuals, and adjoint-weighted residuals from globally coupled EDG degrees of freedom to elements. Edge weights are $\frac{1}{2}$ and node weights are the inverse of the node cardinality [Color figure can be viewed at wileyonlinelibrary.com]

domain or boundary-integral output, as $\mathbf{\Lambda}$ is not used on boundary edges. Note that in this section we drop the h subscript from vectors and matrices to simplify the notation. In this case, the adjoint system in (15) becomes

$$\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}^U \\ \mathbf{\Psi}^\Lambda \end{bmatrix} + \begin{bmatrix} \frac{\partial J}{\partial \mathbf{U}}^T \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{23}$$

where we have lumped the $\mathbf{Q}$ unknowns with the $\mathbf{U}$ unknowns to simplify the notation. Taking the transpose and solving for the adjoint,

$$\begin{bmatrix} \mathbf{\Psi}^U \\ \mathbf{\Psi}^\Lambda \end{bmatrix}^T = - \begin{bmatrix} \frac{\partial J}{\partial \mathbf{U}} \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1}. \tag{24}$$

Using the matrix block-inverse formula with $\mathbf{L} \equiv \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}$

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{L}^{-1} & -\mathbf{L}^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{L}^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{L}^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix}, \tag{25}$$

the adjoint vectors are

$$(\mathbf{\Psi}^U)^T = -\frac{\partial J}{\partial \mathbf{U}}\mathbf{L}^{-1}, \tag{26}$$

$$(\mathbf{\Psi}^\Lambda)^T = \frac{\partial J}{\partial \mathbf{U}}\mathbf{L}^{-1}\mathbf{B}\mathbf{D}^{-1} = -(\mathbf{\Psi}^U)^T\mathbf{B}\mathbf{D}^{-1}. \tag{27}$$

The total adjoint-weighted residual error estimate is

$$\delta J = \delta J^U + \delta J^\Lambda = (\mathbf{\Psi}^U)^T\mathbf{R}^U + (\mathbf{\Psi}^\Lambda)^T\mathbf{R}^\Lambda = (\mathbf{\Psi}^U)^T\mathbf{R}^U - (\mathbf{\Psi}^U)^T\mathbf{B}\mathbf{D}^{-1}\mathbf{R}^\Lambda = (\mathbf{\Psi}^U)^T\underbrace{(\mathbf{R}^U - \mathbf{B}\mathbf{D}^{-1}\mathbf{R}^\Lambda)}_{\text{lumped residual}}. \tag{28}$$

The conversion of the edge residuals to elements requires the application of $\mathbf{D}^{-1}$. Fortunately, $\mathbf{D}$ has the same sparsity pattern as the global Jacobian $\boldsymbol{\mathcal{K}}$, and so the same data structures and solver can be used. Finally, we note that this method also assumes that $\mathbf{D}$ is invertible, which has been observed to be true for all cases tested.

4 | 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, HDG, and EDG requires optimal meshes for each discretization. In previous work, we presented an output-based mesh optimization algorithm for DG,[32] which built on earlier work of Yano.[21] This section describes an extension of this algorithm, Mesh Optimization through Error Sampling and Synthesis (MOESS), to HDG and EDG.

4.1 | Mesh Metrics

A Riemannian metric field, $\mathcal{M}(\vec{x}) \in \mathbb{R}^{\dim \times \dim}$, 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)[33] supports metric-based remeshing 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.[34] Changes to the metric are introduced using a symmetric step matrix, $\mathcal{S} \in \mathbb{R}^{\dim \times \dim}$, according to

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S}) \mathcal{M}_0^{\frac{1}{2}}. \tag{29}$$

4.2 | 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[21]

$$\text{DG:} \quad \mathcal{E}_e = \mathcal{E}_{e0} \exp\left[\text{tr}(\mathcal{R}_e \mathcal{S}_e)\right], \tag{30}$$

where $\mathcal{R}_e$ is an element-specific error rate tensor determined through a sampling procedure, as described in Section 4.5. The total error over the mesh is the sum of the elemental errors, $\mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e$.

For HDG and 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 $\mathcal{S}_e$,

$$\text{HDG and EDG:} \quad \mathcal{E}_e^U = \mathcal{E}_{e0}^U e^{\text{tr}(R_e^U S_e)}, \quad \mathcal{E}_e^Q = \mathcal{E}_{e0}^Q e^{\text{tr}(R_e^Q S_e)}, \quad \mathcal{E}_e^\Lambda = \mathcal{E}_{e0}^\Lambda e^{\text{tr}(R_e^\Lambda S_e)}, \tag{31}$$

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 $\mathcal{E}_e = \mathcal{E}_e^U + \mathcal{E}_e^Q + \mathcal{E}_e^\Lambda$.

4.3 | 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 HDG and EDG. Elemental degrees of freedom can also be included in the case of HDG and 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 $\exp\left[\frac{1}{2}\text{tr}(S_e)\right]$. As the number of new elements occupying the original area Ω_e increases by this factor, the elemental cost model is

$$C_e = C_0 \exp\left[\frac{1}{2}\text{tr}(\mathcal{S}_e)\right]. \tag{32}$$

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

4.4 | 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 ($\tilde{\mathcal{S}}_v$) parts, $\mathcal{S}_v = s_v \mathcal{I} + \tilde{\mathcal{S}}_v$.

The optimization algorithm then consists of the following steps:[21,32]

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: $i = 1 \ldots n_{\text{step}}$
   (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 $\tilde{\mathcal{S}}_v$.
   (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: $\mathcal{S}_v = \mathcal{S}_v + \delta s (\partial \mathcal{E}/\partial \tilde{\mathcal{S}}_v)/(\partial \mathcal{E}/\partial s_v)$.

(e) Rescale $S_v \rightarrow S_v + \beta \mathcal{I}$, where β is a global constant calculated from (32) to constrain the total cost to the desired `dof` value: $\beta = \frac{2}{d} \log \frac{C_{\text{target}}}{C}$, where C_{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, C_{target}, until the error stops changing.

4.5 | Error sampling

The error convergence models in Section 4.2 rely on convergence tensors, for example, $\mathcal{R}_e$ 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 5, and performing a regression. However, as described next and first introduced in our previous work,[32] we never actually modify the mesh when considering the refinement samples, which greatly simplifies the algorithm.

Each element refinement is also associated with the refinement of a certain number of adjacent edges. For HDG and EDG with the edge-based averaging localization, these edge refinements reduce the error contributions of those edges to the element error indicator $\mathcal{E}_e^{\Lambda}$, per (22). Therefore, the same refinements shown in Figure 5 can be used to sample all three error contributions in this localization. To determine how much the error(s) decrease(s) for each refinement option, we use element and edge-local projections of the fine-space adjoints to semi-refined spaces associated with each element/edge refinement option.

