

# Error Estimation and Adaptation in Hybridized Discontinuous Galerkin Methods

Johann P.S. Dahm\*, Krzysztof J. Fidkowski†

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

This paper presents an output-based error estimation and adaptation strategy for hybridized discontinuous Galerkin discretizations of first- and second-order systems of conservation laws. A discrete adjoint solution is obtained by a Schur-complement solver similar to that used in the primal problem. An error estimate is obtained by computing the adjoint on an enriched solution space that consists of uniform order refinement of both the element and the face approximations. The error is given by the adjoint-weighted residual, the localized contributions of which provide an adaptive error indicator for hanging-node  $h$ -refinement. Results for inviscid, laminar, and Reynolds-averaged turbulent compressible Navier-Stokes simulations in two and three dimensions demonstrate some of the potential gains of output-based adaptivity for hybridized discontinuous Galerkin discretizations.

## I. Introduction

The combination of high-order approximation and solution-based adaptivity has made possible high-fidelity computations in which resources (e.g. degrees of freedom) are distributed in an optimal manner throughout the computational domain. While high-order methods such as discontinuous Galerkin (DG) offer an attractive prospect of  $hp$ -refinement, they are sometimes criticized for their high computational expense. The fact that industrial applications still rely on traditional second-order finite volume codes suggests that the accuracy gains of high-order have not as of yet been sufficient to offset the increased expense.

To address the computational costs of high-order DG discretizations, a class of hybridized methods, collectively termed HDG, has been recently introduced. HDG methods have a number of unique properties making them especially useful for steady-state or very stiff time dependent systems of equations. These methods discretize the solution variables as well as their gradients simultaneously, so optimal convergence can be achieved in the state and its gradient.<sup>1</sup> In a standard DG method, gradients will suboptimally converge. Moreover, the resulting linear system in HDG couples only degrees of freedom together on interior faces, resulting in a system that, for sufficiently high order, is smaller than that obtained with DG. In fact, the growth rate of degrees of freedom in the linear system is decreased from  $p^d$  in DG to  $p^{d-1}$  in HDG, where  $p$  is the approximation order and  $d$  is the spatial dimension. Hence, a benefit in terms of nonzeros in the linear system is observed for sufficiently high order.

By way of a quantitative comparison, Table 1 presents the number of globally-coupled unknowns per vertex of a mesh for different discretizations. This comparison assumes typical isotropic meshes and ignores boundary effects.

\*Ph.D. Candidate, jdahm@umich.edu, AIAA Student Member

†Assistant Professor, kfid@umich.edu, AIAA Senior Member

Table 1: The numbers in the below tables indicate approximately how many degrees of freedom (per equation of a system) we need per vertex of the mesh. Note, CG refers to the continuous Galerkin finite-element method.

<i>Triangles:</i>					<i>Quadrilaterals:</i>				
method	$p = 1$	$p = 2$	$p = 3$	$p = 4$	method	$p = 1$	$p = 2$	$p = 3$	$p = 4$
DG	6	12	20	30	DG	4	9	16	25
CG	1	4	9	16	CG	1	4	9	16
HDG	6	9	12	15	HDG	4	6	8	10

<i>Tetrahedra:</i>					<i>Hexahedra:</i>				
method	$p = 1$	$p = 2$	$p = 3$	$p = 4$	method	$p = 1$	$p = 2$	$p = 3$	$p = 4$
DG	24	60	120	210	DG	8	27	64	125
CG	1	8.2	27.4	64.6	CG	1	8	27	64
HDG	36	72	120	180	HDG	12	27	48	75

Some forms of mixed hybridized Galerkin methods have been used for quite some time. These methods were developed for second-order elliptic problems.<sup>2,3</sup> More recently hybrid mixed methods have been applied to the convection-dominated regime,<sup>4,5,6</sup> and they have to-date been expanded to include nonlinear convection-diffusion systems of equations, and they have been applied to adaptive discretizations of two-dimensional laminar compressible Navier-Stokes equations.<sup>7,8</sup>

The aim of this paper is twofold: first, to present the HDG discretization with practical implementation details for a general set of nonlinear convection-diffusion equations; and second, to compare the effectiveness of adaptivity of the HDG and DG discretizations on a range of problems. The rest of the paper is organized as follows. The discretization of the equations used for the comparison is described in Section II. Then the error estimation and adaptation procedure is described in general for both methods in Section III. Section IV shows adaptive results ranging from scalar advection-diffusion problems to Reynolds-averaged turbulent fluid flow. Finally, in Section V, we conclude and look forward to additional improvements that can be made in the HDG discretization and in output-based methods applied to HDG.

## II. Discretization

The equations considered here arise from second-order conservation laws. In general, these are written as a system of equations in the form

$$\partial_t \mathbf{u} + \partial_i \mathbf{H}_i(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0} \quad \text{in } \Omega, \quad (1)$$

where  $\mathbf{u} \in \mathbb{R}^s$  is the state vector,  $\mathbf{H}_i \in \mathbb{R}^s$  is the  $i^{\text{th}}$  component of the total flux,  $1 \leq i \leq d$  indexes the spatial dimension  $d$ , and summation is implied on the repeated index  $i$ . The total flux is decomposed into convective and diffusive parts,

$$\mathbf{H}_i = \mathbf{F}_i(\mathbf{u}) + \mathbf{G}_i(\mathbf{u}, \nabla \mathbf{u}), \quad (2)$$

$$\mathbf{G}_i(\mathbf{u}, \nabla \mathbf{u}) = -\mathbf{K}_{ij}^v(\mathbf{u}) \partial_j \mathbf{u}, \quad (3)$$

where  $\mathbf{F}_i \in \mathbb{R}^s$  is the  $i^{\text{th}}$  component of the inviscid/convective flux,  $\mathbf{G}_i \in \mathbb{R}^s$  is the  $i^{\text{th}}$  component of the viscous flux, and  $\mathbf{K}_{ij}^v \in \mathbb{R}^{s \times s}$  is the  $(i, j)$  component of the viscous diffusivity tensor.

Both methods considered here are finite element methods in which the state is spatially approximated by polynomials on non-overlapping elements, which we denote by  $\Omega_e$  with  $e = 1 \dots N_e$  being

the element index, together forming the set of elements or tessellation  $T_h$  of the spatial domain. The set of interior faces is denoted by  $\mathcal{E}_h$ . Therefore, the approximated state vector  $\mathbf{u}_h$  formally lies in the space  $\mathcal{V}_h = [\mathcal{V}_h]^s$  where

$$\mathcal{V}_h = \{v \in L^2(\Omega) : v|_{\Omega_e} \in \mathcal{P}^p(\Omega_e) \forall \Omega_e \in T_h\}. \quad (4)$$

The polynomial order  $p$  could vary over the elements, but in this work we consider no varying polynomial degree in space. We next briefly describe the DG method employed, then explain in greater detail the HDG method.

### A. Discontinuous Galerkin

The DG weak form is obtained by integrating the equations (1) against test functions,  $\mathbf{w}_h$ , which lie in the same space as the solution,

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h) = \sum_e \mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) = 0, \quad \forall \mathbf{w}_h \in \mathcal{V}_h.$$

The formulation used in this work employs the second form of Bassi and Rebay (BR2) for the viscous discretization.<sup>9</sup> Details of the discretization can be found in many previous works.<sup>10</sup> Briefly, for the viscous part, the idea is to rewrite the second-order equation as a system of first-order equations. This introduces an auxiliary flux variable, which BR2 then locally eliminates from the resulting bilinear form.<sup>11</sup> The resulting semilinear form restricted to each element is

$$\begin{aligned} \mathcal{R}_h(\mathbf{u}_h, \mathbf{w}_h|_{\Omega_e}) &= \int_{\Omega_e} \mathbf{w}_h^T \partial_t \mathbf{u}_h \, d\Omega - \int_{\Omega_e} \partial_i \mathbf{w}_h^T \mathbf{H}_i \, d\Omega \\ &+ \int_{\partial\Omega_e} \mathbf{w}_h^{+T} \left( \widehat{\mathbf{F}}_i n_i + \widehat{\mathbf{G}}_i n_i \right) \, ds + \int_{\partial\Omega_e} \partial_i \mathbf{w}_h^{+T} \widehat{\mathbf{K}}_{ij}^v (\mathbf{u}_h^+ - \widehat{\mathbf{u}}) \, ds, \end{aligned} \quad (5)$$

where  $(\cdot)^+$  denotes the quantity is taken from the inside of the element, and  $\widehat{(\cdot)}$  denotes some form of averaging. The convective flux on interfaces,  $\widehat{\mathbf{F}}$ , is obtained using the Roe approximate Riemann solver.<sup>12</sup> The BR2 method takes the state trace vector  $\widehat{\mathbf{u}} = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$ , and the trace viscous flux

$$\widehat{\mathbf{G}}_i n_i = \frac{1}{2} (\mathbf{G}_i^+ + \mathbf{G}_i^-) n_i^+ - \eta \frac{1}{2} (\boldsymbol{\delta}_i^+ + \boldsymbol{\delta}_i^-) n_i^+.$$

The stabilization parameter  $\eta$  depends on the geometry of the elements and  $\boldsymbol{\delta}_i^+, \boldsymbol{\delta}_i^- \in [\mathcal{V}_h]^d$  are obtained by solving an additional system of equations locally for each face.

