A Hybridized Discontinuous Galerkin Method on Mapped Deforming Domains

Krzysztof Fidkowski*

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

Abstract

In this paper we present a hybridized discontinuous Galerkin (HDG) discretization for unsteady simulations of convection-dominated flows on mapped deforming domains. Mesh deformation is achieved through an arbitrary Lagrangian-Eulerian transformation with an analytical mapping. We present details of this transformation applied to the HDG system of equations, with focus on the auxiliary gradient equation, viscous stabilization, and output calculation. We discuss conditions under which optimal unsteady output convergence rates can be attained, and we show that both HDG and discontinuous Galerkin (DG) achieve these rates in advection-dominated flows. Results for scalar advection-diffusion and the Euler equations verify the implementation of the mesh motion mapping for both discretizations, show that HDG and DG yield similar results on a given mesh, and demonstrate differences in output convergence rates depending on the choice of HDG viscous stabilization. We note that such similar results bode well for HDG, which has fewer globally-coupled degrees of freedom compared to DG. A simulation of the unsteady compressible Navier-Stokes equations demonstrates again very similar results for HDG and DG and illustrates a pitfall of using steady-state adapted meshes for accurate unsteady simulations.

Keywords: Hybridized Discontinuous Galerkin, High Order Finite Elements, Deforming Domains, Mesh Motion

*Corresponding author

Email address: kfid@umich.edu, (tel) 734-615-7247, (fax) 734-763-0578 (Krzysztof Fidkowski)

1. Introduction

High-order accurate methods in Computational Fluid Dynamics can in many cases provide superior accuracy compared to lower order methods, but this accuracy improvement typically comes at the cost of increased computational expense. Details of the trade-off are debatable and depend on the particular high and low-order methods compared, and on their implementation. Nonetheless, a general observation is that high-order methods could benefit from becoming cheaper. In this work, we consider one such method, hybridized discontinuous Galerkin (HDG) [1, 2, 3, 4, 5], which under certain circumstances, notably at high order, can be computationally cheaper than the popular discontinuous Galerkin (DG) method [6, 7, 8, 9, 10, 11, 12].

A relatively fair way to compare the DG and HDG methods is in an output-based adaptive setting, where the mesh resolution is automatically dictated by an error estimate of an output of interest. Much work has been done in this area for steady problems with finite volume and finite element methods [13, 14, 15, 16, 17, 18, 19], and recently with HDG discretizations [20, 21, 22]. Unsteady problems pose additional challenges and computational costs, namely in the unsteady adjoint solution, yet output-based adaptive methods have also been explored, with various modes of adaptation, including static-mesh, dynamic-mesh, space-only, and combined space-time [23, 24, 25, 26, 27, 28, 29, 30, 31].

However, the first step in a comparing DG and HDG for problems on deformable domains is the formulation and verification of the mesh deformation mapping via the Arbitrary Lagrangian-Eulerian formulation. This has already been done for DG [32], including in an adaptive setting [31], but not yet for HDG, and hence this is the subject of the present paper. We note that a space-time HDG formulation that supports mesh motion has already been presented [33]. In this work, we extend unsteady the ALE formulation to HDG, paying particular attention to the gradient equation and viscous stabilization. We discuss implementation details and then compare HDG and DG for

both verification problems and for a simulation of the compressible Navier-Stokes equations. A simple adaptive strategy in which steady-state optimized meshes are used for the unsteady simulation is shown to perform poorly, motivating the need for full unsteady adaptation.

2. Discretization

For the spatial discretization, we compare the discontinuous Galerkin (DG) and hybridized discontinuous Galerkin (HDG) methods. Figure 1 illustrates the primary differences between these discretizations, namely the introduction of additional degrees of freedom that decouple elements from each other and yield a global system of smaller size at high order compared to DG.

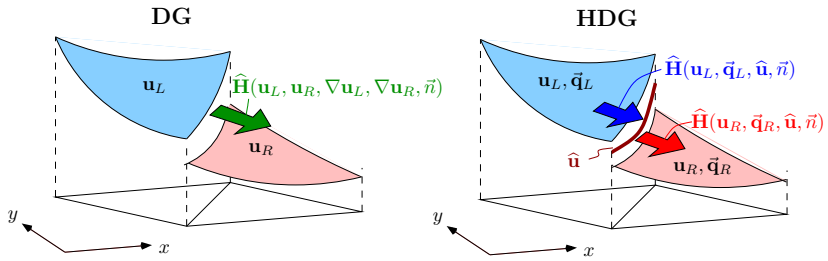


Figure 1: In the HDG method, additional unknowns on element interfaces allow elimination of the element-interior unknowns. This results in a global system in which the number of unknowns scales as $p^{\text{dim}-1}$ instead of p^{dim} for DG.

2.1. Discontinuous Galerkin

Consider a conservation law of the form,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \nabla\mathbf{u}) \quad = \quad \mathbf{0}, \quad \mathbf{u}(\vec{x}, 0) = \mathbf{u}_0(\vec{x}), \quad (\vec{x}, t) \in \Omega \otimes [0, T], \tag{1}$$

where $\mathbf{u}(\vec{x}, t) \in \mathbb{R}^s$ is the state vector, $\mathbf{u}_0(\vec{x})$ is the initial condition, and $\vec{\mathbf{H}} = \vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \nabla\mathbf{u})$ is the total flux consisting of inviscid ($\vec{\mathbf{F}}$) and viscous ($\vec{\mathbf{G}}$) contributions. Boundary conditions are specified on $\partial\Omega$ in the form of knowledge of components of the state

or the flux. The i^{th} spatial component of the viscous flux is linear in $\nabla\mathbf{u}$, $\mathbf{G}_i = -\mathbf{K}_{ij}\partial\mathbf{u}_j$ (summation implied on $j \in 1\ldots\dim$), where $\mathbf{K}_{ij}$ is the (i,j) component of the viscous diffusivity tensor.

Denote by T_h the set of N_e elements in a non-overlapping tessellation of the domain Ω. In DG, the state is approximated by polynomials on each element, with no continuity constraints imposed on the approximations on adjacent elements. Formally, we write that $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where $\mathcal{V}_h = \{u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \ \forall \Omega_e \in T_h\}$, and $\mathcal{P}^p$ denotes polynomials of order p on an element Ω_e.

We obtain the weak form of (1) by multiplying the PDE by test functions $\mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h$ and integrating by parts to couple elements via fluxes. The resulting weak statement reads: find $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h$ such that

$$\begin{aligned}\int_{\Omega_e} \mathbf{w}_h^T \frac{\partial \mathbf{u}_h}{\partial t}\, d\Omega - \int_{\Omega_e} \nabla\mathbf{w}_h^T \cdot \vec{\mathbf{H}}\, d\Omega + \int_{\partial\Omega_e} \mathbf{w}_h^{+T} \widehat{\mathbf{H}} \cdot \vec{n}\, ds \\ - \int_{\partial\Omega_e} \nabla\mathbf{w}_h^T \cdot \widehat{\mathbf{K}} \cdot \vec{n}(\mathbf{u}_h - \hat{\mathbf{u}}_h)\, ds \;=\; 0 \qquad \forall \mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h, \quad (2)\end{aligned}$$

where $(\cdot)^T$ denotes transpose, and on the element boundary $\partial\Omega_e$ the notations $(\cdot)^+, (\cdot)^-, (\cdot)^b$ respectively denote quantities taken from the element interior, neighbor element, and boundary. The last term on the left hand side symmetrizes the semilinear form for adjoint consistency [34]. $\widehat{\mathbf{H}} = \widehat{\mathbf{F}} + \widehat{\mathbf{G}}$ is the numerical flux, consisting of Roe's approximate Riemann solver [35] for $\widehat{\mathbf{F}}$, and the second form of Bassi and Rebay (BR2) [34] for the viscous flux, $\widehat{\mathbf{G}}$. In particular, on an interior face σ^f, the BR2-stabilized viscous flux is

$$\widehat{\mathbf{G}} = \frac{1}{2}\left(\mathbf{G}_i^+ + \mathbf{G}_i^-\right) n_i^+ + \eta \frac{1}{2}\left(\boldsymbol{\delta}_i^+ + \boldsymbol{\delta}_i^-\right) n_i^+, \quad (3)$$

where the auxiliary variables $\boldsymbol{\delta}_i^+, \boldsymbol{\delta}_i^- \in [\boldsymbol{\mathcal{V}}_h]^d$ have support on the two elements adjacent to the interior face σ^f and are obtained by solving $\forall \boldsymbol{\tau}_{hi} \in \boldsymbol{\mathcal{V}}_h$

$$\int_{\Omega_e^+} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i^+\, d\Omega + \int_{\Omega_e^-} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i^-\, d\Omega \;=\; \int_{\sigma^f} \frac{1}{2}\left(\boldsymbol{\tau}_{hi}^{+T} \boldsymbol{K}_{ij}^+ + \boldsymbol{\tau}_{hi}^{-T} \boldsymbol{K}_{ij}^-\right)(\mathbf{u}_h^+ - \mathbf{u}_h^-) n_j^+\, ds. \quad (4)$$

η is a stabilization factor that should not be less than the number of faces per element, and in our work is taken to be (at least) twice the maximum number of faces on adjacent elements.