4.5.1 | 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 5: 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,

$$\mathcal{E}_{ei} \equiv \mathcal{E}_{e0} - \Delta\mathcal{E}_{ei}. \tag{33}$$

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

$$\Delta\mathcal{E}_{ei} \equiv \mathbf{\Psi}_{h,e}^{Hi\ T} \mathcal{R}_{h,e}(\mathbf{U}_h^H), \tag{34}$$

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 (34), with all calculations occurring at order $p+1$ on the original element. Using L_2 projections in reference space, the combination of projections can be encapsulated into one transfer matrix that converts $\mathbf{\Psi}_h$ into $\mathbf{\Psi}_h^{Hi}$, both represented in the order $p+1$ space on the original element:

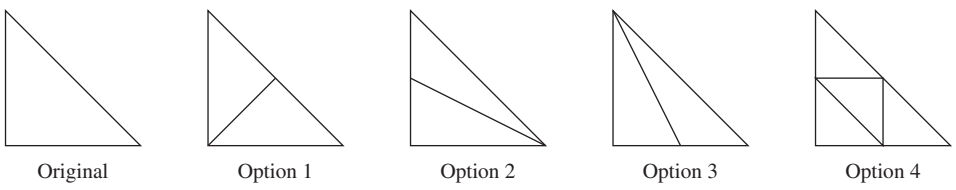


FIGURE 5 Four refinement options for a triangle, with edge refinements highlighted. Each one is considered implicitly during error sampling, though the elements are never actually refined

$$\boldsymbol{\Psi}_h^{Hi} = \mathbf{T}_i \boldsymbol{\Psi}_h, \tag{35}$$

$$\mathbf{T}_i = [\mathbf{M}_0(\phi_0^{p+1}, \phi_0^{p+1})]^{-1} \sum_{k=1}^{n_i} \mathbf{T}_{ik}, \tag{36}$$

$$\mathbf{T}_{ik} = \mathbf{M}_k(\phi_0^{p+1}, \phi_k^p)[\mathbf{M}_k(\phi_k^p, \phi_k^p)]^{-1}\mathbf{M}_k(\phi_k^p, \phi_k^{p+1})[\mathbf{M}_k(\phi_k^{p+1}, \phi_k^{p+1})]^{-1}\mathbf{M}_k(\phi_k^{p+1}, \phi_0^{p+1}). \tag{37}$$

In these equations, n_i is the number of subelements in refinement option i, k is an index over these subelements, ϕ_k^p, ϕ_k^{p+1} are order p and $p+1$ basis functions on subelement 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

$$\mathbf{M}_k(\phi_l, \phi_m) = \int_{\Omega_k} \phi_l \phi_m d\Omega, \quad \mathbf{M}_0(\phi_l, \phi_m) = \int_{\Omega_0} \phi_l \phi_m d\Omega, \tag{38}$$

where Ω_k is subelement 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.

4.5.2 | Hybrid and embedded discontinuous Galerkin

In HDG and 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. In addition, when using the residual lumping localization approach presented in Section 3.3.2, no additional steps are needed, as the edge residuals are already included with the element residuals. On the other hand, when using the edge-based averaging approach to compute $\mathcal{E}_e^\Lambda$, the elemental error indicator is the sum of adjoint-weighted flux residuals integrated over the edges of Ω_e, as given in (22). Let $\mathcal{E}_f$ be the error indicator associated with one edge σ_f, which from (21) is

$$\mathcal{E}_f \equiv \left| \int_{\sigma_f} \boldsymbol{\psi}_h^{\Lambda T} \{\hat{\mathbf{H}} \cdot \vec{n}|_L + \hat{\mathbf{H}} \cdot \vec{n}|_R\} \, ds \right|.$$

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

Performing 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 edges are reduced to $\mathcal{E}_{fj}$ using the results of the edge sampling procedure. The residual distribution is then performed again, yielding $\mathcal{E}_{ei}^\Lambda$.

4.5.3 | Regression

After calculating the element error indicators, for example, $\mathcal{E}_{ei}$ for DG, and the errors remaining after each refinement option i according to (33), we use least-squares regression to estimate the corresponding rate tensors, for example, $\mathcal{R}_e$ for DG. Note that for triangles, we have four refinement options and three independent entries in the symmetric $\mathcal{R}_e$ tensor. Using (30), we formulate the regression to minimize the following error, summed over refinement options,

$$\sum_i \left[\log \frac{\mathcal{E}_{ei}}{\mathcal{E}_{e0}} - \mathrm{tr}(\mathcal{R}_e \mathcal{S}_{ei}) \right]^2. \tag{39}$$

In this equation, $\mathcal{S}_{ei}$ is the step matrix associated with refinement option i, given by (from Equation (29)),

$$\mathcal{S}_{ei} = \log\left(\mathcal{M}_0^{-\frac{1}{2}} \mathcal{M}_i \mathcal{M}_0^{-\frac{1}{2}} \right), \tag{40}$$

where $\mathcal{M}_i$ is the affine-invariant metric average of the mesh-implied metrics of all subelements in refinement option i. Differentiating (39) with respect to the independent components of $\mathcal{R}_e$ yields a linear system for these components. For HDG and EDG, this regression is performed separately for the two or three element-based errors appearing in Equation (31).

5 | RESULTS

This section presents results that compare the performance of the DG, HDG, and EDG discretizations in an output-based mesh-optimization framework. All three discretizations are implemented in a unified code and share the same structures, functions, and algorithms. The implementations differ only in the high-level residual and Jacobian matrix construction procedures.

The discrete global systems are solved using GMRES, preconditioned by a block-Jacobi smoother. The size of the blocks is specific to the discretization. For DG, the blocks are element-based, and for HDG, the blocks are edge-based. In EDG, there is no clear grouping of the globally coupled unknowns into blocks, and so instead a "point"-Jacobi method is used, where a "point" actually corresponds to a $s \times s$ block of unknowns associated with one degree of freedom. The smaller block size makes the EDG preconditioner less powerful for a given Jacobian matrix size, in that more iterations are required to achieve a desired error tolerance. This number of iterations grows with the mesh size, such that large problems will warrant more powerful preconditioners.

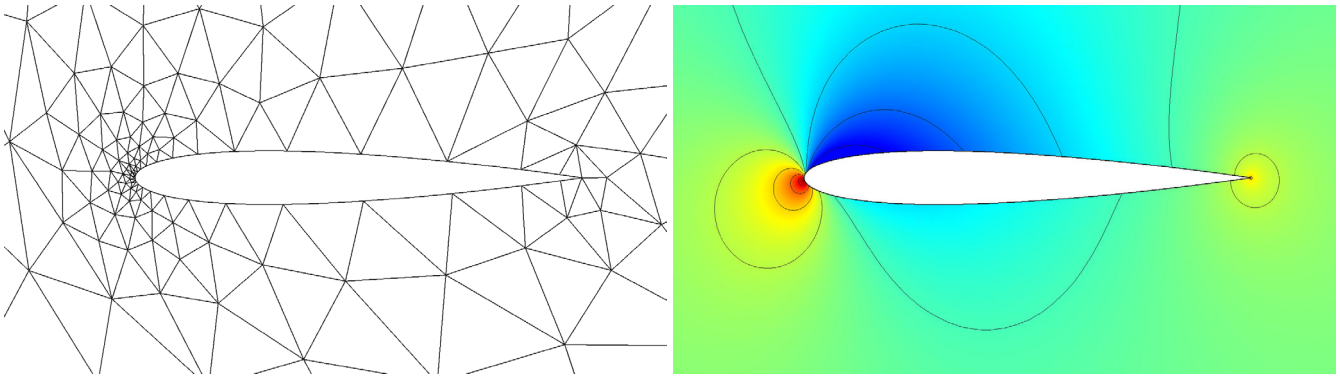
MOESS adaptations are performed at various orders using a target number of element-interior (U) degrees of freedom. Computational time results are obtained using serial runs on a workstation with Intel Xeon Sandy Bridge 2.0 GHz processors with 15MB L3 Cache and 64GB total RAM.

5.1 | Inviscid 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, which is expected to be zero but in fact converges to a nonzero value in the limit of infinite resolution due to the imposition of farfield boundary conditions at a finite distance (100 chord lengths) from the airfoil. As such, the exact value of the drag is computed on a very fine adapted mesh of 45 000 elements and approximation order $p = 4$. Figure 6 shows the initial mesh and Mach number contours. The mesh is curved to the geometry using a cubic mapping from reference space.