### B. Hybridized discontinuous Galerkin

Assuming the second-order term is present in (1), in HDG we rewrite the second-order equations as a system of first-order equations. The system of equations becomes,

$$\begin{aligned} \mathbf{q}_i - \partial_i \mathbf{u} &= \mathbf{0}, \quad i = 1 \dots d, \\ \partial_t \mathbf{u} + \partial_i \mathbf{H}_i(\mathbf{u}, \vec{\mathbf{q}}) &= \mathbf{0}. \end{aligned} \quad (6)$$

Instead of locally eliminating  $\vec{\mathbf{q}}$ , HDG solves for it. That is,  $\vec{\mathbf{q}}_h \in [\mathcal{V}_h]^d$  is an approximated unknown. Some HDG works in the past have instead defined the diffusive *flux* as the auxiliary variable, but we have found the definition  $\vec{\mathbf{q}} = \nabla \mathbf{u}$  to be more versatile.

DG defined a unique flux on an interface using the states on the two adjacent elements. HDG, on the other hand, defines *two* fluxes at every point on an interface: one flux for the left element

and one flux for the right element. Our conservation instinct tells us that multi-valued fluxes may be problematic, but HDG resolves this duplicity by a weak flux-continuity enforcement, as we will shortly see.

A “one-sided” flux in HDG is computed with the state on one of the elements,  $(\vec{\mathbf{q}}_h, \mathbf{u}_h)$ , and a “trace” state  $\hat{\mathbf{u}}$  defined on all faces. This flux, dotted with the normal, is

$$\hat{\mathbf{H}}_i n_i = \hat{\mathbf{F}}_i n_i + \hat{\mathbf{G}}_i n_i = \mathbf{F}_i(\hat{\mathbf{u}}) n_i + \mathbf{G}_i(\hat{\mathbf{u}}, \mathbf{q}_h) n_i + \boldsymbol{\tau}(\hat{\mathbf{u}}, \vec{n})(\mathbf{u}_h - \hat{\mathbf{u}}), \quad \boldsymbol{\tau} = \boldsymbol{\tau}^{\mathbf{F}} + \boldsymbol{\tau}^{\mathbf{G}}, \quad (7)$$

where  $\vec{n}$  (components  $n_i$ ) is an element-outward-pointing normal, and  $\boldsymbol{\tau}$  is a stabilization tensor discussed shortly. The trace state is defined on interior faces, and the degrees of freedom in the trace state approximation are treated as additional unknowns. On boundary faces, we compute a boundary state from the neighboring element state and the boundary condition. We denote by  $\hat{\mathbf{u}} = \boldsymbol{\lambda}_h$  the unknown degrees of freedom on interior faces, so  $\boldsymbol{\lambda}_h \in \mathcal{M}_h = [\mathcal{M}_h]^s$  where

$$\mathcal{M}_h = \{ \mu \in L^2(\mathcal{E}_h) : \mu|_F \in \mathcal{P}^p(F) \forall F \in \mathcal{E}_h \}.$$

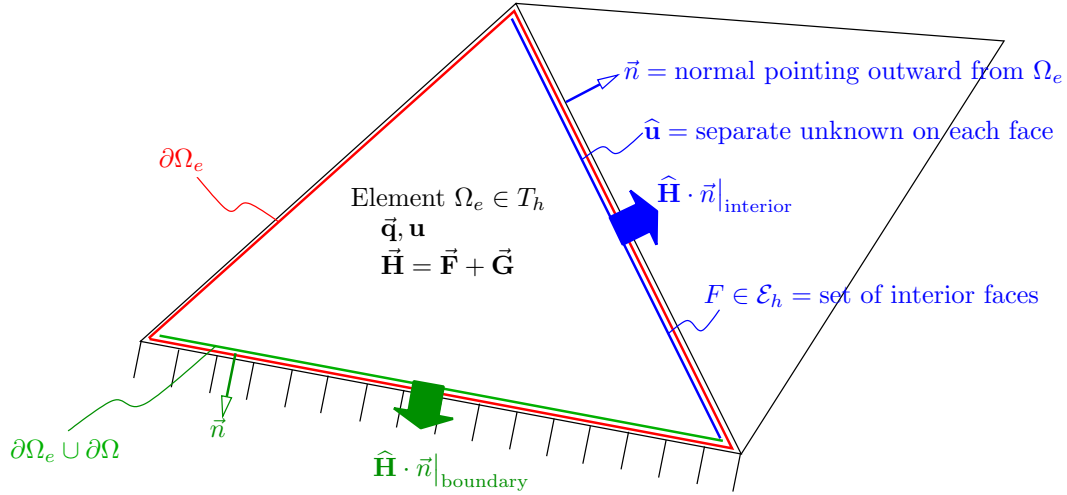


Figure 1: Definition of various quantities used in the HDG method.

The addition of the gradient variable as an unknown necessitates another set of equations that enforce  $\vec{\mathbf{q}} = \nabla \mathbf{u}$  weakly. Similarly, the addition of the trace states as unknowns requires more equations – these are the weak enforcements of flux continuity on the interfaces, as mentioned above. Hence, the full set of semilinear forms reads,

$$\mathcal{R}_h^{\mathbf{q}}(\cdot, \vec{\mathbf{v}}_h) = \sum_{e=1}^{N_e} \left\{ \int_{\Omega_e} \mathbf{v}_{h,i}^T \mathbf{q}_{h,i} d\Omega + \int_{\Omega_e} \partial_i \mathbf{v}_{h,i}^T \mathbf{u}_h d\Omega - \int_{\partial\Omega_e} \mathbf{v}_{h,i}^T \hat{\mathbf{u}} n_i ds \right\} = 0, \quad \forall \vec{\mathbf{v}}_h \in [\mathcal{V}_h]^d \quad (8)$$

$$\mathcal{R}_h^{\mathbf{u}}(\cdot, \mathbf{w}_h) = \sum_{e=1}^{N_e} \left\{ \int_{\Omega_e} \mathbf{w}_h^T \partial_t \mathbf{u}_h d\Omega - \int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \vec{\mathbf{H}} d\Omega + \int_{\partial\Omega_e} \mathbf{w}_h^T \hat{\mathbf{H}} \cdot \vec{n} ds \right\} = 0, \quad \forall \mathbf{w}_h \in \mathcal{V}_h \quad (9)$$

$$\mathcal{R}_h^{\boldsymbol{\lambda}}(\cdot, \boldsymbol{\mu}_h) = \sum_{F \in \mathcal{E}_h} \int_F \boldsymbol{\mu}_h^T [[\hat{\mathbf{H}}_i n_i]] ds = 0, \quad \forall \boldsymbol{\mu}_h \in \mathcal{M}_h \quad (10)$$

In practice, the above semilinear forms are evaluated using element-local or face-local basis functions as the test functions, resulting in discrete residual vectors on the elements and faces. We denote

these residual vectors as  $\mathbf{R}^Q, \mathbf{R}^U$  associated with elements, and  $\mathbf{R}^\Lambda$  associated with interior faces. Also denoting lumped vectors of unknowns via  $\mathbf{Q}, \mathbf{U}$ , and  $\mathbf{\Lambda}$ , the resulting discrete equations are

$$\begin{aligned}\mathbf{R}^Q(\mathbf{Q}, \mathbf{U}, \mathbf{\Lambda}) &= 0, \\ \mathbf{R}^U(\mathbf{Q}, \mathbf{U}, \mathbf{\Lambda}) &= 0, \\ \mathbf{R}^\Lambda(\mathbf{Q}, \mathbf{U}, \mathbf{\Lambda}) &= 0.\end{aligned}\tag{11}$$

### C. Stabilization

An important choice in the HDG discretization is the type of stabilization, namely the tensor  $\boldsymbol{\tau}$  in (7). Numerical evidence for systems of equations such as Navier-Stokes indicates that the choice of flux stabilization, especially for the diffusive part, has an important effect on the stiffness of the linear systems and the robustness of the nonlinear solver.