$\widehat{\mathbf{K}}$ is the viscous diffusivity tensor computed from the element-interior state, $\mathbf{u}_h^+$, for interior faces and from the boundary state, $\mathbf{u}_h^b$, for boundary faces. This boundary state is a function (projection) of the interior state and the boundary-condition data, $\mathbf{u}_h^b = \mathbf{u}_h^b(\mathbf{u}_h^+, \text{BC})$. The unique state on an interior face is $\widehat{\mathbf{u}}_h = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$, whereas for boundary states it is $\widehat{\mathbf{u}}_h = \mathbf{u}_h^b$.

On a boundary face σ^b, fluxes are typically computed directly from the boundary state, $\mathbf{u}^b$. One exception is the convective flux for a boundary condition in which a complete exterior state is specified: in such a case, an approximate-Riemann solver is used to compute the boundary convective flux, $\widehat{\mathbf{F}}^b$. The BR2-stabilized boundary viscous flux is

$$\widehat{\mathbf{G}}^b = \Pi_G^{\text{BC}} \left[\vec{\mathbf{G}}(\mathbf{u}_h^b, \nabla \mathbf{u}_h^+) \cdot \vec{n} + \eta \vec{\boldsymbol{\delta}} \cdot \vec{n} \right], \tag{5}$$

where η is a stabilization parameter typically set to the number of faces of the boundary element, and the auxiliary BR2 variable $\vec{\boldsymbol{\delta}} \in [\mathcal{V}_h]^{\text{dim}}$ has support on the element adjacent to the boundary face σ^b and is obtained by solving

$$\int_{\Omega_e} \mathbf{v}_{hi}^T \boldsymbol{\delta}_i \, d\Omega \quad = \quad \int_{\sigma^b} \mathbf{v}_{hi}^{+T} \boldsymbol{K}_{ij}^b (\mathbf{u}_h^+ - \mathbf{u}_h^b) n_j^+ \, ds \qquad \forall \vec{\mathbf{v}}_h \in [\mathcal{V}_h]^{\text{dim}}. \tag{6}$$

The projection Π_G^{BC} in (5) incorporates boundary conditions on the viscous flux, such as a prescribed heat flux in the compressible Navier-Stokes equations. For example, in the case of an adiabatic wall, Π_G^{BC} would zero out both the mass component (no flow through the wall) and the energy component (no heat transfer through the wall) of the flux.

In steady state, choosing a basis for the trial/test spaces turns Eqn. 2 into a nonlinear

system of equations, $\mathbf{R}(\mathbf{U}) = \mathbf{0}$, where $\mathbf{U}$ and $\mathbf{R}$ are the discrete state and residual vectors, respectively. We solve this system using a preconditioned Newton-GMRES method, with full Jacobian storage.

2.2. Hybridized Discontinuous Galerkin

We can write (1) as a first-order system by introducing the state gradient, $\vec{\mathbf{q}} \in [\mathbb{R}^s]^{\text{dim}}$, as a new variable,

$$\vec{\mathbf{q}} - \nabla \mathbf{u} = \vec{\mathbf{0}}, \tag{7}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \vec{\mathbf{q}}) = \mathbf{0}. \tag{8}$$

In HDG, we approximate $\mathbf{u}$ by $\mathbf{u}_h \in \mathcal{V}_h$, the same space as in DG. The state gradient, $\vec{\mathbf{q}}$, is an additional unknown that we represent by $\vec{\mathbf{q}}_h \in [\mathcal{V}_h]^{\text{dim}}$. Finally, HDG introduces a new unknown, $\hat{\mathbf{u}}$, the state on the faces between elements, which is approximated by $\hat{\mathbf{u}}_h \in \mathcal{M}_h$, where $\mathcal{M}_h = [\mathcal{M}_h]^s$, and $\mathcal{M}_h = \left\{\lambda \in L_2(\mathcal{E}_h) : \lambda|_{\sigma_f} \in \mathcal{P}^p(\sigma_f)\ \forall \sigma_f \in \mathcal{E}_h\right\}$. $\mathcal{E}_h$ is the set of *interior* faces, σ_f, in the tessellation, and $\mathcal{P}^p(\sigma_f)$ is the space of polynomials of order p on face σ_f. Note that we do not define or use $\hat{\mathbf{u}}$ on domain boundary faces, where we instead use the boundary state $\mathbf{u}^b$. We obtain the weak form of the PDE system by weighting the above equations with appropriate test functions, integrating by parts, and using the interface variable $\hat{\mathbf{u}}$ for the face state,

$$\int_{\Omega_e} \vec{\mathbf{v}}_h^T \cdot \vec{\mathbf{q}}_h \, d\Omega + \int_{\Omega_e} \nabla \cdot \vec{\mathbf{v}}_h^T \mathbf{u}_h \, d\Omega - \int_{\partial\Omega_e} \vec{\mathbf{v}}_h^T \cdot \vec{n}\, \widehat{\mathbf{u}}_h \, ds = 0 \qquad \forall \vec{\mathbf{v}}_h \in [\mathcal{V}_h]^{\text{dim}}, \tag{9}$$

$$\int_{\Omega_e} \mathbf{w}_h^T \frac{\partial \mathbf{u}_h}{\partial t} \, d\Omega - \int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \vec{\mathbf{H}} \, d\Omega + \int_{\partial\Omega_e} \mathbf{w}_h^T \widehat{\mathbf{H}} \cdot \vec{n} \, ds = 0 \qquad \forall \mathbf{w}_h \in \mathcal{V}_h, \tag{10}$$

$$\int_{\sigma_f} \boldsymbol{\mu}_h^T \left\{ \widehat{\mathbf{H}} \cdot \vec{n}\big|_L + \widehat{\mathbf{H}} \cdot \vec{n}\big|_R \right\} ds = 0 \qquad \forall \boldsymbol{\mu}_h \in \mathcal{M}_h, \tag{11}$$

The third equation weakly imposes the continuity of the flux across interfaces and is required to close the system. Note that the $\boldsymbol{\mu}_h$ are test functions taken from the approximation space of $\widehat{\mathbf{u}}$, $\mathcal{M}_h$. As constants are part of this space, conservation is guaranteed.

The "one-sided" fluxes are defined as

$$\widehat{\mathbf{H}} \cdot \vec{n} \;=\; \vec{\mathbf{H}}(\hat{\mathbf{u}}_h, \vec{\mathbf{q}}_h) \cdot \vec{n} + \boldsymbol{\tau}(\hat{\mathbf{u}}_h, \mathbf{u}_h, \vec{n}), \tag{12}$$

where $\boldsymbol{\tau} = \boldsymbol{\tau}_F + \boldsymbol{\tau}_G$ is a stabilization flux consisting of convective ($\boldsymbol{\tau}_F$) and viscous ($\boldsymbol{\tau}_G$) contributions. $\boldsymbol{\tau}_F$ is chosen to yield a Roe-like flux,

$$\boldsymbol{\tau}_F = \left| \frac{\partial}{\partial \mathbf{u}_h} (\vec{\mathbf{F}}(\hat{\mathbf{u}}_h) \cdot \vec{n}) \right| (\mathbf{u}_h - \hat{\mathbf{u}}_h). \tag{13}$$

$\boldsymbol{\tau}_G$ is chosen to penalize jumps in the state, and two forms of $\boldsymbol{\tau}_G$ are considered,

$$\boldsymbol{\tau}_G = \frac{1}{\ell_{\text{visc}}} n_i \mathbf{K}_{ij} n_j (\mathbf{u}_h - \hat{\mathbf{u}}_h) \quad \text{or} \quad \boldsymbol{\tau}_G = \eta \vec{\boldsymbol{\delta}} \cdot \vec{n}, \tag{14}$$

where ℓ_{visc} is an $O(1)$ user-defined viscous length scale, and $\eta\vec{\boldsymbol{\delta}}$ is the BR2 stabilization term that lifts the jump $\mathbf{u}_h - \hat{\mathbf{u}}_h$ from faces to element interiors. The BR2 stabilization results in an $O(1/h)$ viscous length scale, which is not theoretically optimal for convergence [36] but which can improve solver robustness.