Before considering performance of the adaptive algorithm, we present results from a uniform refinement study. All three discretizations, DG, HDG, and EDG, are run using approximation orders $p = 1, 2, 3$, on the same sequence of meshes obtained by uniformly refining the initial mesh. Figure 7 shows the convergence of the error in the drag coefficient using four measures of cost: the number of elements, the number of globally coupled degrees of freedom, the number of nonzeros in the residual Jacobian matrix, and the computational time.

From the plot VS the number of elements, we see that for the same mesh, DG and HDG yield nearly identical results, whereas the EDG result has more error, particularly at $p = 1$. This is expected due to the continuous approximation of



(A) Initial mesh, 326 elements (B) Mach number contours

FIGURE 6 Inviscid flow: initial mesh and solution Mach contours [Color figure can be viewed at wileyonlinelibrary.com]

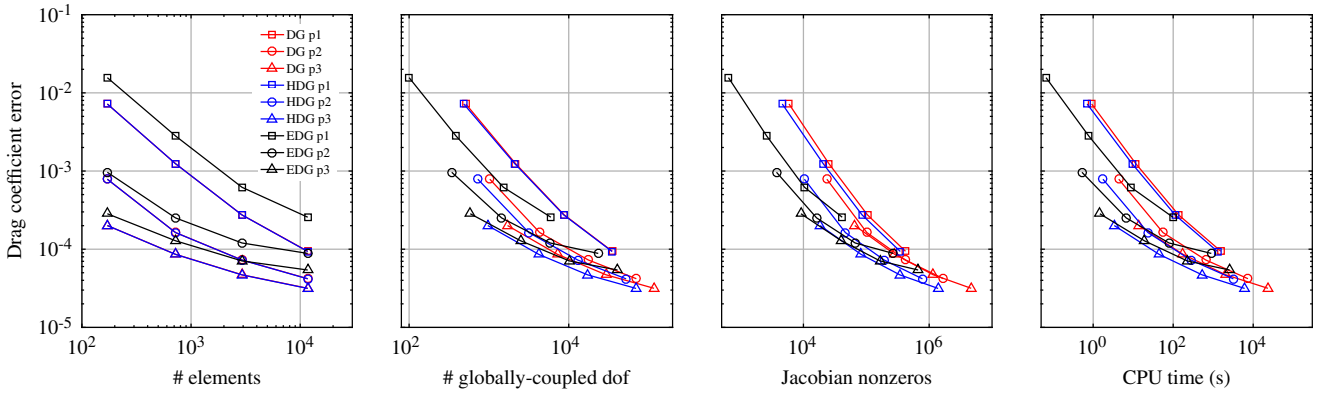


FIGURE 7 Inviscid flow: uniform-refinement output convergence results

the trace state in EDG, which is associated with generally larger jumps between the element and trace solutions and results in a more dissipative scheme. The comparison VS globally coupled degrees of freedom is a left-shift of the EDG and HDG data relative to DG. The large shift of $p = 1$ EDG makes it advantageous over DG and HDG, both of which have the same number of degrees of freedom at $p = 1$ per Table 1. For $p = 2$ and $p = 3$, whereas EDG shows a slight advantage on coarser meshes, by the finer meshes, HDG and DG are more efficient. A similar trend is observed VS the number of Jacobian nonzeros, where EDG shows benefits down to slightly finer meshes, but then is overtaken by HDG and DG. Finally, the CPU timing results are similar, with EDG performing well at $p = 1$, and HDG becoming most efficient at the higher-orders. Recall that the block preconditioner for DG and HDG is more powerful (larger blocks) compared to EDG. We next show how these conclusions change when adaptive meshes are used instead of uniform refinement.

Figure 8 shows the adaptive convergence of the drag coefficient output with the same four cost measures as in the uniform refinement study. Results for $p = 1$ and $p = 3$ are shown separately, as three adaptive methods are now considered for EDG and HDG. These methods differ in the error localization approach: "U only" indicates that only the element-based errors are used to drive the adaptation, and that edge errors are ignored; "edge-based" denotes the averaging of edge/node errors as described in Section 3.3.1; and "res lump" denotes the residual-lumping approach described in Section 3.3.2. The mesh optimizations in each case were run for 20 adaptive iterations, starting with the initial mesh, for each target number of element-based degrees of freedom. The data points shown are the average errors and costs over the last six optimization iterations.

For a given number of elements, DG and HDG exhibit nearly identical errors at both orders, which is similar to the results of the uniform-refinement study. At $p = 1$, EDG fares consistently worse for a given number of elements. In addition, the difference between the three adaptive approaches is small: all EDG errors are approximately three times larger than those of DG and HDG. At $p = 3$, the EDG results including edge error estimates close the gap relative to DG and HDG in terms of number of elements. However, the U-only EDG adaptive indicator lags behind, particularly on the coarser meshes, where it produces larger errors compared to the other adaptive approaches.

Looking next at the comparison VS the globally coupled degrees of freedom, we see that the left-shift of the EDG errors has a significant impact on the results. For a given error level, the EDG simulations consume about a third of the degrees of freedom of DG, for both $p = 1$ and $p = 3$. HDG coincides with DG at $p = 1$, but then exhibits a reduction in the degrees of freedom by $p = 3$ and ends up between EDG and DG. The relationship between the data remains similar in the comparison VS Jacobian nonzeros, with HDG more closely approaching EDG. Nevertheless, in all cases, EDG demonstrates a clear advantage over DG and HDG in the relevant cost measures, with the largest benefits at the lower orders.

Figure 9 shows the final (20th iteration, 200 target elements) adapted meshes for the various discretizations at $p = 3$. 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 mesh shows less refinement of the trailing edge compared to the other methods, including EDG with an edge error treatment. 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.