Some guidelines for choosing this stabilization term can be found in previous works.<sup>13</sup> The choice of convective stabilization is similar to choosing the numerical flux for DG. Nguyen, *et al* present an energy inequality for the flux stabilization and give several examples.<sup>14</sup> Literature since then has primarily focused on two types of convective stabilization.<sup>1,15,8</sup> The first, and most straightforward, is to simply take  $\boldsymbol{\tau}^{\mathbf{F}} = |c_{\max}|\mathbf{I}$ , where  $\mathbf{I}$  is the  $s \times s$  identity matrix. This can be shown to be equivalent to the Lax-Friedrichs flux. The second, and the approach employed in this work, yields a ‘‘Roe’’ flux from the choice

$$\boldsymbol{\tau}^{\mathbf{F}}(\hat{\mathbf{u}}, \vec{n}) = \left| \frac{\partial \vec{\mathbf{F}}_i(\hat{\mathbf{u}}) n_i}{\partial \hat{\mathbf{u}}} \right|.\tag{12}$$

Most previous works on HDG for systems of nonlinear convection diffusion equations have used a constant diffusive flux stabilization, such as  $\boldsymbol{\tau}^{\mathbf{G}} = (\nu/\ell_{\text{visc}})\mathbf{I}$ , where  $\ell_{\text{visc}}$  is a viscous length scale. However, this approach adds viscous stabilization to all equations, even those without a second-order term. Our experience is that this adds stiffness to the problem and hinders convergence. Instead, in the results shown here we set, unless otherwise noted,

$$\boldsymbol{\tau}^{\mathbf{G}} = \frac{n_i \mathbf{K}_{ij}^v n_j}{\ell_{\text{visc}}}.\tag{13}$$

This has the benefit of adding stabilization only to the equations that require it, and adds it proportional to the amount of viscosity in the equation. The somewhat-ambiguous length scale  $\ell_{\text{visc}}$ , must still be chosen, and we are presently working to remedy this remaining ‘‘knob’’ in the discretization.

### D. Hybridization

The introduction of the trace states, and the further localization of the numerical flux allows a linear system to be efficiently built coupling only degrees of freedom of the trace together through a process known as *static condensation*. The system of equations resulting from a Newton iteration of the nonlinear set of equations in (11) have the form,

$$\left[ \begin{array}{cc|c} \frac{\partial \mathbf{R}^Q}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^Q}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^Q}{\partial \mathbf{\Lambda}} \\ \frac{\partial \mathbf{R}^U}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^U}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^U}{\partial \mathbf{\Lambda}} \\ \hline \frac{\partial \mathbf{R}^\Lambda}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^\Lambda}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^\Lambda}{\partial \mathbf{\Lambda}} \end{array} \right] = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix},\tag{14}$$

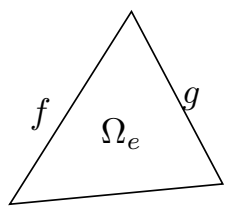
where we have divided the matrix into blocks  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ ,  $\mathbf{D}$  as indicated by the lines on the left-hand side. At this point we introduce the notation  $\mathbf{QU}$  to denote the vector of unknowns on element interiors, i.e. the concatenation of  $\mathbf{Q}$  and  $\mathbf{U}$ . So we can write  $\mathbf{A} = \partial \mathbf{R}^{QU} / \partial \mathbf{QU}$ .

Taking the Schur complement of the matrix in (14) results in a smaller matrix  $\mathbf{K}$ , which is the same size as  $\mathbf{D}$ ,

$$\mathbf{K} = \mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}. \quad (15)$$

This matrix contains information on coupling between face degrees of freedom after taking into account the ‘‘transmission’’ of residuals across elements through element-interior degrees of freedom. The sparsity pattern of  $\mathbf{K}$  is such that off-diagonal blocks (groups of degrees of freedom associated with faces) of  $\mathbf{K}$  are nonzero only if the two faces are adjacent to the same element. Note that the inversion of  $\mathbf{A}$  in (14) is an element-local operation due to the fact that element-interior degrees of freedom are not coupled to each other.

The reduced Jacobian (15) looks computationally expensive to build, but in fact may be efficiently computed by a single loop through elements  $\Omega_e$ . On each element, we compute the element-interior and element-boundary (i.e. face) integral contributions to the residuals (and their linearizations) in (11). Consider all pairs of faces adjacent to  $\Omega_e$ : label them  $f$  and  $g$  as illustrated below. The contribution from the pair of faces added to the reduced Jacobian is



$$\mathbf{K}_{fg} += \frac{\partial \mathbf{R}_f^\Lambda}{\partial \Lambda_g} \delta_{fg} - \underbrace{\begin{bmatrix} \frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{Q}_e} & \frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{U}_e} \\ \frac{\partial \mathbf{R}_e^U}{\partial \mathbf{Q}_e} & \frac{\partial \mathbf{R}_e^U}{\partial \mathbf{U}_e} \end{bmatrix}}_{\mathbf{A}_e^{-1}} \begin{bmatrix} \frac{\partial \mathbf{R}_e^Q}{\partial \Lambda_g} \\ \frac{\partial \mathbf{R}_e^U}{\partial \Lambda_g} \end{bmatrix}.$$

In this expression, the subscripts on the residuals and states indicate the components on an element ( $e$ ), or on a face ( $f, g$ ). From an implementation standpoint, when assembling  $\mathbf{K}$ , a certain amount of local storage is needed for the local matrix blocks above. This local storage may be preallocated and reused for efficiency.

In a direct implementation, extra cost is incurred when inverting the  $\mathbf{A}_e$  matrix, which becomes very large in multiple dimensions when diffusive terms are present. However, this cost can be greatly reduced from the observation that again  $\mathbf{A}_e$  is not without a block structure, which may be exploited in calculating its inverse. In fact for HDG,  $\frac{\partial \mathbf{R}_e^Q}{\partial \mathbf{Q}_e}$  consists of  $d$  block-diagonal mass matrices, which are the same for every equation of the system and which are easily invertible (and likely already available from real or pseudo time-stepping). Thus the cost of inverting the  $\mathbf{A}_e$  matrix need not be significantly higher for diffusive problems compared to inviscid problems.

As  $\mathbf{K}$  is built, a similar procedure is performed to compute the statically-condensed residual vector  $\tilde{\mathbf{R}} = \mathbf{R}^\Lambda - \mathbf{CA}^{-1}\mathbf{R}^{QU}$ . The Newton update for the discrete unknowns  $\Delta \mathbf{QU}$  and  $\Delta \Lambda$  can then be reduced to solving

$$\mathbf{K}\Delta\Lambda + \tilde{\mathbf{R}} = \mathbf{0}.$$

After the above linear system of algebraic equations is solved, the updates on the elements  $\Delta \mathbf{QU}$  may be obtained element-wise by applying an element-wise static evaporation procedure,

$$\Delta \mathbf{QU} = -\mathbf{A}^{-1} (\mathbf{R}^{QU} + \mathbf{B}\Delta\Lambda).$$

## E. Solver

The nonlinear solver uses a preconditioned Newton-Krylov method with a pseudo-transient continuation.<sup>16</sup> The linear systems are solved with a preconditioned restarted General Minimal Residual

(GMRES) Krylov subspace method. The initial guess is usually a constant state, and the pseudo timestep is increased for every full Newton step until the nonlinear residuals drop below a certain tolerance (typically 8-10 orders of magnitude). Care must be taken when adding the pseudo timestep to the HDG system. Simply adding the weighted mass-matrices to the on-diagonal blocks of  $\mathbf{K}$  is incorrect – the weights must be added to the on-diagonal blocks of  $\partial \mathbf{R}^U / \partial \mathbf{U}$  prior to static condensation.

Two preconditioners are used in this work: line-based Jacobi smoothing and zero-fill incomplete lower-upper (ILU0) factorization. The line-based Jacobi algorithm for DG<sup>17</sup> employs a connectivity measure based on the discretization of a linear advection-diffusion equation to determine the inter-element coupling. In HDG, we instead need to know how faces are coupled, after static condensation, and for this we have found that the Frobenius norm of the off-diagonal Jacobian blocks works well as a line connectivity measure.

For the linear systems resulting from the problems in this work, the ILU0 preconditioner with minimal discarded fill<sup>18,19</sup> seems to be superior to a line-based Jacobi smoothing for HDG, while this is not always the case for DG. This finding motivates further investigation into solvers.