In steady-state, choosing bases for the trial/test spaces turns Eqns. 9–11 into a nonlinear system of equations, $\vec{\mathbf{R}}^Q = \vec{\mathbf{0}}$, $\mathbf{R}^U = \mathbf{0}$, $\mathbf{R}^\Lambda = \mathbf{0}$, with the Newton update system

$$\begin{bmatrix} \mathbf{A}^{QQ} & \mathbf{A}^{QU} & \mathbf{B}^Q \\ \mathbf{A}^{UQ} & \mathbf{A}^{UU} & \mathbf{B}^U \\ \mathbf{C}^Q & \mathbf{C}^U & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta\vec{\mathbf{Q}} \\ \Delta\mathbf{U} \\ \Delta\boldsymbol{\Lambda} \end{bmatrix} + \begin{bmatrix} \vec{\mathbf{R}}^Q \\ \mathbf{R}^U \\ \mathbf{R}^\Lambda \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{0}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{15}$$

where $\vec{\mathbf{Q}}$, $\mathbf{U}$, and $\boldsymbol{\Lambda}$ are the discrete unknowns in the approximation of $\vec{\mathbf{q}}_h$, $\mathbf{u}_h$, and $\hat{\mathbf{u}}_h$, respectively; $\vec{\mathbf{R}}^Q$, $\mathbf{R}^U$, and $\mathbf{R}^\Lambda$ are the discrete residual vectors; and $[\mathbf{A}, \mathbf{B}; \mathbf{C}, \mathbf{D}]$ is the primal Jacobian matrix partitioned into element-interior and interface unknowns. Note that (15) shows the Q and U block components of matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. Statically

condensing out the element-interior states gives a smaller system,

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

where $\mathcal{K}$ is a sparse, compact-stencil, condensed Jacobian matrix for the face degrees of freedom.

Boundary conditions for HDG are incorporated in the same fashion as in DG: we define a boundary flux using the element-interior state, the outward-pointing normal, the element-interior $\vec{\mathbf{q}}$, and boundary condition data,

$$\widehat{\mathbf{H}} \cdot \vec{n}\big|_{\text{boundary}} \quad = \quad \widehat{\mathbf{H}}^{b}(\mathbf{u}, \vec{\mathbf{q}}_h, \vec{n}, \text{BCs}) + \boldsymbol{\tau}(\mathbf{u}_h^{b}, \mathbf{u}_h^{+}, \vec{n}). \tag{16}$$

We emphasize that there is no separate face-based state defined on the boundary faces. The boundary condition comes in through a flux that is constructed directly using the element interior state and boundary condition data. As in DG, stabilization is included in the boundary flux to penalize jumps between the element-interior state, $\mathbf{u}_h^{+}$, and the boundary state, $\mathbf{u}_h^{b}$.

2.3. Unsteady

The above discretizations extend to implicit unsteady formulations. For DG, the unsteady discrete system is

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R} \quad = \quad \mathbf{0}, \tag{17}$$

where $\mathbf{M}$ is the spatial mass matrix. In HDG, the unsteady term appears only in (10), so that the unsteady discrete system is

$$\begin{aligned} \vec{\mathbf{R}}^{Q} &= \vec{\mathbf{0}}, \\ \mathbf{M}^{U}\frac{d\mathbf{U}}{dt} + \mathbf{R}^{U} &= \mathbf{0}, \\ \mathbf{R}^{\Lambda} &= \mathbf{0}, \end{aligned} \tag{18}$$

where $\mathbf{M}^{U}$ is the spatial mass matrix built using the basis functions for $\mathbf{U}$ – this is an invertible matrix. Define the solution vector $\mathbf{W} \equiv [\vec{\mathbf{Q}}; \mathbf{U}; \mathbf{\Lambda}]$, and the agglomerated residual $\mathbf{R} \equiv [\vec{\mathbf{R}}^{Q}; \mathbf{R}^{U}; \mathbf{R}^{\Lambda}]$, so that (18) can be written compactly as

$$\mathbf{M}\frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \tag{19}$$

where the general, non-invertible, mass matrix is

$$\mathbf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^{U} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Note, for DG, $\mathbf{W} = \mathbf{U}$ and $\mathbf{M} = \mathbf{M}^{U}$.

2.3.1. Diagonally-Implicit Runge Kutta

When time-marching (19) using an n_{stage} DIRK scheme, each time step requires n_{stage} nonlinear solves. The algorithm for advancing from $\mathbf{W}^0$ at t^0 to $\mathbf{W}^1$ at t^1 in a time

step of size Δt proceeds as follows:

$$
\begin{aligned}
&\text{for} \quad i = 1 : n_{\text{stage}} \\
&\qquad \mathbf{S}_i = -\frac{\mathbf{M}}{\Delta t}\mathbf{W}^0 + \sum_{j=1}^{i-1} a_{ij}\mathbf{R}(\mathbf{W}_j, t_j) \\
&\qquad \text{solve: } \frac{\mathbf{M}}{\Delta t}\mathbf{W}_i + a_{ii}\mathbf{R}(\mathbf{W}_i, t_i) + \mathbf{Z}^U\mathbf{S}_i = \mathbf{0} \\
&\text{end}
\end{aligned}
$$

where $t_i = t^0 + b_i\Delta t$, and $\mathbf{Z}^U$ is a mask matrix that zeros out all components of a vector except for those associated with the $\mathbf{U}$ unknowns – this is just the identity for DG. At the end of these stages, $\mathbf{W}^1$ is set to $\mathbf{W}_{n_{\text{stage}}}$. In this work we use a five-stage, fourth-order accurate scheme, for which

$$
a_{ij} = \begin{bmatrix}
\frac{1}{4} & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 \\
\frac{17}{50} & -\frac{1}{25} & \frac{1}{4} & 0 & 0 \\
\frac{371}{1360} & -\frac{137}{2720} & \frac{15}{544} & \frac{1}{4} & 0 \\
\frac{25}{24} & -\frac{49}{48} & \frac{125}{16} & -\frac{85}{12} & \frac{1}{4}
\end{bmatrix}, \quad
b_i = \begin{bmatrix}
\frac{1}{4} \\ \frac{3}{4} \\ \frac{11}{20} \\ \frac{1}{2} \\ 1
\end{bmatrix}.
$$

3. Arbitrary Lagrangian-Eulerian Formulation

3.1. Mapping

In an arbitrary Lagrangian-Eulerian (ALE) method, the mesh can move at a velocity different from that of the flow, which is useful for modeling problems in which objects move or deform. The ALE method uses a map between the deforming physical domain and a static reference domain and solves transformed equations on the reference domain [32]. This transformation is illustrated graphically in Figure 2, and Table 1 defines key quantities.

The expressions for the transformations of the normals are obtained using $dv = gdV$ for infinitesimal volumes and $d\vec{l} = \mathcal{G}d\vec{L}$ for infinitesimal vectors. In this work the

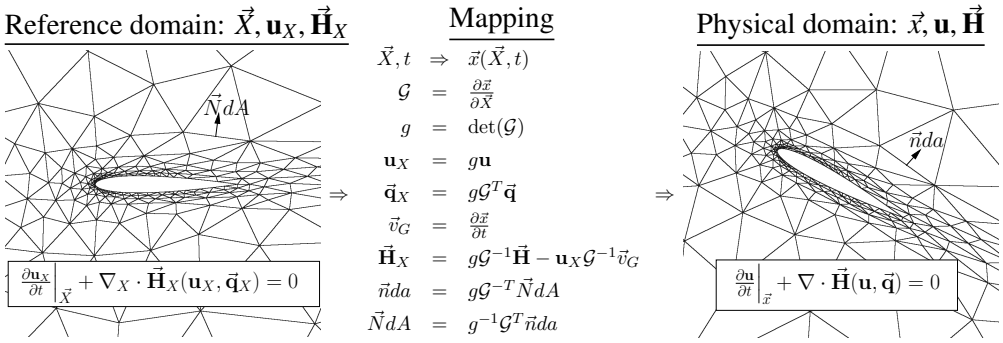


Figure 2: Summary of the mapping between reference and physical domains. The equations are solved on the reference domain, which remains fixed for all time. When denoting reference-domain quantities, we use a subscript X.