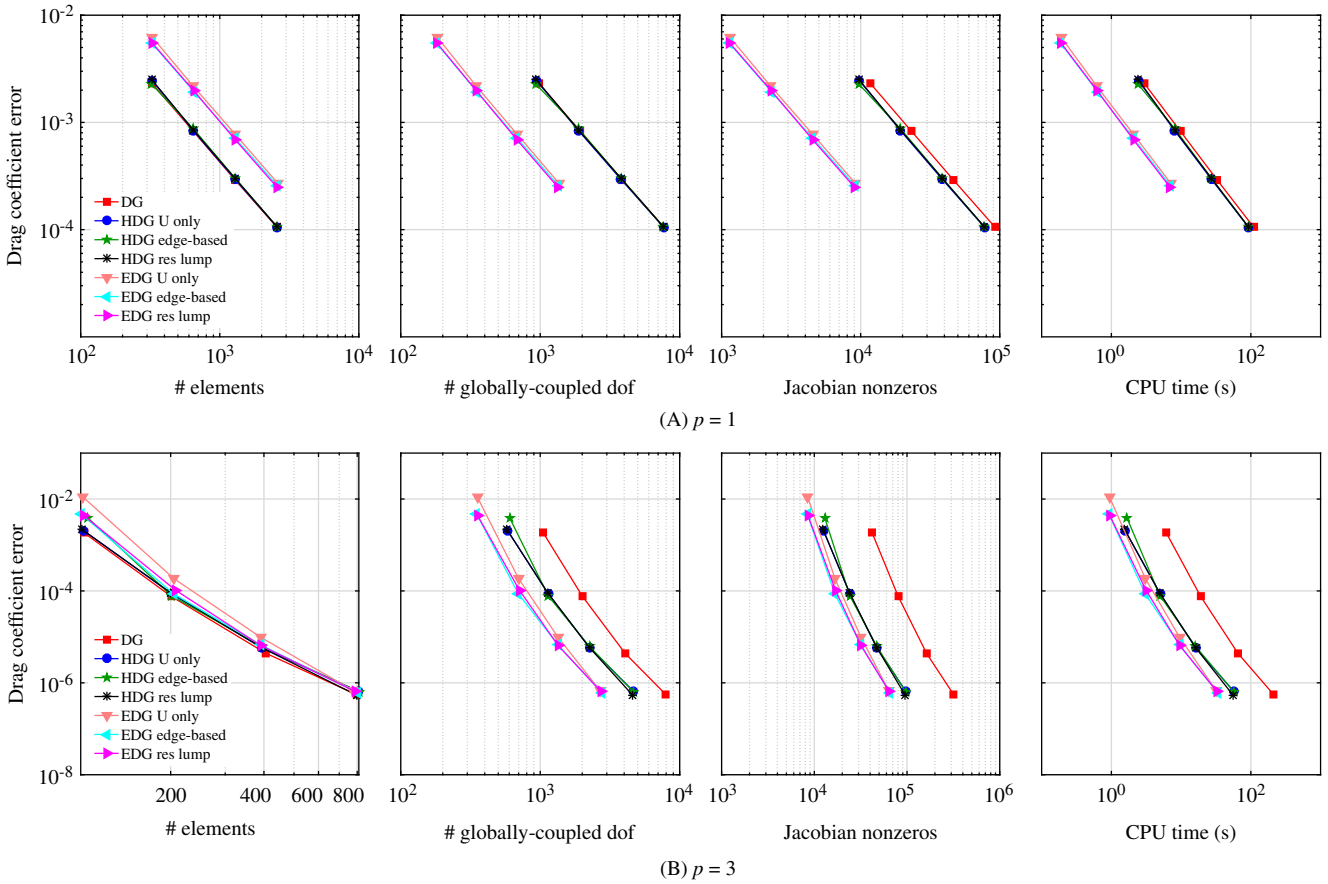


FIGURE 8 Inviscid flow: adaptive output convergence results [Color figure can be viewed at wileyonlinelibrary.com]

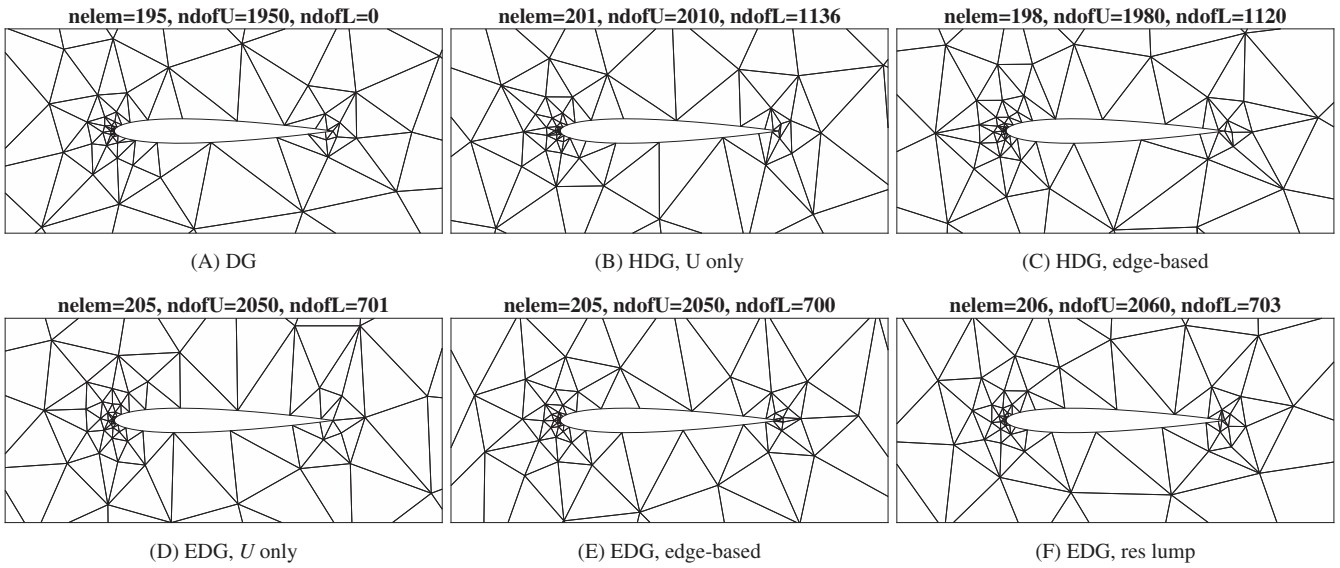
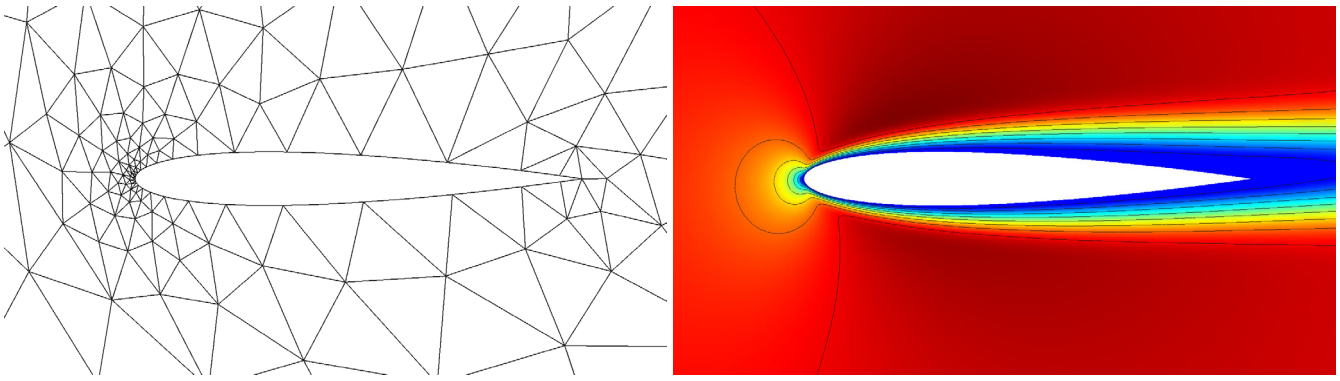


FIGURE 9 Inviscid flow: meshes adapted for $p = 3$ approximation using various discretizations and adaptive methods, with approximately 200 elements

5.2 | 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, and the exact value for error calculations is computed with $p = 4$ approximation on a mesh adapted to significantly more degrees of freedom than that used for the comparisons. Figure 10 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 10 Viscous flow: initial mesh and solution Mach contours [Color figure can be viewed at wileyonlinelibrary.com]