## F. Parallelization

For large cases, even the reduced-size matrix  $\mathbf{K}$  may be too large to fit on the memory allotted to a typical computational core. Furthermore, the local element operations involved in the static condensation are embarrassingly-parallel, and hence splitting the problem up over many cores is beneficial for performance.

When running in parallel, we partition the domain so that each processor is responsible for a subset of the unknowns in the problem. Over all processors, these subsets need to be disjoint, and their union needs to be the complete set of unknowns. However, for communication purposes, the partitions residing on each processor (self) also include what we call “halo” degrees of freedom, which are unknowns that reside on another processor but which are used by the self processor for constructing the self residuals. For example, in HDG, to construct a Schur-complemented face residual, we need unknowns on all adjacent faces, which are faces that border either of the two elements next to the face in question.

Figure 2 defines the various types of faces and elements that occur when running HDG in parallel. Partitioning is performed using ParMetis,<sup>20</sup> called on the graph of elements. This returns a load-balanced partition of the elements. However, we also care about load-balancing interior faces. Interior faces between self elements are automatically associated with the processor of the elements. Interior faces that straddle processor interfaces, i.e. those faces for which the adjacent elements are on different processors, are assigned to the processor that, at the time of query, has fewer interior faces. This is a greedy approach that works well as long as the partitioning is not too fine-grained. Finally, Jacobian matrix storage is allocated for halo unknowns so that, after communication, matrix-vector multiplication can proceed smoothly.

## III. Output-based adaptation

The adaptation strategy employed in this work considers an output of interest, calculates an error estimate, and based on this estimate adapts the mesh spatially to lower the numerical error in the output. The central ideas of output-based error estimation and mesh adaptation have been presented in numerous previous works,<sup>21</sup> including recently in HDG,<sup>8</sup> and in the following we present a brief overview and some specifics relevant to HDG.

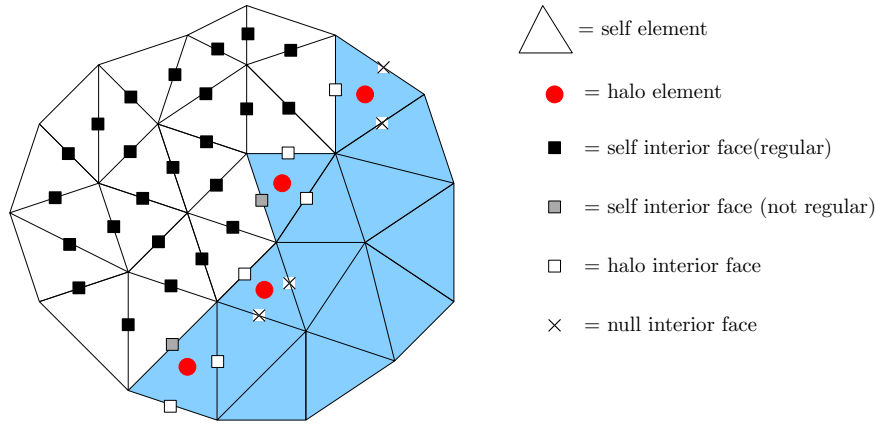


Figure 2: Definition of self and halo elements and faces. The “self” processor in this case contains elements shaded in white and interior faces marked with solid black or gray squares. Interior faces are “regular” if both adjacent elements are self elements. Calculations on non-regular self faces may have to wait for communication of halo element data. Each self face needs to talk to a set of surrounding faces for calculating residuals. This is why we introduce halo interior faces, marked with white squares: the self processor does not own unknowns on these halo faces, but it makes space for them and receives values for them during inter-processor communication. Halo elements, marked with a red circle, are needed for computing residuals on non-regular self interior faces.

### A. Output error estimation

The output error estimation uses an adjoint-weighted residual approach that calculates the approximate output error relative to a finer discretization space. Consider first a scalar output  $J_H(\mathbf{U}_H)$  computed on a coarse (current) discretization level ( $H$ ) with DG. The solution  $\mathbf{U}_H$  satisfies

$$\mathbf{R}_H(\mathbf{U}_H) = \mathbf{0}. \quad (16)$$

Now, suppose we have access to a finer discretization space ( $h$ ). Injecting the coarse solution into the fine space gives  $\mathbf{U}_h^H \equiv \mathbf{I}_h^H \mathbf{U}_H$ . This state does not in general satisfy the fine-space equations, so that

$$\mathbf{R}_h(\mathbf{U}_h^H) \neq \mathbf{0}. \quad (17)$$

The fine-space discrete adjoint vector,  $\Psi_h$ , associated with the output of interest gives the sensitivity of the output to residual source perturbations. From the point of view of the fine space, (17) is a (negative) residual source perturbation, which means that we can estimate the output error via an adjoint-weighted residual,

$$\delta J_h = J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) = -\Psi_h^T \mathbf{R}_h(\mathbf{U}_h^H).$$

We can also replace  $\Psi_h$  in this equation with  $\Psi_h - \mathbf{I}_h^H \Psi_H$ , which will not change the error estimate for discretizations with strict Galerkin orthogonality, and which otherwise will target the “remaining error” (on which we want to adapt). The fine-space adjoint is then obtained by solving<sup>21</sup>

$$\left( \frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h} \right)^T \Psi_h + \left( \frac{\partial J_h}{\partial \mathbf{U}_h} \right)^T = \mathbf{0}. \quad (18)$$



In HDG, the equations produce three sets of discrete residuals,  $\mathbf{R}_h^Q$ ,  $\mathbf{R}_h^U$ , and  $\mathbf{R}_h^\Lambda$ , and there are adjoint variables associated with each residual. The adjoint system is given by (dropping the subscript  $h$  for clarity)

$$\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix} \begin{bmatrix} \Psi^{QU} \\ \Psi^\Lambda \end{bmatrix} + \begin{bmatrix} \frac{\partial J}{\partial \mathbf{QU}}^T \\ \frac{\partial J}{\partial \Lambda}^T \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Statically condensing out the element-interior degrees of freedom, we obtain the following system for the face adjoints,

$$\underbrace{[\mathbf{D}^T - \mathbf{B}^T \mathbf{A}^{-T} \mathbf{C}^T]}_{\mathbf{K}^T} \Psi^\Lambda + \begin{bmatrix} \frac{\partial J}{\partial \Lambda}^T & -\mathbf{B}^T \mathbf{A}^{-T} \frac{\partial J}{\partial \mathbf{QU}}^T \end{bmatrix} = \mathbf{0}.$$

So, as expected, the Schur-complement adjoint system matrix is just the transpose of the corresponding matrix for the primal system. The adjoint residual is also of similar form. For HDG, the output error estimate is

$$\delta J \approx \underbrace{-(\Psi_h^Q)^T \mathbf{R}_h^Q}_{\delta J^Q} - \underbrace{(\Psi_h^U)^T \mathbf{R}_h^U}_{\delta J^U} - \underbrace{(\Psi_h^\Lambda)^T \mathbf{R}_h^\Lambda}_{\delta J^\Lambda}, \quad (19)$$

where all the residuals are evaluated using the coarse state injected into the fine space,  $\mathbf{Q}_h^H$ ,  $\mathbf{U}_h^H$ ,  $\Lambda_h^H$ . For the fine space, we increment the approximation order by one on each element and interface. We obtain the fine-space adjoint by solving exactly or approximately (via iterative block-Jacobi smoothing) on this fine space. Note that in (19) we separated the error estimate into three components, one for each residual.

## B. Adaptation

In the present work, we restrict our attention to adapting the mesh ( $h$ -adaptation). To do so, an error indicator must be formed for each element. The adjoint-weighted residual error estimate may be localized by considering only the terms of the summation using the degrees of freedom on the mesh element. In DG, the error indicator associated with an element is given by

$$\epsilon_e^{\text{DG}} = |\delta \Psi_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)|, \quad (20)$$

where the subscript  $e$  on the adjoint and residual denotes components associated with element  $e$ .

In HDG, we can localize  $\delta J^Q$  and  $\delta J^U$  to elements, since both of these expressions involve residuals associated with test functions on element interiors. Similarly, we can localize  $\delta J^\Lambda$  to *faces* because  $\delta J^\Lambda$  involves residuals obtained from test functions on faces. What we do with these error indicators depends in part on the chosen adaptation mechanics. In the most general case, we could consider a situation where faces and elements are adapted independently, e.g. through approximation order; we could then incorporate some measure of cost to define a merit function to decide which elements or faces needed to be refined.