Table 1: Definitions of variables used in the ALE mapping. Bold indicates a state vector and an arrow indicates a spatial vector.

$\vec{X}$	=	reference-domain coordinates	$\vec{x}$	=	physical-domain coordinates
$\mathbf{u}_X$	=	state on reference domain	$\mathbf{u}$	=	physical state
$\vec{\mathbf{q}}_X$	=	gradient variable on reference domain	$\vec{\mathbf{q}}$	=	physical gradient variable
$\vec{\mathbf{H}}_X$	=	flux vector on reference domain	$\vec{\mathbf{H}}$	=	flux vector on physical domain
dA	=	differential area on reference domain	da	=	differential area on physical domain
$\vec{N}$	=	normal vector on reference domain	$\vec{n}$	=	normal vector on physical domain
V	=	reference domain (static)	$v(t)$	=	physical domain (dynamic)
$\mathcal{G}$	=	mapping Jacobian matrix	$\vec{v}_G$	=	grid velocity, $\partial\vec{x}/\partial t$
g	=	determinant of Jacobian matrix			

mapping $\vec{x}(\vec{X}, t)$ is analytical, obtained by blending rigid body motion in the vicinity of the moving object to zero far away from the object [32]. The resultant mapping Jacobian determinant, g, may not be polynomial in $\vec{X}$, so that a constant physical state may not be representable with polynomial trial functions in reference space. This leads to slight conservation errors that can be mitigated with a geometric conservation law [32, 31]. However, as these errors become smaller with higher-order approximation and adaptation [31], in this work we forgo a GCL.

3.2. Transformed Equations

To obtain the transformed conservation laws in reference space, we integrate the evolution PDE in (8) over a time-varying volume $v(t)$,

$$\int_{v(t)} \frac{\partial \mathbf{u}}{\partial t} dv + \int_{\partial v(t)} \vec{\mathbf{H}} \cdot \vec{n}\, da = \mathbf{0}. \tag{20}$$

The two integrals in (20) transform as

$$\begin{aligned} \int_{v(t)} \frac{\partial \mathbf{u}}{\partial t} &= \frac{d}{dt} \int_{v(t)} \mathbf{u}\, dv - \int_{\partial v(t)} (\mathbf{u}\vec{v}_G) \cdot \vec{n}\, da & (21) \\ &= \int_V \frac{\partial (g\mathbf{u})}{\partial t}\, dV - \int_{\partial V} (g\mathbf{u}\mathcal{G}^{-1}\vec{v}_G) \cdot \vec{N}\, dA, & (22) \\ \int_{\partial v(t)} \vec{\mathbf{H}} \cdot \vec{n}\, da &= \int_{\partial V} \vec{\mathbf{H}} \cdot (g\mathcal{G}^{-T}\vec{N})\, dA = \int_{\partial V} (g\mathcal{G}^{-1}\vec{\mathbf{H}}) \cdot \vec{N}\, dA. & (23) \end{aligned}$$

Substituting these expressions into (20) and applying the divergence theorem gives the PDE on the reference domain,

$$\left.\frac{\partial \mathbf{u}_X}{\partial t}\right|_{\vec{X}} + \nabla_X \cdot \vec{\mathbf{H}}_X(\mathbf{u}_X, \nabla_X \mathbf{u}_X) = \mathbf{0}, \tag{24}$$

where $\mathbf{u}_X = g\mathbf{u}$, $\vec{\mathbf{H}}_X = g\mathcal{G}^{-1}\vec{\mathbf{H}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G$, and ∇_X denotes the gradient with respect to the reference coordinates. The transformed flux, $\vec{\mathbf{H}}_X$, separates into inviscid and viscous

contributions,

$$\vec{\mathbf{H}}_X = \vec{\mathbf{F}}_X + \vec{\mathbf{G}}_X, \qquad \vec{\mathbf{F}}_X = g\mathcal{G}^{-1}\vec{\mathbf{F}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G, \qquad \vec{\mathbf{G}}_X = g\mathcal{G}^{-1}\vec{\mathbf{G}}. \tag{25}$$

(24) can now be discretized on the reference mesh. For DG, this discretization proceeds in a standard fashion [32, 31]. For HDG, we also need to transform the equation for the gradient, $\vec{\mathbf{q}}$, (7). To do this, we write $\nabla\mathbf{u}$ in reference-space variables via the chain and product rules,

$$\nabla\mathbf{u} = \nabla_X(g^{-1}\mathbf{u}_X)\mathcal{G}^{-1} = g^{-1}\mathcal{G}^{-T}(\nabla_X\mathbf{u}_X - \mathbf{u}_X g^{-1}\nabla_X g), \tag{26}$$

Substituting into (7), we have

$$\begin{aligned} \vec{\mathbf{q}} - g^{-1}\mathcal{G}^{-T}(\nabla_X\mathbf{u}_X - \mathbf{u}_X g^{-1}\nabla_X g) &= \vec{\mathbf{0}}, & (27)\\ \underbrace{g\mathcal{G}^T\vec{\mathbf{q}}}_{\vec{\mathbf{q}}_X} - \nabla_X\mathbf{u}_X + \mathbf{u}_X g^{-1}\nabla_X g &= \vec{\mathbf{0}}, & (28) \end{aligned}$$

where we have defined the transformed reference-space gradient variable as $\vec{\mathbf{q}}_X = g\mathcal{G}^T\vec{\mathbf{q}}$. In summary, for HDG, the system of equations to be solved in reference space is

$$\begin{aligned} \vec{\mathbf{q}}_X - \nabla_X\mathbf{u}_X + \mathbf{u}_X g^{-1}\nabla_X g &= \vec{\mathbf{0}}, & (29)\\ \frac{\partial\mathbf{u}_X}{\partial t} + \nabla_X\cdot\vec{\mathbf{H}}_X(\mathbf{u}_X, \nabla_X\mathbf{u}_X) &= \mathbf{0}. & (30) \end{aligned}$$

Note the addition of a source term into the equation for $\vec{\mathbf{q}}$.

3.3. Implementation

An ALE solver for problems with mesh motion must operate on the reference-space equations in Eqns. 29–30. Equipping a DG or HDG code with such mesh motion capability does not require a wholesale rewrite. One can "retro-fit" an existing code to solve the reference space equations through relatively minor changes, mostly via pre/post-

processing operations on fluxes based on the reference-to-global mapping and its derivatives.

The weak form of Eqns. 29–30 is obtained by multiplying by reference-space test functions and integrating by parts over the reference-domain elements. The discretization would be straightforward were it not for the fact that fluxes and boundary conditions are specified on the physical domain. To minimize intrusion into the code, we express the reference-space fluxes and boundary conditions in terms of the physical fluxes and boundary conditions.

The inviscid flux on the reference domain includes the standard Galilean transformation expected from changing reference frames and also a multiplication by $g\mathcal{G}^{-1}$, which is done by post-processing the equation-set specific flux,

$$\vec{\mathbf{F}}_X = g\mathcal{G}^{-1}\vec{\mathbf{F}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G = g\mathcal{G}^{-1}\left(\vec{\mathbf{F}} - \mathbf{u}\vec{v}_G\right). \tag{31}$$

To account for the Galilean transformation on element interfaces, the Riemann solver needs to operate on $\vec{\mathbf{F}} - \mathbf{u}\vec{v}_G$ instead of just $\vec{\mathbf{F}}$. The reference-domain viscous flux is related to the physical viscous flux through

$$\vec{\mathbf{G}}_X = g\mathcal{G}^{-1}\vec{\mathbf{G}} = -g\mathcal{G}^{-1}\mathbf{K}\vec{\mathbf{q}} = -\underbrace{\mathcal{G}^{-1}\mathbf{K}\mathcal{G}^{-T}}_{\mathbf{K}_X}\vec{\mathbf{q}}_X. \tag{32}$$

In practice, the reference-space viscous flux is just obtained by multiplying the physical flux by $g\mathcal{G}^{-1}$, with the caveat that linearizations must be transformed from with respect to the physical states/gradients to with respect to the reference state/gradients.