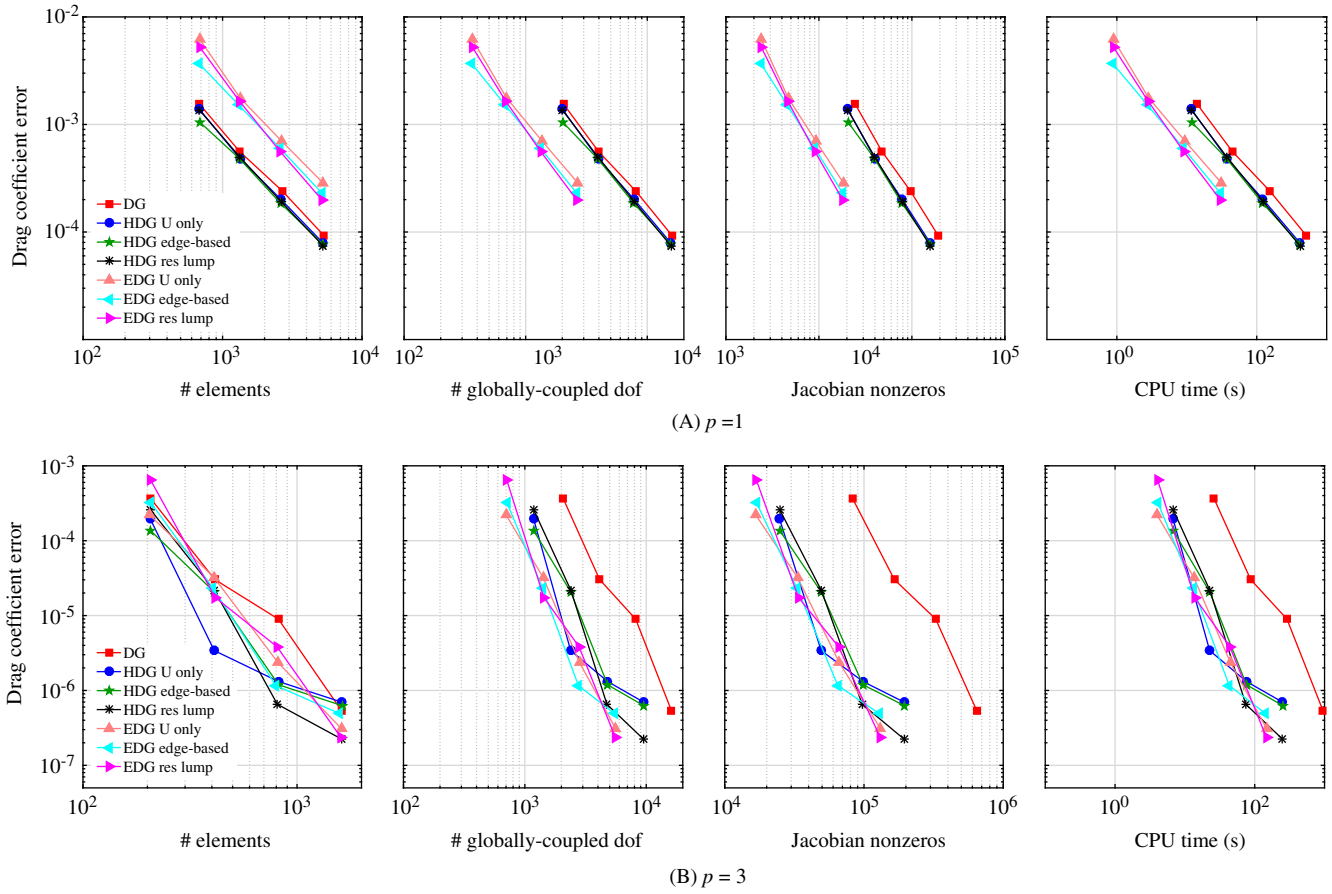


FIGURE 11 Viscous flow: adaptive output convergence results [Color figure can be viewed at wileyonlinelibrary.com]

We consider adaptive simulations using the various discretizations and edge-error localization approaches. Figure 11 shows the convergence of the output using $p = 1$ and $p = 3$ approximation orders, for the four different cost measures. As in the previous study, mesh optimizations in each case were run for 20 adaptive iterations, starting with the initial mesh, for each target number of element-based degrees of freedom. The data points shown are the average errors and costs over the last six mesh optimization iterations.

As in the inviscid case, for $p = 1$, DG and HDG are 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 and HDG, which again show similar results. In addition, edge-based and residual-lumping EDG again performs better than U-only EDG. This benefit diminishes at higher orders, but in general, incorporating the edge error estimates reduces the output errors in the optimized meshes. Note that, in contrast to the EDG case, incorporating edge errors into the HDG adaptation

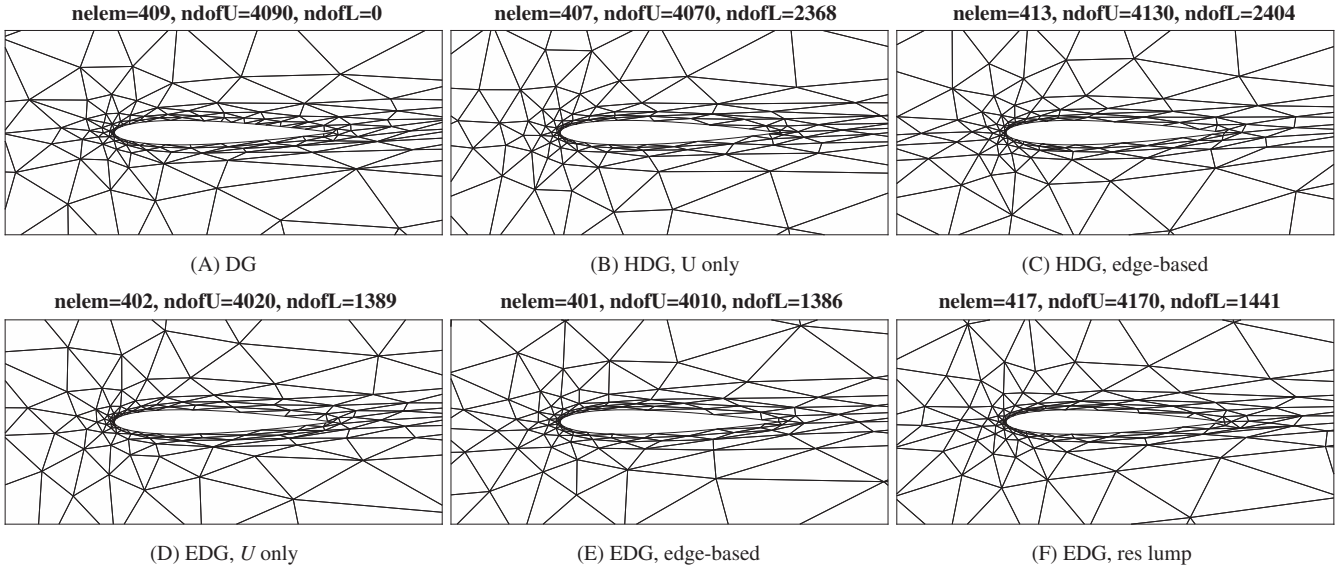


FIGURE 12 Viscous flow: meshes adapted for $p = 3$ approximation using various discretizations and adaptive methods, with approximately 400 elements

does not significantly impact its convergence. For $p = 3$, the EDG results VS number of elements are on par with DG and HDG.

Comparing the same data VS number of globally coupled degrees of freedom, we see the effect of the left-shift in the results favors the hybridized methods. At $p = 1$, EDG is clearly the most advantageous, with a factor of about three savings in degrees of freedom relative to HDG and DG. At $p = 3$, the HDG results are closer to EDG, which is still about the same factor cheaper than DG. The comparison VS Jacobian nonzeros shows similar trends – at $p = 3$, DG requires approximately a factor of five more Jacobian nonzeros than EDG and HDG. The advantage of EDG persists in CPU time for $p = 1$, and both hybridized methods are advantageous over DG at $p = 3$, by a factor of 8-10.