At present, however, since we restrict our attention to pure  $h$ -refinement, we only require error indicators on the elements. The elemental error indicators are calculated as

$$\epsilon_e^{\text{HDG}} = \epsilon_e^Q + \epsilon_e^U, \quad (21)$$

where  $\epsilon_e^Q$  and  $\epsilon_e^U$  are localizations of  $\delta J^Q$  and  $\delta J^U$ , respectively. We could incorporate the localization of  $\delta J^\Lambda$  from the faces, by, for example, equally distributing this error indicator to the

two adjacent elements. However, in this work we simply ignore these face error indicators when calculating the elemental error indicators for adaptation. The justification for this is that  $\delta J^\Lambda$  is typically much smaller in magnitude in comparison to  $\delta J^Q$  and  $\delta J^U$  – in fact, in certain cases we can prove that  $\delta J^\Lambda = 0$ .

Once the error indicator is obtained, a certain number of the highest indicators are chosen to be adapted. These elements are adapted, allowing for the creation of hanging nodes on element boundaries. This procedure is favorable, especially toward HDG, because very few new faces are introduced and the adaptation is better able stay local in regions which require it most. To the authors’ knowledge, this is the first adaptive study using HDG with hanging nodes.

## IV. Results

In this section we demonstrate the adaptive HDG method on a number of test cases. First, a linear scalar convection-diffusion equation on a square domain. Then we consider cases of aerospace interest: inviscid and viscous turbulent Navier-Stokes equations.

### A. Scalar Convection-Diffusion

First we consider the linear convection-diffusion equations, where

$$\begin{aligned}\vec{F}(u) &= \vec{v}u, \quad \vec{v} = [0.6, 0.8], \\ \vec{G}(\nabla u) &= -\nu \nabla u, \quad \nu = 0.02.\end{aligned}$$

The domain is a unit square,  $\Omega = [0, 1]^2$ , and functional Dirichlet boundary conditions are imposed,  $u|_{\partial\Omega} = g(x, y)$ , where

$$g(x, y) = \exp[0.5 \sin(-4x + 6y) - 0.8 \cos(3x - 8y)].$$

For the HDG stabilization, we set  $\tau = |\vec{v}| + \nu$ , so the viscous length scale is set to unity. The output is set to the integral of  $\vec{G} \cdot \mathbf{n}$  over the right boundary of the domain. Initially, the mesh has 64 uniform quadrilateral elements. The solution and HDG adapted mesh are plotted in Figure 3.

Uniform and adaptive refinement are compared for both DG and HDG discretizations in Figure 4. The adaptation strategy used here chooses 10% of the elements at each adaptive iteration for refinement. It is interesting to note that in this case HDG converges one order faster than DG. This occurs here because HDG is discretizing the gradient  $\vec{q}$  with an order- $p$  polynomial, which is converging optimally under these conditions. For the  $p = 1$  results, the faster convergence of  $\vec{q}$  is enabling even the HDG corrected output to converge faster than the DG corrected output. This effect disappears at  $p = 2$ , but even here the HDG corrected output seems to do as well or better than that of DG.

### B. Inviscid flow over NACA 0012 airfoil

Next, we consider adaptation for the two-dimensional inviscid Navier-Stokes equations at  $M_\infty = 0.5$ ,  $\alpha = 2^\circ$  angle of attack. Initially, the mesh consists of 140  $q = 4$  (i.e. quartically curved) quadrilateral elements. The output for adaptation is the drag on the airfoil. The Mach contours and an adapted mesh for the airfoil are shown in Figure 5.

Output convergence for both discretizations is plotted in Figure 6. The important takeaway from this case is that the DG and HDG results are nearly identical for a given mesh. In fact, the methods seem to target the same elements for refinement. So, under these conditions the optimal mesh seems to be the same. The number of nonzeros required to obtain a certain drag error in DG

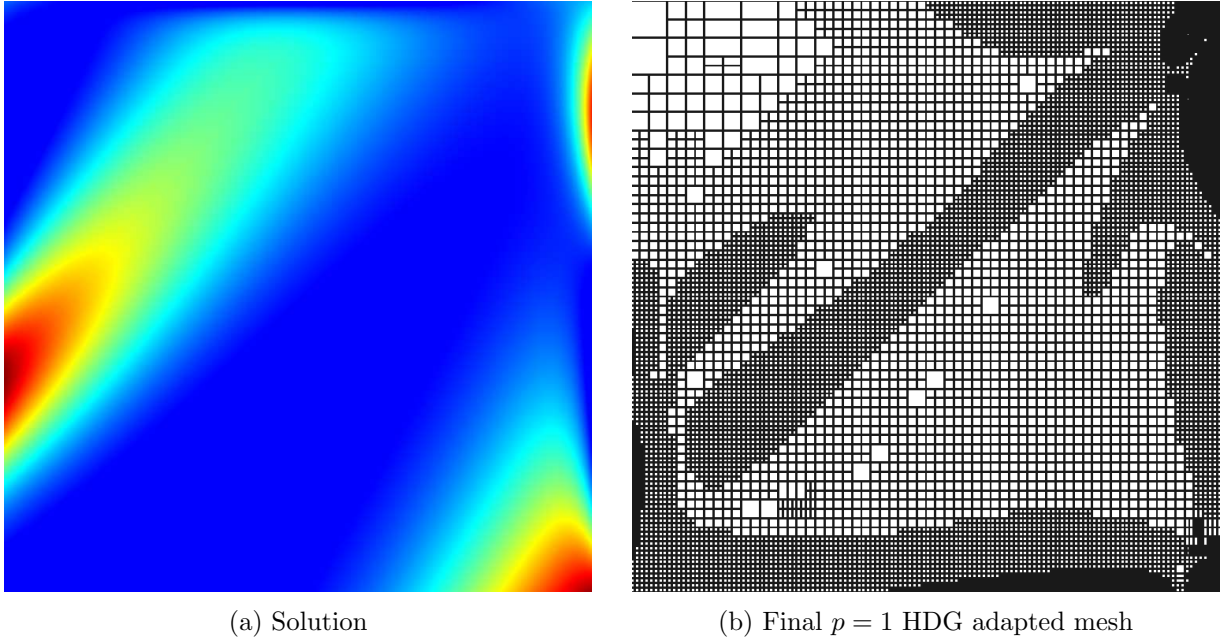


Figure 3: Solution to the convection-diffusion problem and adapted mesh for the case described in Sec. IV.A.

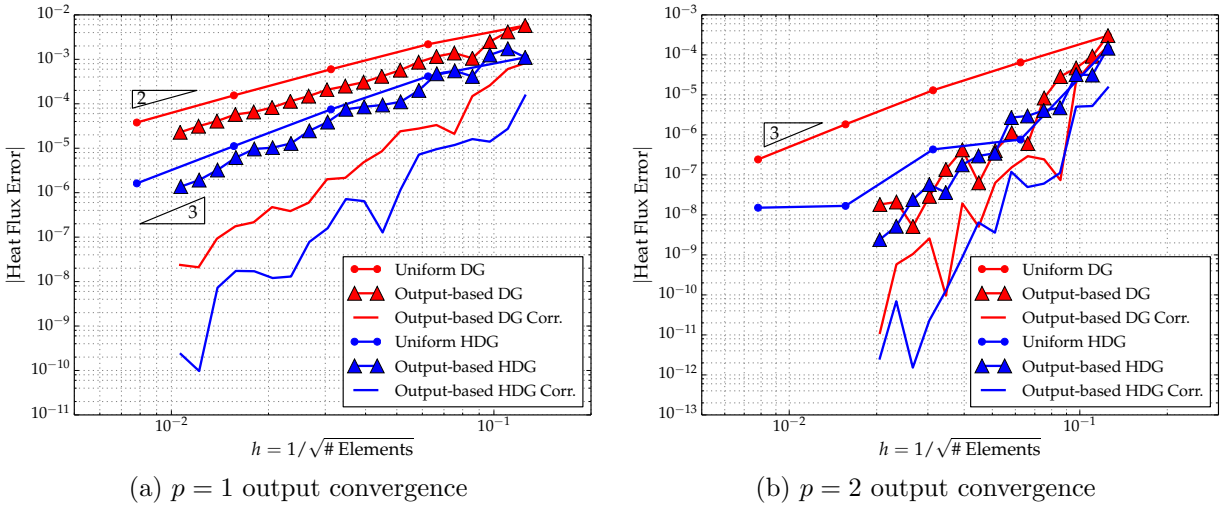


Figure 4: HDG and DG convergence results for the case described in Sec. IV.A. The  $p + 1$  fine space used in error estimation was solved exactly.