The definition of the reference-space diffusion matrix $\mathbf{K}_X$ in (32) is useful in analyzing the form of stabilization and dual-consistency terms, which arise in terms of reference-space variables but are most conveniently implemented using physical-space quantities to minimize code intrusion. For example, consider the first viscous stabiliza-

tion in (14), which adds the following term to the reference space weak form,

$$\begin{aligned}\int_{\partial\Omega_{X,e}} \mathbf{w}^T \frac{\vec{N}\cdot\mathbf{K}_X\cdot\vec{N}}{\ell_{\text{X,visc}}}(\mathbf{u}_X-\hat{\mathbf{u}}_X)dA &= \int_{\partial\Omega_{X,e}} \mathbf{w}^T \frac{\mathcal{G}^{-1}\mathbf{K}\mathcal{G}^{-T}\vec{N}}{\ell_{\text{X,visc}}}(g\mathbf{u}-g\hat{\mathbf{u}})\vec{N}dA \\ &= \int_{\partial\Omega_e} \mathbf{w}^T \frac{\mathcal{G}^{-1}\mathbf{K}\mathcal{G}^{-T}\vec{N}}{\ell_{\text{X,visc}}}(\mathbf{u}-\hat{\mathbf{u}})\mathcal{G}^T\vec{n}da \\ &= \int_{\partial\Omega_e} \mathbf{w}^T \frac{\vec{n}\cdot\mathbf{K}\cdot\vec{n}}{\ell_{\text{visc}}}(\mathbf{u}-\hat{\mathbf{u}})da \end{aligned} \tag{33}$$

where we have transformed the viscous length via the relation $\vec{N}/\ell_{\text{X,visc}} = \mathcal{G}^T\vec{n}/\ell_{\text{visc}}$, which follows from assuming that the viscous length transforms like the element volume divided by the face area. Note that the expression in (33) is in terms of physical variables and is exactly the same stabilization that would be added for a discretization in the physical domain. The only change for mesh motion is then in the linearizations, which needs to be with respect to the reference quantities, and which can be handled via a chain-rule post-processing of the physical-variable linearizations.

Boundary conditions also require modifications when simulating problems on deformable domains. In particular, the physical boundary flux must be aware of motion on the boundary, $\vec{v}_G$. For example, on a moving wall, the flow tangency boundary condition states that the normal component of the fluid velocity is equal to the normal component of the boundary motion velocity (which would be zero without mesh motion). This physical consideration is separate from the subtraction of $\mathbf{u}^b\vec{v}_G$ from the flux – both must be included.

Calculation of the viscous contribution on a boundary requires not only the boundary state, $\mathbf{u}^b$, but also the boundary flux. For pure Dirichlet boundary conditions, the state gradient information is taken from the interior. In other cases, the physical viscous flux is prescribed on the boundary (e.g. zero heat flux for an adiabatic wall), and in these cases, the viscous flux contribution is added directly to the residual. Note that no

transformation needs to be applied to any flux dotted with the normal, since

$$\vec{\mathbf{H}} \cdot \vec{n} da = \left(g^{-1} \mathcal{G} \vec{\mathbf{H}}_X \right) \cdot \left(g \mathcal{G}^{-1} \vec{N} \right) dA = \vec{\mathbf{H}}_X \cdot \vec{N} dA. \tag{34}$$

Finally, a notable difference between the physical and transformed system of equations is the appearance of a source term in the reference-space gradient equation, (29). In the weak form this source term is evaluated in a straightforward manner via an element-interior integration.

3.4. Output Calculation

Output calculations on deforming domains often involve boundary integrals of a linear combination of the fluxes. In physical space, the general form of such outputs, which include forces, moment, and power, is

$$J = \int_{\partial\Omega} \mathbf{o}^T \widehat{\mathbf{H}} \cdot \vec{n} \, da, \tag{35}$$

where $\mathbf{o} \in \mathbb{R}^s$ is the output "test function" containing the weights of each conservation-law flux in the linear combination that forms the scalar output. For example, $\mathbf{o}$ could contain $\cos(\alpha)$ and $\sin(\alpha)$ terms weighting the conservation of momentum equations to produce a lift or drag force of an object at an angle of attack α. Moments could be computed by including physical-space coordinates in these weights. A power output can be obtained by including the grid velocity, $\vec{v}_G$, in the output test function $\mathbf{o}$. For example, for the Navier-Stokes equations, define $\mathbf{o}$ to pick of the conservation of momentum equations with weights given by $\vec{v}_G$. The resulting instantaneous power output, which can be time-integrated to yield work, is

$$J = \int_{\text{object}} \vec{v}_G \cdot \vec{f}_{\text{surf}} \, da,$$

where $\vec{f}_{\text{surf}}$ is the surface stress vector, i.e. the momentum flux components, and the integral is taken over the surface of the object, e.g. an airfoil. We can see this is a power by considering the special case of a grid velocity given by $\vec{v}_G = \vec{v}_0 + \vec{\omega} \times \vec{r}$, where $\vec{r}$ is a position vector relative to some origin of rotation. J becomes

$$J = \int_{\text{object}} (\vec{v}_0 + \vec{\omega} \times \vec{r}) \cdot \vec{f}_{\text{surf}}\, da = \int_{\text{object}} [\vec{v}_0 \cdot \vec{f}_{\text{surf}} + \vec{\omega} \cdot (\vec{r} \times \vec{f}_{\text{surf}})]\, da = \vec{v}_0 \cdot \vec{F}_{\text{net}} + \vec{\omega} \cdot \vec{T}_{\text{net}},$$

where $\vec{F}_{\text{net}}$ is the net force and $\vec{T}_{\text{net}}$ is the net torque exerted by the fluid on the object. This is the expected form of power that takes into account both translational and rotational contributions.

We note that the expression in (35) is written in physical-space variables, and this is convenient since no transformations are required from the physical flux calculations. When calculating boundary fluxes, stabilization terms from the discretization are retained for adjoint-consistency reasons [37].

4. Results

4.1. Scalar Advection Diffusion

For the purpose of verification, we consider a simple one-dimensional advection-diffusion problem. The governing equation and initial condition are

$$\begin{aligned} \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} &= 0, \quad x \in [0, L] \\ u(x, t = 0) &= u^0(x) = \exp\left[-(2x/L)^2\right]. \end{aligned}$$

In this work we use convenient, $O(1)$, units: $L = 4$, $a = 4$, and $\nu = aL/Pe$, where Pe is the Peclet number. The domain is periodic, and the output of interest for all runs is a weighted integral of the scalar u at the final time $T = 1$,

$$J = \int_0^L w(x)u(x, T)\, dx, \quad w(x) = \exp\left[-(2(x - x_0)/L)^2\right], \tag{36}$$

where $x_0 = L/40$.

Although this problem does not require mesh motion, we impose movement anyway to test the implementation of the ALE formulation and to compare errors and convergence rates to the case of no motion. The mapping from the reference domain to the physical domain is given by,

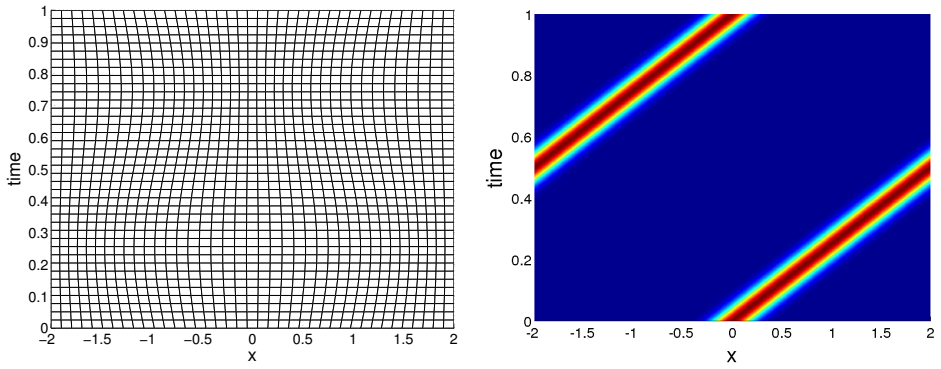
$$x(X,t) \quad = \quad X + \frac{L}{40}\sin(2\pi X/L)\sin(2\pi t/T).$$

Figure 3(a) illustrates this mapping.