Figure 12 shows the final (20th iteration, 400 target elements) adapted meshes for the various discretizations at $p = 3$. 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 three 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.

5.3 | 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 the static pressure is specified on the right boundary. The RANS-SA equations, with negative turbulent viscosity modification,[35,36] 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.[37] The output of interest is the drag coefficient on the flat plate. Figure 13 shows the initial mesh, to scale and with a zoom applied in the vertical direction.

Figure 14 shows the adaptive convergence results for $p = 1$ and $p = 3$, using the various discretizations and edge-error localization approaches. 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 three 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 the DG and HDG results, at a given number of elements. When the edge-based contribution is included into EDG, the results improve. In this case, the most improvement is observed

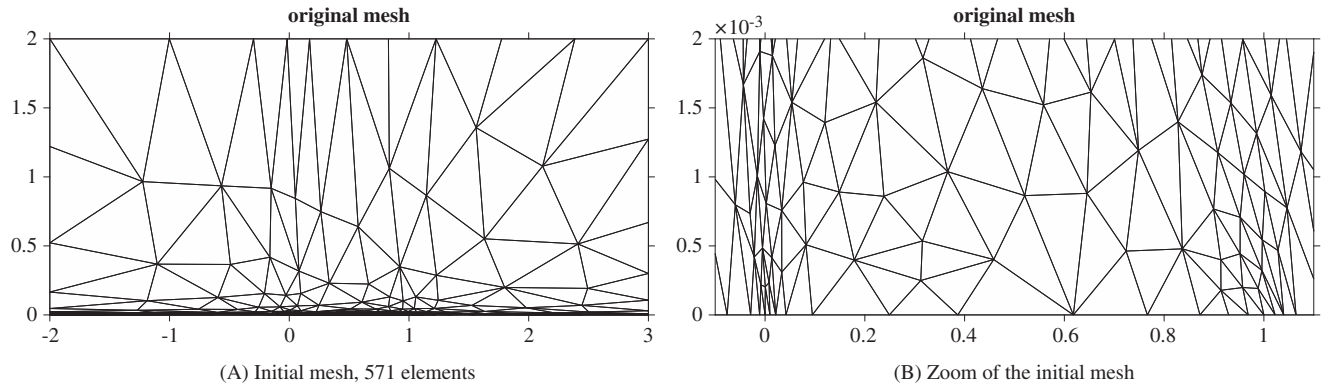


FIGURE 13 RANS flow: initial mesh, with a zoomed-in view of the boundary layer

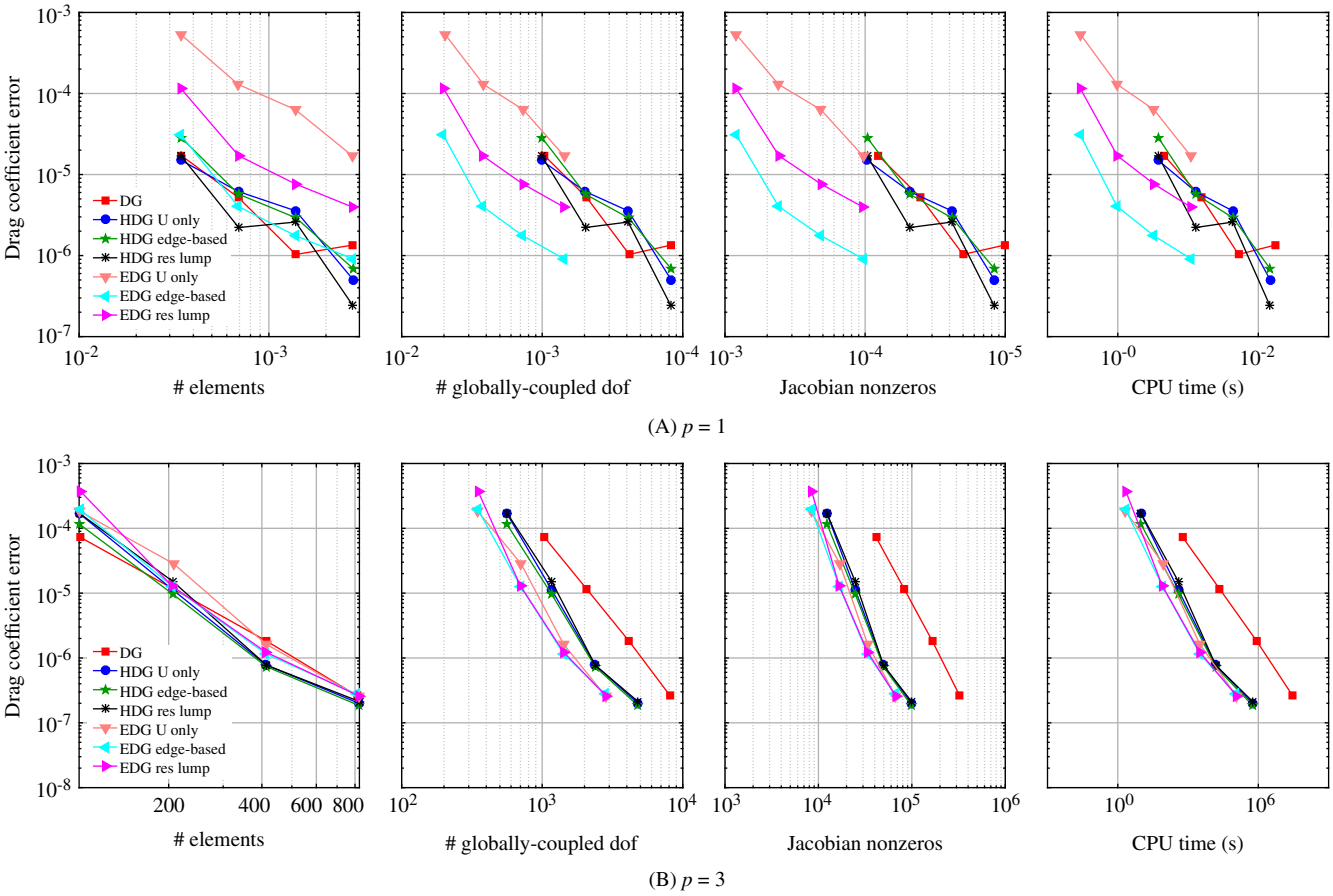


FIGURE 14 RANS flow: adaptive output convergence results [Color figure can be viewed at wileyonlinelibrary.com]

from the edge-based averaging error localization, as opposed to the residual lumping. This difference diminishes significantly for $p = 3$, where the three sets of EDG results are close and nearly coincident with DG and HDG. Note again that incorporating edge error estimates into the error localization for HDG does not significantly impact the adaptive performance.