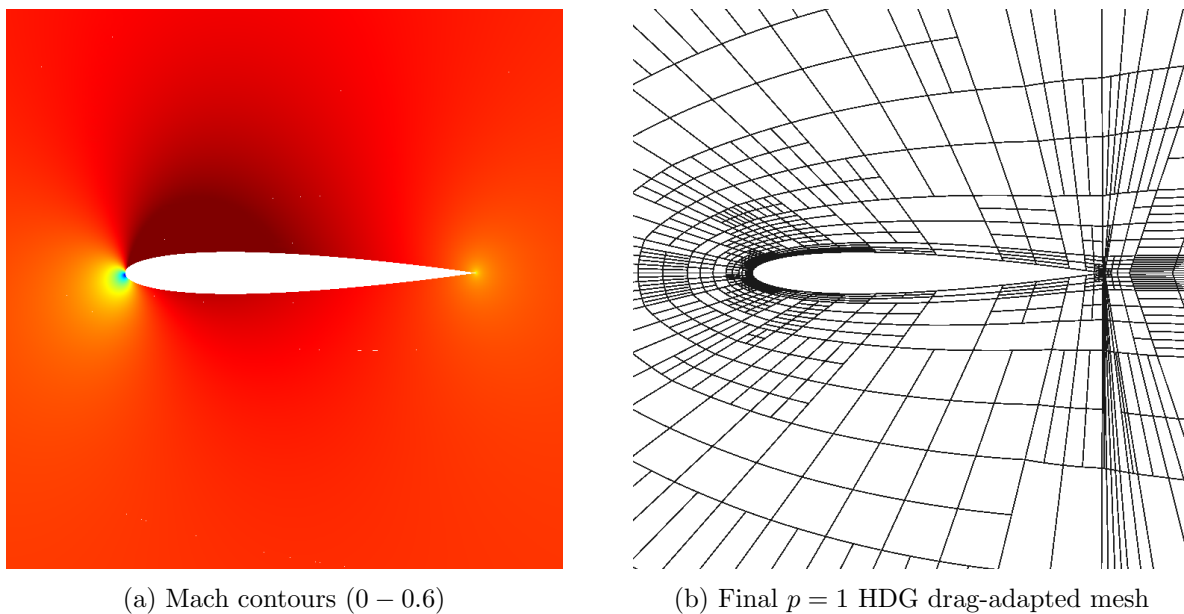


Figure 5: Mach contours and adapted mesh for the case described in Sec. IV.B.

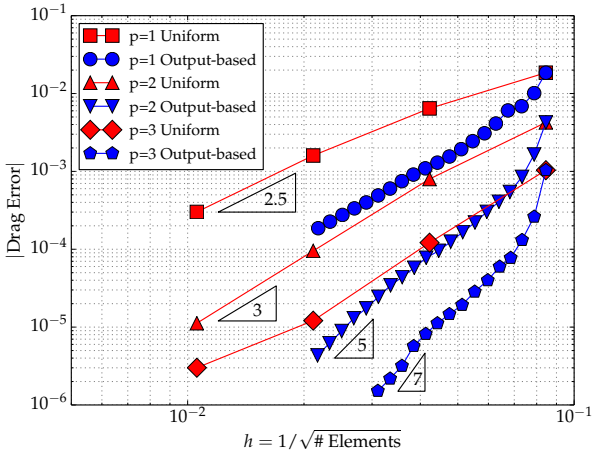
seem to follow a trend for any order. In HDG, however, as the order increases, the increase in the number of nonzeros is reduced, so a benefit at high order is achieved.

Recall that adaptation for HDG is ignoring the contribution from  $\delta J^\Lambda$ . It turns out that under these conditions,  $\delta J^\Lambda$  is driven to machine zero. The only terms appearing in the flux continuity equation, (10), are stabilization terms. If the face and both elements use degree  $p$  solution approximations, and the stabilization is constant, testing against degree  $p$  polynomials will always result in a pointwise zero integrand. Therefore, even on the fine space, these contributions are zero.

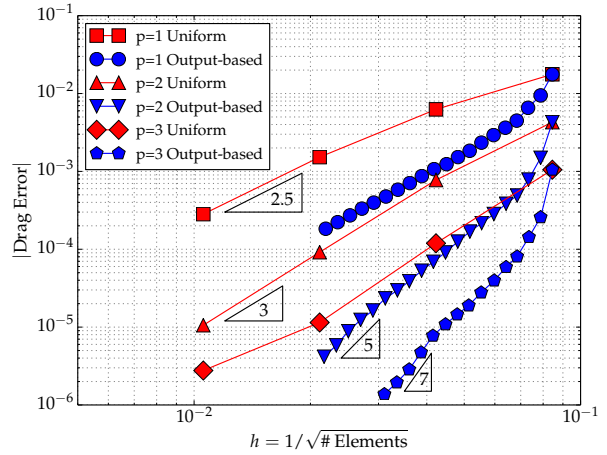
### C. Turbulent flow over RAE 2822 airfoil

For this test, we compare HDG output-based adaptation to DG for turbulent flow over an RAE 2822 airfoil. The conditions are  $M_\infty = 0.5$ ,  $\text{Re} = 10^5$ ,  $\alpha = 1^\circ$  angle of attack, and the turbulence is handled using the Spalart-Allmaras one-equation model for the Reynolds-averaged Navier-Stokes equations.<sup>22</sup> The output adaptation here is targeting the drag on the airfoil. The initial mesh for the adaptation consists of 374 quadrilateral elements with high-aspect ratio (but relatively coarse size) near the wall. At each adaptive iteration, 5% of the elements are isotropically refined, for a total of 8 adaptive iterations. The turbulence working variable  $\tilde{\nu}$  and adapted mesh are shown in Figure 7.

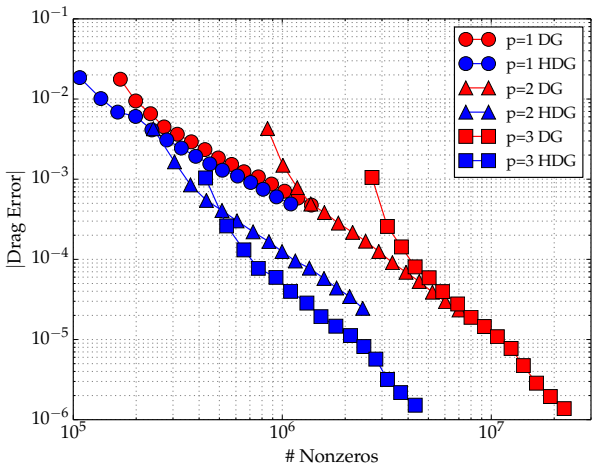
The convergence plotted in Figure 8 shows adaptive HDG working comparably to DG, but with many fewer nonzeros in the Jacobian matrix, especially at  $p = 3$ . The adapted meshes are quite similar, and we hypothesize that much of the difference in the outputs is due to differences in stabilization – on the highly-anisotropic elements in these meshes, a constant viscous length scale,  $\ell_{\text{visc}}$ , may not be optimal for HDG.



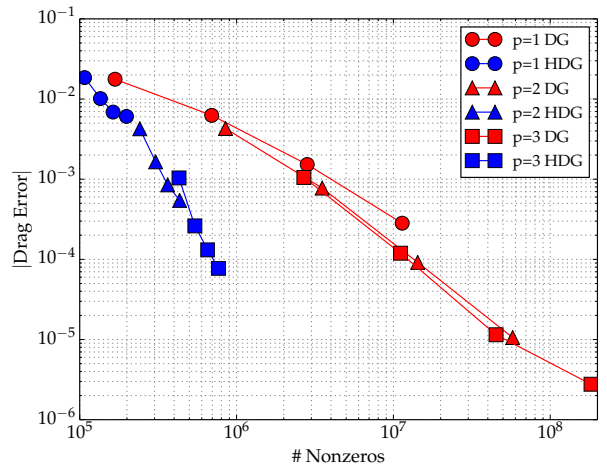
(a) HDG convergence



(b) DG convergence



(c) Adaptive refinement



(d) Uniform refinement

Figure 6: HDG and DG convergence results for the case described in Sec. IV.B. Here the  $p + 1$  fine space was only solved approximately, using 10 block-Jacobi smoothing iterations.