4.1.1. Inviscid: $Pe \to \infty$

We first consider the inviscid case, $\nu = 0$, for which the initial condition advects undisturbed, as shown in Figure 3(b). Without viscosity, the HDG discretization behaves similarly to the DG discretization, since the state gradient is not approximated. We run both discretizations using orders of approximation $p = 1, 2, 3, 4$, on a sequence of uniformly refined meshes, starting with $N_{\text{elem}} = 5$ at the coarsest level. We also compute a "truth" solution on the finest mesh using $p = 6$, and output errors are calculated relative to the output from this truth solution. Fourth-order DIRK (DIRK4) is used as the time marching scheme, and the number of time steps for each run is chosen large enough such that the temporal error is small relative to the spatial error. We therefore expect the convergence rates to reflect the spatial order of accuracy.

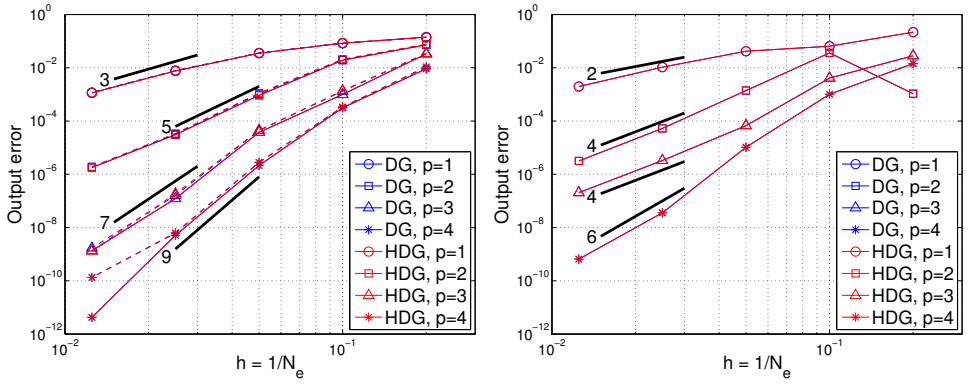
Figure 4(a) compares the convergence of DG and HDG, with and without mesh motion, for all four spatial orders, p. We see that, asymptotically, convergence rates of approximately $2p + 1$ are attained. This is the expected rate under uniform refinement for a scalar output computed in a problem that is free from singularities (both in the primal and the adjoint solution). The HDG and DG results, distinguished by a different color, are visually identical. In addition, the mesh motion results, shown as dashed lines accompanying each solid curve, are also nearly on top of the no-motion results,



(a) Mesh motion (b) $Pe \to \infty$ solution

Figure 3: Scalar advection diffusion: mesh motion and inviscid solution.

with the exception of the finest-mesh $p = 4$ case, where machine round-off and residual tolerances pollute the results.



(a) Least-squares projected initial condition (b) Interpolated initial condition

Figure 4: Scalar advection diffusion, $Pe \to \infty$: convergence of the final-time weighted integral output for DG and HDG, without mesh motion (solid lines) and with mesh motion (dashed lines). In (a), the initial condition is obtained by least-squares projection, while in (b) it is obtained by interpolation at Lagrange nodes.

Whether or not the optimal spatial convergence rate of $2p + 1$ is obtained depends on how the solution is initialized at $t = 0$. For the results in Figure 4(a), a least-squares projection was employed. An alternate method is a simple interpolation of the initial condition at equally-spaced nodes of order p Lagrange polynomials. With this interpo-

lated initial condition, we obtain the results in Figure 4(b). Note the larger errors and lower rates – between $p+1$ and $p+2$ – compared to the least-squares initialization.

The sub-optimality of this interpolation can be explained by looking at the output error introduced during initialization at $t = 0$. Consider the exact initial condition, $u^0_{\text{exact}}(x)$, projected or interpolated to the space of order p polynomials to yield $u^0_h(x) = u_{\text{exact}}(x) + \delta u^0_h(x)$. Polynomial interpolation theory tells us that for an order p representation on elements of size Δx, we have $\delta u^0_h(x) = O(\Delta x^{p+1})$, where Δx is the element size. We now evaluate the output J via (36) using $u^0_h(x)$,

$$J_h = \int_0^L w(x) u^0_h(x) dx = \int_0^L w(x) \left[u_{\text{exact}}(x) + \delta u^0_h(x) \right] dx = J_{\text{exact}} + \int_0^L w(x) \delta u^0_h(x)\, dx$$

Based on the estimate $\delta u^0_h(x) = O(\Delta x^{p+1})$, and if we don't expect any cancellation in the weighted integral, the error in the output (for just the initial condition) will be

$$\delta J_h = \int_0^L w(x) \delta u^0_h(x)\, dx = O(\Delta x^{p+1}).$$

Indeed, this is the lower bound (observed for odd orders), on the rate attained for initial condition interpolation at Lagrange nodes, in Figure 4(b). However, for least-squares projection, the integral of $\delta u_h(x)$ multiplied by any polynomial of order p should be zero. So our initial-condition output error in this case should converge like

$$\begin{aligned} \delta J_h &= \int_0^L w(x) \delta u^0_h(x)\, dx = \int_0^L (w(x) - w_h(x)) \delta u^0_h(x)\, dx \\ &= \int_0^L \delta w_h(x) \delta u^0_h(x)\, dx = O(\Delta x^{2p+2}), \end{aligned}$$

where $w_h(x)$ is the least-squares projection of $w(x)$ onto order p polynomials on each element and $\delta w_h(x) = O(\Delta x^{p+1})$ is the resulting difference. This rate can be easily verified by measuring the output at $t = 0$, and indeed it is attained. The additional errors incurred in propagating the solution from $t = 0$ to $t = T$ bring the final observed rate to

$2p + 1$.

4.1.2. Advection-dominated: $Pe = 320$

We next run the same problem but with a small amount of viscosity, such that $Pe = 320$. The HDG discretization now differs from DG fundamentally in that HDG approximates the state gradient. For DG, we use the BR2 viscous discretization, whereas for HDG we consider viscous stabilization using (1) a BR2-like term and (2) a constant viscous length scale term, as presented in (14). In this simple verification test case, no robustness problems were encountered with either viscous discretization.

The same suite of test cases described in the previous section was run: orders $p = 1, 2, 3, 4$, mesh motion on/off, and three discretizations. Figure 5 presents the convergence of the output error under uniform mesh refinement. We first observe that runs

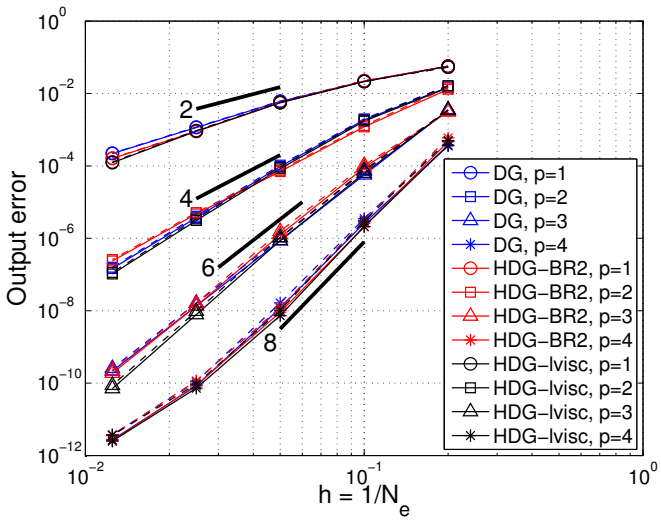


Figure 5: Scalar advection diffusion, $Pe = 320$: convergence of the final-time weighted integral output for DG and HDG, without mesh motion (solid lines) and with mesh motion (dashed lines). HDG was run with two viscous dicretizations, BR2 and "lvisc", which indicates a constant viscous length scale, $\ell_{\text{visc}} = \nu$.

with mesh motion on (dashed lines) show nearly identical errors to corresponding runs with mesh motion off (solid lines). Therefore, the presence of mesh motion does not pollute the output to a noticeable degree. In addition, we see that for each order p, the

results from the different discretizations are all close, with spreads occurring at the finest meshes. DG and HDG-BR2 exhibit asymptotic convergence rates of approximately $2p$, which is optimal for a discretization of an equation with second derivatives. However, as HDG discretizes a first-order system, we would expect its optimal rate to still be $2p + 1$. As discussed in [36], this optimal rate is attainable for an appropriate choice of viscous stabilization: for example, a constant viscous length scale. Indeed, we see that HDG with the constant viscous length stabilization does begin to exhibit a faster convergence rate at the fine meshes. However, the difference is not very large for this problem, as the Peclet number is high.