Comparing the errors VS number of globally coupled degrees of freedom, we see that EDG is again advantageous compared to DG and HDG. For $p = 1$, the poorly performing "U-only" EDG method is actually still on par with DG and HDG, and the "edge-based" EDG results are approximately a factor of 5 cheaper. For $p = 3$, EDG still consumes the fewest degrees of freedom, but HDG is closing the gap. This gap between the hybridized methods at $p = 3$ is even lower in the

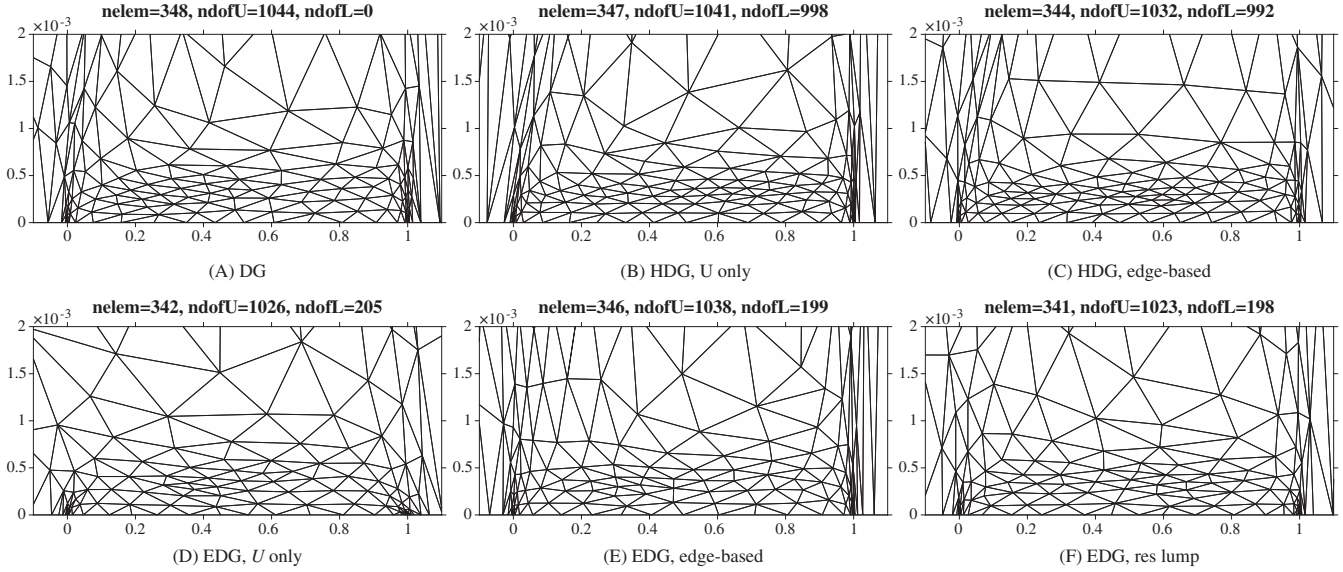


FIGURE 15 RANS flow: meshes adapted for $p = 1$ approximation using various discretizations and adaptive methods, with approximately 345 elements

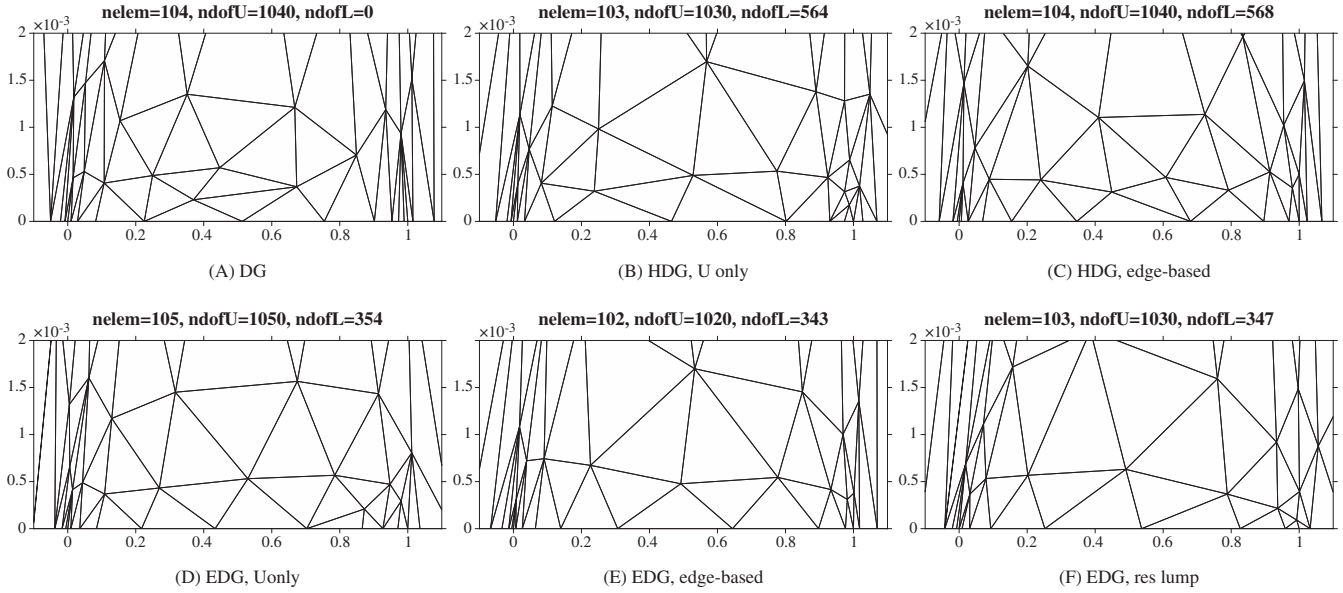


FIGURE 16 RANS flow: meshes adapted for $p = 3$ approximation using various discretizations and adaptive methods, with approximately 100 elements

comparison with Jacobian nonzeros and CPU time. At $p = 1$ for both of these cost measures, EDG remains the most efficient.

Figures 15 and 16 show the final (20th iteration, coarsest target cost) adapted meshes for the various discretizations using $p = 1$ and $p = 3$. 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. The anisotropy required at higher orders does not diminish noticeably. At $p = 1$, a marked difference between the U-only EDG mesh and the other meshes is the resolution near the leading and trailing edges of the flat plate. 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 error-localization approaches.

6 | CONCLUSIONS

This paper compares standard and hybridized discontinuous Galerkin discretizations in terms of cost and accuracy on three representative aerodynamic problems. Two hybridized methods are considered: HDG and EDG, with the latter a variant of HDG in which the trace space is continuous, a property that yields fewer globally coupled degrees of freedom, particularly at lower orders. All methods are compared in an output-based mesh optimization setting, where for a given cost, an optimal mesh for predicting a scalar output is constructed iteratively through metric-based global remeshing. The optimization is specific to each discretization, as differences in the number and type of equations yield different contributions to the adjoint-weighted residual cost estimate. To this end, two methods are proposed for accounting for the error contribution arising from the weak enforcement of flux continuity on faces in a hybridized discretization: (i) averaging to elements and creating a separate error model for the element-based quantity; and (ii) lumping face residuals to elements through an inverse Schur complement solve and using only the element-based adjoint-weighted residual. Both methods are implemented for HDG and EDG and compared alongside the simpler approach of ignoring the face error contributions during mesh optimization.

Conclusions obtained from convergence studies can be summarized as follows:

1. Uniform refinement studies may be misleading when comparing the discretizations. For a simple inviscid test case, the benefit of EDG in lower computational costs is eventually overcome by its increased dissipation at the singular trailing edge, and all methods coalesce to similar error versus cost curves for a given order.
2. Mesh optimization is critical to obtaining the expected convergence rates for each discretization, and all discretizations, including EDG, show convergence results that do not plateau when using optimized meshes.
3. HDG and DG results are generally similar for a given mesh, and HDG shows cost benefits over DG for $p > 1$ on triangles, as expected from a-priori degree of freedom calculations.
4. On optimized meshes, EDG is more dissipative than DG and HDG at $p = 1$ but this difference diminishes at higher orders.
5. Incorporating face errors into the mesh optimization procedure can significantly affect the error convergence of EDG, particularly in the presence of singularities at low orders. On the other hand, HDG is less impacted by the incorporation of face errors, and the simpler approach of optimizing only on element contributions works well.
6. Out of all the cases tested, EDG consistently yields the most efficient solution method under various cost measures: degrees of freedom, Jacobian nonzeros (memory storage), and computational time.