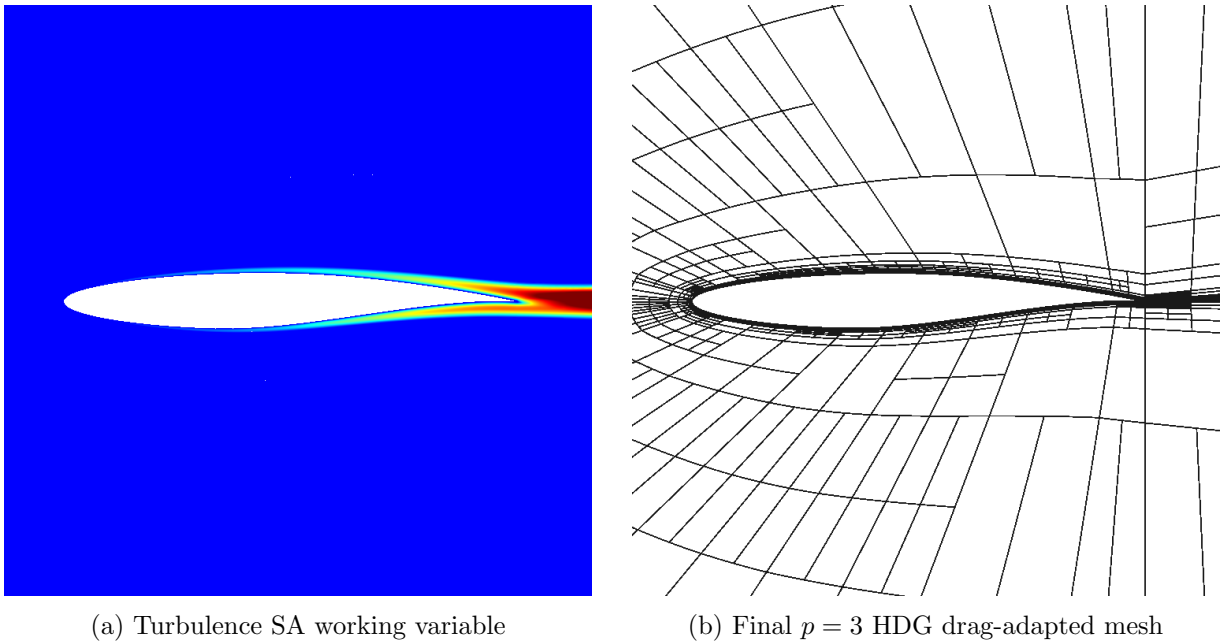


Figure 7: SA working variable  $\tilde{\nu}$  and final HDG drag-adapted mesh for the RAE2822 airfoil case described in Sec. IV.C.

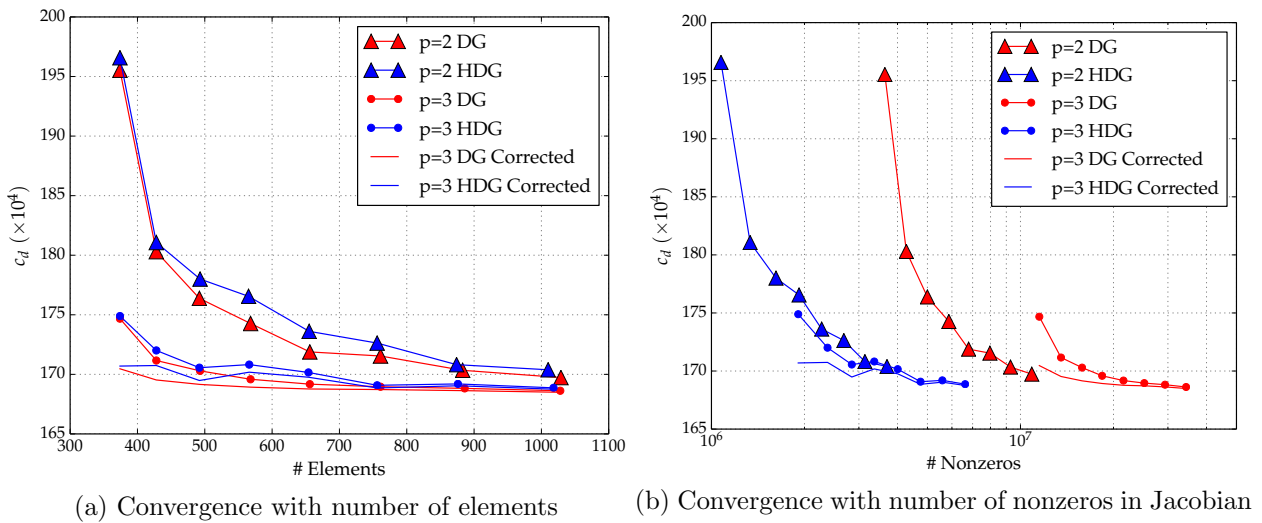
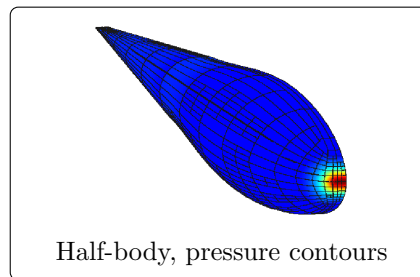


Figure 8: Convergence for the RAE2822 airfoil case described in Sec. IV.C.

## D. Three-Dimensional Body of Revolution

This case tests the output-based adaptation for the three-dimensional Euler equations to calculate lift on an immersed body. This test was used to assess the state of high-order CFD codes at the 1<sup>st</sup> high-order CFD workshop.<sup>23</sup> The flow is described by the inviscid Navier-Stokes equations at  $M = 0.5$  and  $\alpha = 1^\circ$  angle of attack. The immersed body is shown to the right with the HDG adapted boundary mesh. Initially, the mesh has 768  $q = 4$  hexahedral elements.



An interesting effect shows up in the adaptation for this case. Recall that in the inviscid NACA 0012 airfoil case shown previously, the convergence history between DG and HDG was nearly identical, owing to the machine-zero  $\delta J^A$  on the faces. In this case, although still inviscid, the adaptation is targeting different elements. In fact, consider the two meshes at the same iteration shown in Figure 9: most of the elements targeted for the next refinement in HDG are already refined in DG. Further investigation is required to determine the source of these differences.

The convergence shown in Figure 10 is similar for both methods. HDG uses approximately a factor of 8 fewer nonzeros for a nearly exact value of lift. It is interesting to note that here the error estimates for HDG seem to decrease slower than those in DG, but are still quite accurate.

## V. Conclusion and Future Work

In this paper we presented output-based error estimation and adaptation for an HDG method applied to the compressible Navier-Stokes equations, including those augmented with a Reynolds-averaged turbulence model. The HDG method takes advantage of a decoupled Jacobian matrix structure to statically condense the system, thereby reducing the growth rate with order of the globally-coupled system and the number of nonzeros in the resulting global Jacobian matrix. Some practical details of the HDG discretization were presented, including parallelization on distributed-memory architectures. A new viscous stabilization method was introduced, which applies diffusive stabilization to only those equations that require it.

An  $h$ -adaptation method for HDG was discussed and the HDG method was compared to DG in an existing solver framework for various problems of aerospace engineering interest. A computational benefit both in terms of memory was observed for all cases. It was shown that under certain conditions the adaptive convergence is nearly identical for both methods.

Looking forward, the most pressing item is to improve viscous stabilization. Even when adding only viscous stabilization to those equations that require it, a constant viscous length scale seems to add stiffness. The largest benefits will likely come from combined  $hp$  adaptation, which is forthcoming. Mechanics for addition of the face error indicators, and under what conditions to choose  $h$  or  $p$  are slightly more difficult than in DG, so care must be taken to pick the correct strategy.

## Acknowledgments

The authors acknowledge support of the University of Michigan and the National Science Foundation Graduate Research Fellowship.