4.1.3. Diffusion-dominated: $Pe = 16$

Finally, we consider the same problem as in the previous section but with more viscosity, such that $Pe = 16$. We again run a convergence study of the output error for a final-time weighted integral of the scalar unknown. Figure 6 shows the resulting convergence rates. We see that in this diffusion-dominated regime, the output error converges at a rate of approximately $2p$ for DG and HDG-BR2. Again, this is expected for a diffusion discretization with DG. More clear in this case is the benefit of the constant viscous length stabilization for HDG: we see markedly lower errors in these runs compared to DG and HDG-BR2, and convergence rates of approximately $2p + 1$. However, as this example shows, the advantage is most relevant for diffusion-dominated problems. In the case of the compressible Navier-Stokes equations, we consider high Reynolds number flows, and in these cases we have observed solver robustness improvements for HDG-BR2, without noticeable effects on accuracy. Finally, mesh motion does not significantly change the results, except in the pre-asymptotic regime and at low error levels.

4.2. Euler Vortex

As another verification test, this time for a system of equations, we consider an analytical vortex solution to the Euler equations [32]. The state vector at point (x, y) and time t is obtained via

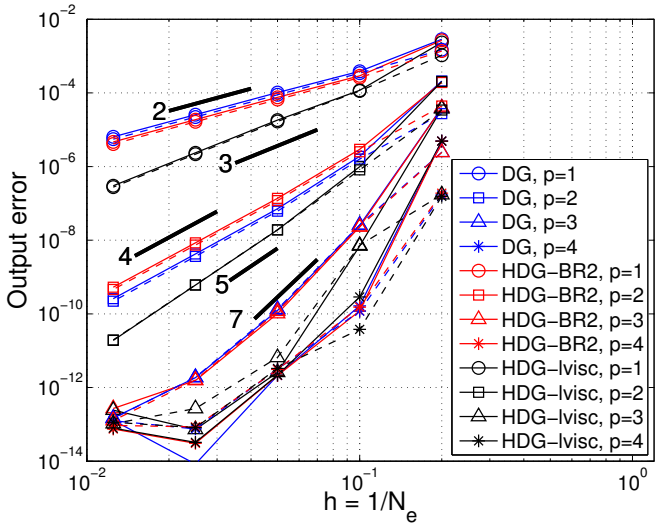


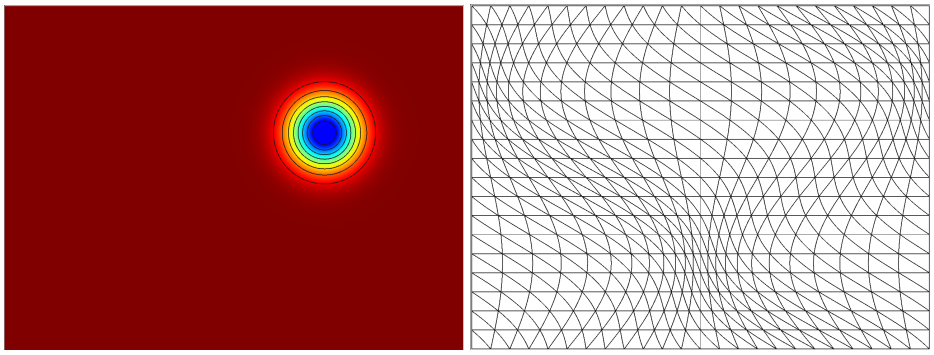
Figure 6: Scalar advection diffusion, $Pe = 16$: convergence of the final-time weighted integral output for DG and HDG, without mesh motion (solid lines) and with mesh motion (dashed lines). HDG was run with two viscous dicretizations, BR2 and "lvisc", which indicates a constant viscous length scale, $\ell_{\text{visc}} = \nu$.

$$
\begin{aligned}
f_0 &= 1 - |\vec{x} - \vec{x}_0 - \vec{V}_\infty t|^2 / r_c^2 \\
M_\infty &= |\vec{V}_\infty| \sqrt{\rho_\infty / (\gamma p_\infty)} \\
f_1 &= 1 - \epsilon^2 (\gamma - 1) M_\infty^2 \exp(f_0)/(8\pi^2) \\
\rho &= \rho_\infty f_1^{1/(\gamma-1)} \\
p &= p_\infty f_1^{\gamma/(\gamma-1)}
\end{aligned}
\qquad
\begin{aligned}
f_2 &= |\vec{V}_\infty| \epsilon \exp(f_0/2)/(2\pi r_c) \\
u &= U_\infty - f_2 (y - y_0 - V_\infty t) \\
v &= V_\infty + f_2 (x - x_0 - U_\infty t) \\
\mathbf{u} &= \left[\rho, \rho u, \rho v, \frac{p}{\gamma - 1} + \frac{1}{2} \rho (u^2 + v^2) \right]
\end{aligned}
$$

where we use the following constants (convenient units): $\vec{x}_0 = (5, 5)$, $\vec{V}_\infty = (U_\infty, V_\infty) = (2, 1)/\sqrt{5}$, $\rho_\infty = 1$, $p_\infty = 20/7$, $\gamma = 1.4$, $\epsilon = 0.3$, $r_c = 1.5$. We solve this problem on a rectangular domain, $[0, 20] \times [0, 15]$. Though no mesh motion is required, for verification we impose the following domain deformation from $\vec{X} = (X, Y)$ to $\vec{x}(t) = (x(t), y(t))$:

$$
\begin{aligned}
x(t) &= X + 2 \sin(2\pi X/20) \sin(2\pi Y/15) \sin(2\pi t), \\
y(t) &= Y + 1.5 \sin(2\pi X/20) \sin(2\pi Y/15) \sin(4\pi t).
\end{aligned}
$$

Figure 7 shows the final-time ($t = 10$) solution and a deformed mesh at $t = 2.5$. We



(a) Pressure at $t = 10$ (b) Mesh at $t = 2.5$

Figure 7: Euler vortex verification: pressure contours at final time and one mesh from the refinement study, shown deformed at an intermediate time of $t = 2.5$. The exact solution is known in this case and L_2 errors of the state can be computed.

simulate this case using both DG and HDG, for various state approximation orders p and DIRK4 time stepping, and we compute the L_2 norm of the state error at the final time, $t = 10$. Figure 8 shows the convergence of this error with uniform spatial mesh refinement. The time steps were chosen small enough so as to keep the temporal errors relatively small. The HDG results lie virtually on top of the DG results in this case, and both schemes attain the expected $p + 1$ L_2 error convergence rates.

4.3. NACA 0012 Airfoil in Pitch and Plunge Motion

In this case we solve the compressible Navier-Stokes equations ($\gamma = 1.4, Pr = 0.72$, constant viscosity) to simulate flow over a modified NACA 0012 airfoil undergoing pitch and plunge motion. The geometry of the NACA 0012 airfoil is modified to close the trailing edge via the following equation,

$$y(x) \quad = \quad \pm 0.6(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1036x^4), \qquad x \in [0, 1].$$

The initial condition is a steady state solve at a free-stream Mach number of $M_\infty = 0.2$, and a Reynolds number $Re = 5000$. We use convenient units in which the airfoil chord is

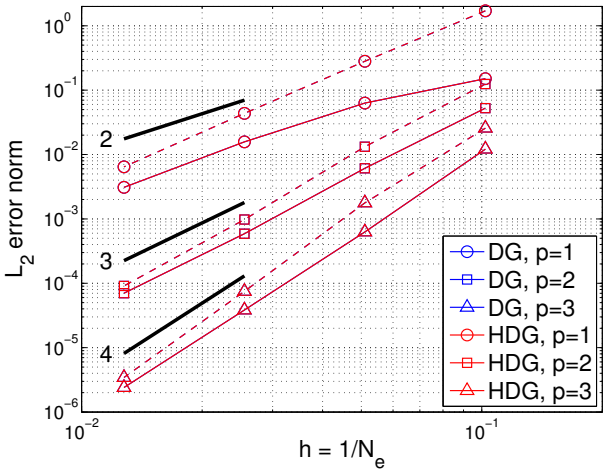


Figure 8: Euler vortex verification: convergence of the L_2 norm of the state error with uniform spatial refinement for DG and HDG (curves visually overlap). The solid lines correspond to static meshes (no motion), and the dashed lines correspond to deforming meshes.