Remaining questions concern the performance of these methods for three-dimensional test problems, where cost savings of hybridization diminish at higher orders, and performance on more taxing aerodynamic simulations, such as those with shocks. These are the areas of our continuing work.

ACKNOWLEDGEMENTS

The authors acknowledge support from the Department of Energy under grant DE-FG02-13ER26146/DE-SC0010341, and from The Boeing Company, contract 1200596, with technical monitor Dr Mori Mani.

ORCID


Krzysztof J. Fidkowski https://orcid.org/0000-0002-5106-136X

REFERENCES

1. Reed W, Hill T. Triangular Mesh Methods for the Neutron Transport Equation. Los Alamos Scientific Laboratory Technical Report LA-UR-73-479; 1973.
2. Bassi F, Rebay S. High–order accurate discontinuous finite element solution of the 2-D Euler equations. *Journal of Computational Physics*. 1997;138:251-285.

3. Houston P, Schwab C, Süli E. Discontinuous *hp*-Finite Element Methods for First-Order Hyperbolic Problems. *SIAM Journal on Numerical Analysis*. 2000;37(5):1618-1643.
4. Cockburn B, Shu CW. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of Scientific Computing*. 2001;16(3):173-261.
5. Hartmann R, Houston P. Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. *Journal of Computational Physics*. 2002;183(2):508-532.
6. Sun S, Wheeler M. Mesh adaptation strategies for discontinuous Galerkin methods applied to reactive transport problems. In: Chu H, Savoie M, Sanchez B, eds. *International Conference on Computing, Communication and Control Technologies*. Vol 1. Austin, TX: International Institute of Informatics and Systemics; 2004:223-228.
7. Nguyen N, Peraire J, Cockburn B. an implicit high-order hybridizable discontinuous Galerkin, method for linear convection-diffusion equations. *Journal of Computational Physics*. 2009;228:3232-3254.
8. Cockburn B, Gopalakrishnan J, Lazarov R. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*. 2009;47(2):1319-1365.
9. Nguyen NC, Peraire J, Cockburn B. Hybridizable Discontinuous Galerkin Methods. In: Hesthaven JS, Rønquist EM, BT J, et al., eds. *Spectral and High Order Methods for Partial Differential Equations. 76 of Lecture Notes in Computational Science and Engineering*. Berlin Heidelberg: Springer-Verlag; 2011:63-84. https://doi.org/10.1007/978-3-642-15337-2_4.
10. Peraire J, Nguyen NC, Cockburn B. An embedded discontinuous Galerkin method for the compressible Euler and Navier-Stokes equations. AIAA Paper 2011-3228; 2011.
11. Rhebergen S, Cockburn B. Space-time hybridizable discontinuous Galerkin method for the advection-diffusion equation on moving and deforming meshes. In: de Moura CA, Kubrusly CS, eds. *The Courant-Friedrichs-Lewy (CFL) Condition*. Boston: Birkhäuser; 2013:45-63.
12. Fernandez P, Nguyen N, Peraire J. The hybridized discontinuous Galerkin method for implicit large-eddy simulation of transitional turbulent flows. *Journal of Computational Physics*. 2017;336:308-329.
13. Castro-Diaz MJ, Hecht F, Mohammadi B, Pironneau O. Anisotropic unstructured mesh adaptation for flow simulations. *International Journal for Numerical Methods in Fluids*. 1997;25:475-491.
14. Baker TJ. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*. 1997;25:243-273.
15. Buscaglia GC, Dari EA. Anisotropic mesh optimization and its application in adaptivity. *International Journal for Numerical Methods in Engineering*. 1997;40(22):4119-4136.
16. Habashi WG, Dompierre J, Bourgault Y, Ait-Ali-Yahia D, Fortin M, Vallet MG. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles. *International Journal for Numerical Methods in Fluids*. 2000;32:725-744.
17. Venditti DA, Darmofal DL. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. *Journal of Computational Physics*. 2003;187(1):22-46.
18. Park MA. Three-dimensional turbulent RANS adjoint–based error correction. AIAA Paper 2003-3849; 2003.
19. Fidkowski KJ, Darmofal DL. A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*. 2007;225:1653-1672. https://doi.org/10.1016/j.jcp.2007.02.007.
20. Yano M, Modisette J, Darmofal D. The Importance of mesh adaptation for higher-order discretizations of aerodynamics flows. AIAA Paper 2011-3852; 2011.
21. Yano M. An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes (PhD thesis). Massachusetts Institute of Technology, Cambridge, Massachusetts; 2012.
22. Pierce NA, Giles MB. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*. 2000;42(2):247-264.
23. Becker R, Rannacher R. An optimal control approach to a posteriori error estimation in finite element methods. In: Iserles A, ed. *Acta Numerica*. Cambridge, UK: Cambridge University Press; 2001:1-102.
24. Nemec M, Aftosmis MJ. Error estimation and adaptive refinement for embedded-boundary Cartesian meshes. AIAA Paper 2007-4187; 2007.
25. Fidkowski KJ, Darmofal DL. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal*. 2011;49(4):673-694. https://doi.org/10.2514/1.J050073.
26. Fidkowski K. High-order output-based adaptive methods for steady and unsteady aerodynamics. In: Deconinck H, Abgrall R, eds. *37th Advanced CFD Lectures Series*. Sint-Genesius-Rode, BelgiumL: von Karman Institute for Fluid Dynamics; 2013.
27. Dahm JP, Fidkowski KJ. Error estimation and adaptation in hybridized discontinous Galerkin methods. AIAA Paper 2014-0078; 2014
28. Woopen M, Balan A, May G, Schütz J. A comparison of hybridized and standard DG methods for target-based hp-adaptive simulation of compressible flow. *Computers & Fluids*. 2014;98:3-16.
29. Roe P. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*. 1981;43:357-372.
30. Bassi F, Rebay S. GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations. In: Cockburn B, Karniadakis G, Shu CW, eds. *Discontinuous Galerkin Methods: Theory, Computation and Applications*. Berlin, Germany: Springer; 2000:197-208.
31. Fidkowski KJA. Hybridized Discontinuous Galerkin Method on Mapped Deforming Domains. *Computers and Fluids*. 2016;139(5):80-91. https://doi.org/10.1016/j.compfluid.2016.04.004.
32. Fidkowski KJ. A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization. AIAA Paper 2016-0835; 2016
33. Borouchaki H, George P, Hecht F, Laug P, Saltel E. Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes. INRIA-Rocquencourt, France. Tech Report No. 2741; 1995.
34. Pennec X, Fillard P, Ayache N. A Riemannian framework for tensor computing. *International Journal of Computer Vision*. 2006;66(1):41-66.

35. Allmaras S, Johnson F, Spalart P. Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model. *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*. 2012;1902.
36. Ceze MA, Fidkowski KJ. Drag Prediction Using Adaptive Discontinuous Finite Elements. *AIAA Journal of Aircraft*. 2014;51(4):1284-1294. https://doi.org/10.2514/1.C032622.
37. Ceze MA, Fidkowski KJ. Constrained pseudo-transient continuation. *International Journal for Numerical Methods in Engineering*. 2015;102:1683-1703. https://doi.org/10.1002/nme.4858.

How to cite this article: Fidkowski KJ, Chen G. Output-based mesh optimization for hybridized and embedded discontinuous Galerkin methods. *Int J Numer Methods Eng*. 2020;121:867–887. https://doi.org/10.1002/nme.6248