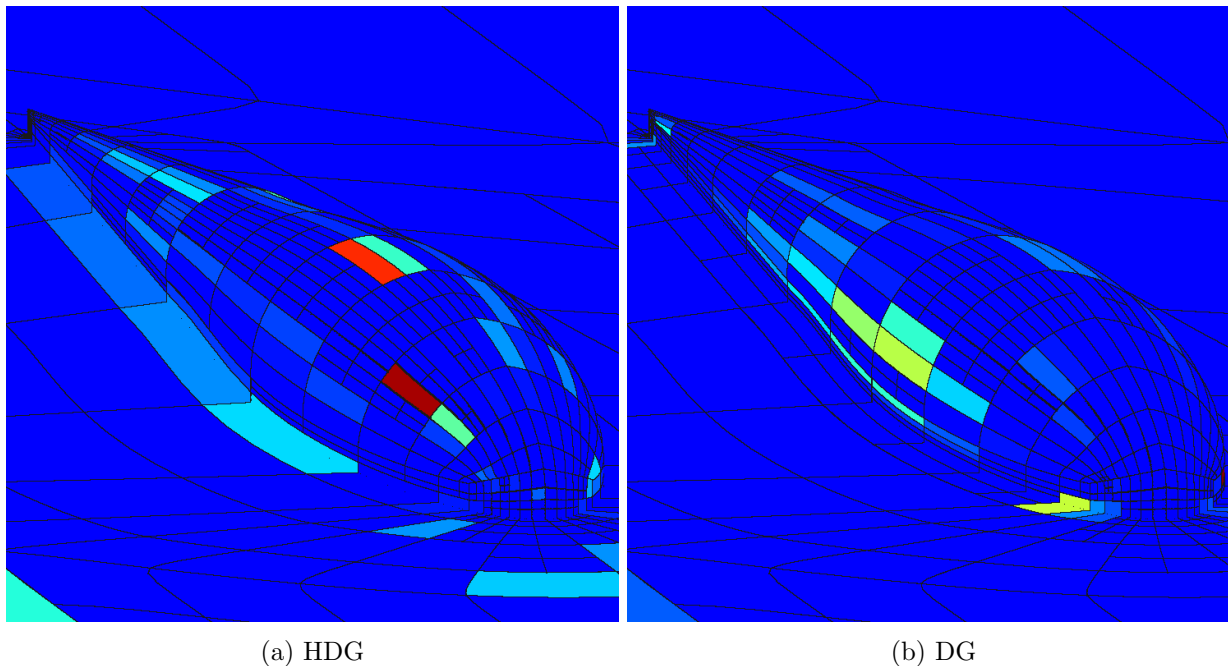


Figure 9: HDG and DG adapted meshes projected on the half-body at the same iteration colored by the error indicator for the case described in Sec. IV.D. The meshes are very similar, and interestingly many of the elements targeted for the next refinement in HDG are already refined with DG.

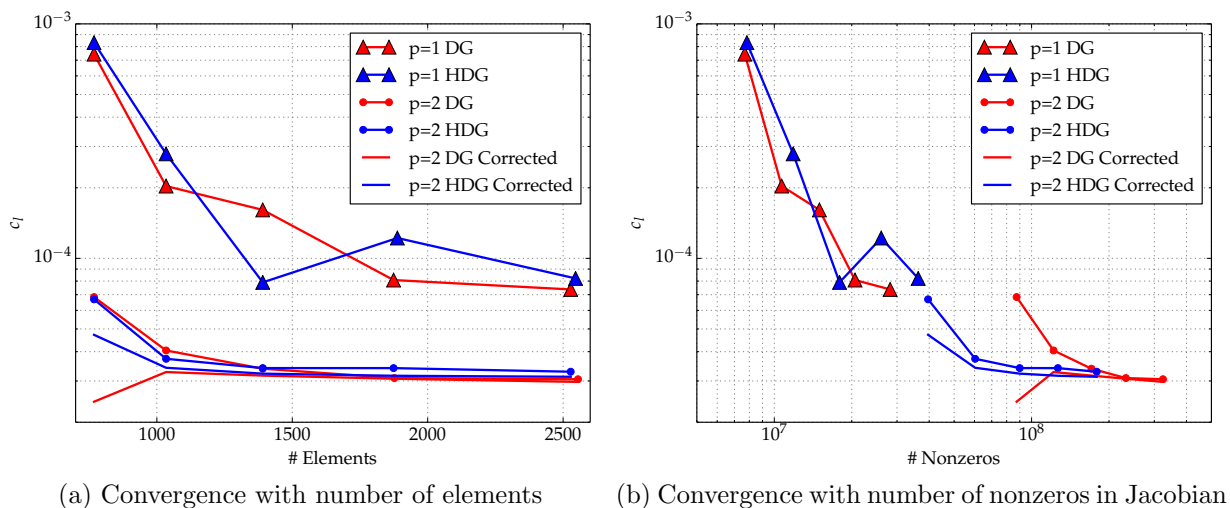


Figure 10: Convergence for the case described in Sec. IV.D.



## References

- <sup>1</sup>Peraire, J., Nguyen, N., and Cockburn, B., “A hybridizable discontinuous Galerkin method for the compressible Euler and Navier-Stokes equations,” AIAA Paper 2010-363, 2010.
- <sup>2</sup>Brezzi, F., Douglas Jr, J., and Marini, L. D., “Two families of mixed finite elements for second order elliptic problems,” *Numerische Mathematik*, Vol. 47, No. 2, 1985, pp. 217–235.
- <sup>3</sup>Brezzi, F. and Fortin, M., *Mixed and hybrid finite element methods*, Springer-Verlag New York, Inc., 1991.
- <sup>4</sup>Celiker, F. and Cockburn, B., “Superconvergence of the numerical traces of discontinuous Galerkin and hybridized methods for convection-diffusion problems in one space dimension,” *Mathematics of Computation*, Vol. 76, No. 257, 2007, pp. 67–96.
- <sup>5</sup>Cockburn, B., Dong, B., Guzmán, J., Restelli, M., and Sacco, R., “A hybridizable discontinuous Galerkin method for steady-state convection-diffusion-reaction problems,” *SIAM Journal on Scientific Computing*, Vol. 31, No. 5, 2009, pp. 3827–3846.
- <sup>6</sup>Nguyen, N. C., Peraire, J., and Cockburn, B., “An implicit high-order hybridizable discontinuous Galerkin method for linear convection–diffusion equations,” *Journal of Computational Physics*, Vol. 228, No. 9, 2009, pp. 3232–3254.
- <sup>7</sup>Schütz, J., Woopen, M., and May, G., “A hybridized DG/mixed scheme for nonlinear advection-diffusion systems, including the compressible Navier-Stokes equations,” AIAA Paper 2012-729, 2012.
- <sup>8</sup>Balan, A., Woopen, M., and May, G., “Adjoint-based hp-adaptation for a class of high-order hybridized finite element schemes for compressible flows,” AIAA Paper 2013-2938, 2013.
- <sup>9</sup>Bassi, F. and Rebay, S., “GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations,” *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by K. Cockburn and Shu, Springer, Berlin, 2000, pp. 197–208.
- <sup>10</sup>Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., “ $p$ -Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations,” *Journal of Computational Physics*, Vol. 207, 2005, pp. 92–113.
- <sup>11</sup>Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D., “Unified analysis of discontinuous Galerkin methods for elliptic problems,” *SIAM journal on numerical analysis*, Vol. 39, No. 5, 2002, pp. 1749–1779.
- <sup>12</sup>Roe, P. L., “Approximate Riemann solvers, parameter vectors, and difference schemes,” *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.
- <sup>13</sup>Kirby, R. M., Sherwin, S. J., and Cockburn, B., “To CG or to HDG: a comparative study,” *Journal of Scientific Computing*, Vol. 51, No. 1, 2012, pp. 183–212.
- <sup>14</sup>Nguyen, N. C., Peraire, J., and Cockburn, B., “An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection–diffusion equations,” *Journal of Computational Physics*, Vol. 228, No. 23, 2009, pp. 8841–8855.
- <sup>15</sup>Nguyen, N., Peraire, J., and Cockburn, B., “Hybridizable discontinuous Galerkin methods,” *Spectral and High Order Methods for Partial Differential Equations*, Springer, 2011, pp. 63–84.
- <sup>16</sup>Ceze, M. and Fidkowski, K. J., “A robust adaptive solution strategy for high-order implicit CFD solvers,” , No. 2011-3696, 2011.
- <sup>17</sup>Fidkowski, K. J., *A High-order Discontinuous Galerkin Multigrid Solver for Aerodynamic Applications*, MS thesis, M.I.T., Department of Aeronautics and Astronautics, June 2004.
- <sup>18</sup>Persson, P.-O. and Peraire, J., “Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations,” *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 2709–2733.
- <sup>19</sup>Diosady, L. and Darmofal, D., “Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations,” *Journal of Computational Physics*, Vol. 228, 2009, pp. 3917–3935.
- <sup>20</sup>Karypis, G. and Kumar, V., “Parallel multilevel k-way partitioning scheme for irregular graphs,” *SIAM Review*, Vol. 41, No. 2, 1999, pp. 278–300.
- <sup>21</sup>Fidkowski, K. J. and Darmofal, D. L., “Review of output-based error estimation and mesh adaptation in computational fluid dynamics,” *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694.
- <sup>22</sup>Oliver, T. A., *A High-order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-Averaged Navier-Stokes Equations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.
- <sup>23</sup>Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., et al., “High-order CFD methods: current status and perspective,” *International Journal for Numerical Methods in Fluids*, 2013.