$c = 1$ and the free-stream density and speed are unity, so that the free-stream conservative state vector is

$$[\rho, \rho u, \rho v, \rho E] = \left[1,\ 1,\ 0,\ 0.5+1/[M_\infty^2 \gamma(\gamma-1)]\right].$$

Full-state boundary conditions are imposed on the farfield boundary, which is 2000 chord-lengths away. The meshes consist of unstructured triangles, curved to quartic geometry representation on the boundary.

Following a steady-state solve for the initialization, the unsteady simulation begins with a combined pitch/plunge motion. The vertical displacement, $h(t)$, and the pitch angle, $\theta(t)$, are given by

$$h(t) = A\frac{1-\cos \pi t}{2}, \qquad \theta(t) = \tan^{-1}\left(B \sin \pi t\right),$$

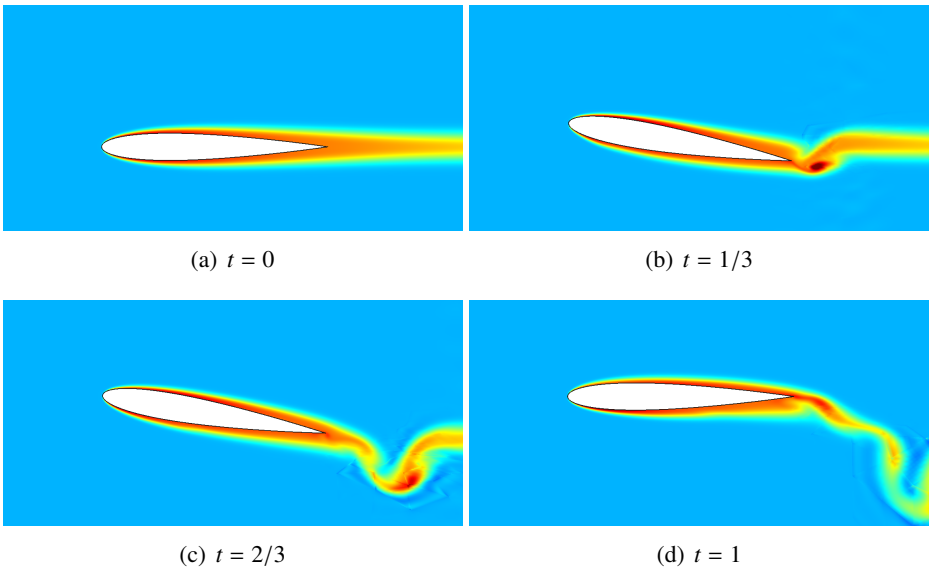
where $A = 0.2$ and $B = \pi/16$. Finally, the output of interest is the total energy integrated

over the domain at the final time $t = T = 1$:

$$J = \int_{\Omega} \rho E(\vec{x}, T)\, dA. \tag{37}$$

Note that this output measures the work done on the flow by the airfoil – this work appears as the energy deposited in the flow.

Figure 9 shows contours of entropy in the flow at various times during the unsteady simulation. The combined pitch-up and upward-plunge motion of the airfoil disturbs both the wake and the boundary layer near the trailing edge of the airfoil.

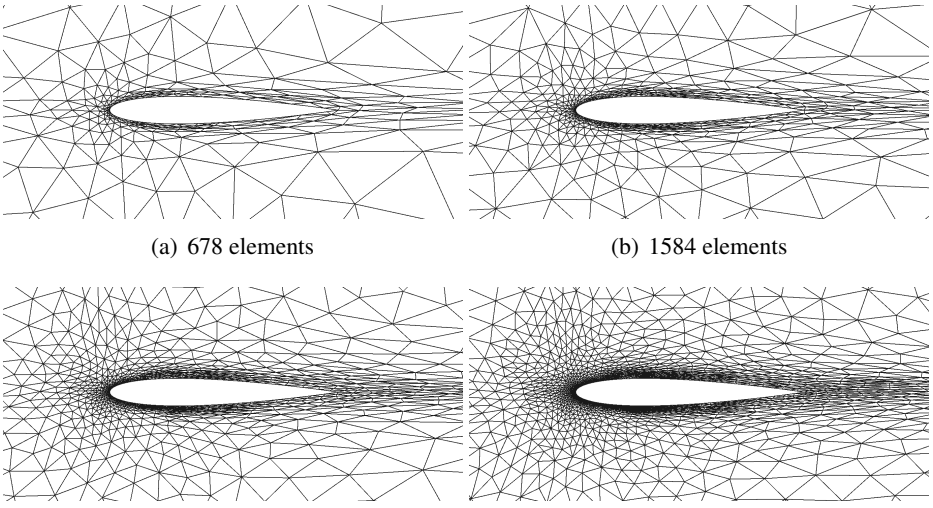


(a) $t = 0$ (b) $t = 1/3$

(c) $t = 2/3$ (d) $t = 1$

Figure 9: Compressible Navier-Stokes flow over a pitching and plunging NACA 0012 airfoil: contours of entropy at various times in the course of the simulation.

To asses the convergence of the output with mesh refinement, both DG and HDG (BR2 viscous discretization) were run on a sequence of quasi-uniformly refined meshes, shown in Figure 10. The starting mesh for this sequence was a mesh adapted anistoropically to predict steady-state drag. Subsequent meshes were obtained by global re-meshing using a grid-implied metric scaled by a constant factor to create an approximate doubling of degrees of freedom on each finer mesh.



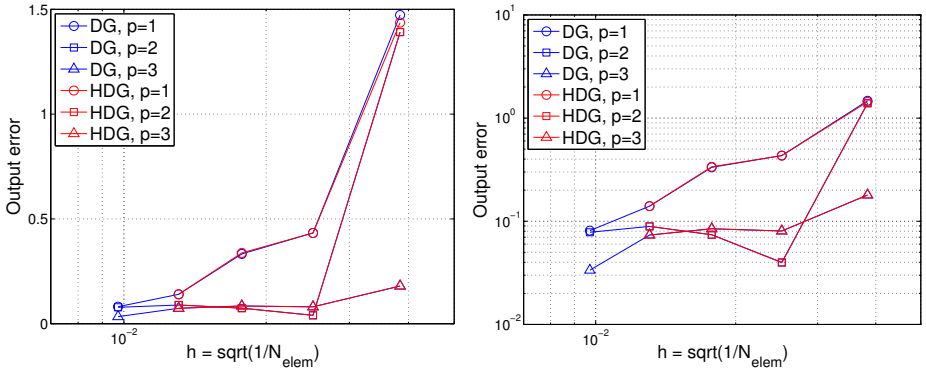
(a) 678 elements (b) 1584 elements

(c) 3169 elements (d) 5866 elements

Figure 10: Compressible Navier-Stokes flow over a pitching and plunging NACA 0012 airfoil: first four meshes in sequence for uniform refinement convergence studies. Elements adjacent to the airfoil surface are curved to provide fourth-order geometry approximation.

DIRK4 was the time-marching method for all runs, with the number of time steps sufficiently high so that the temporal error was small relative to the spatial error. A truth solution was computed on a fine mesh consisting of one more uniform refinement and spatial approximation order $p = 4$. Figure 11 shows the convergence of the error in the final flow energy output using the mesh sequence in Figure 10. We see that HDG and DG produce nearly identical results for all of the runs, with the largest deviations on the coarsest meshes. In addition, as expected, the errors decrease with increasing approximation order and with mesh refinement. This result demonstrates that the HDG ALE formulation is very similar to the DG ALE formulation for this problem, which is characterized by a relatively high Reynolds number. As we observed in the one-dimensional verification study, for diffusion-dominated flows, HDG can attain higher accuracy over DG. In addition, for increasing orders p, the growth of the system scales only linearly for HDG, whereas it scales quadratically for DG.

The results in Figure 11 show a drop in the error that is not always monotonic (e.g.

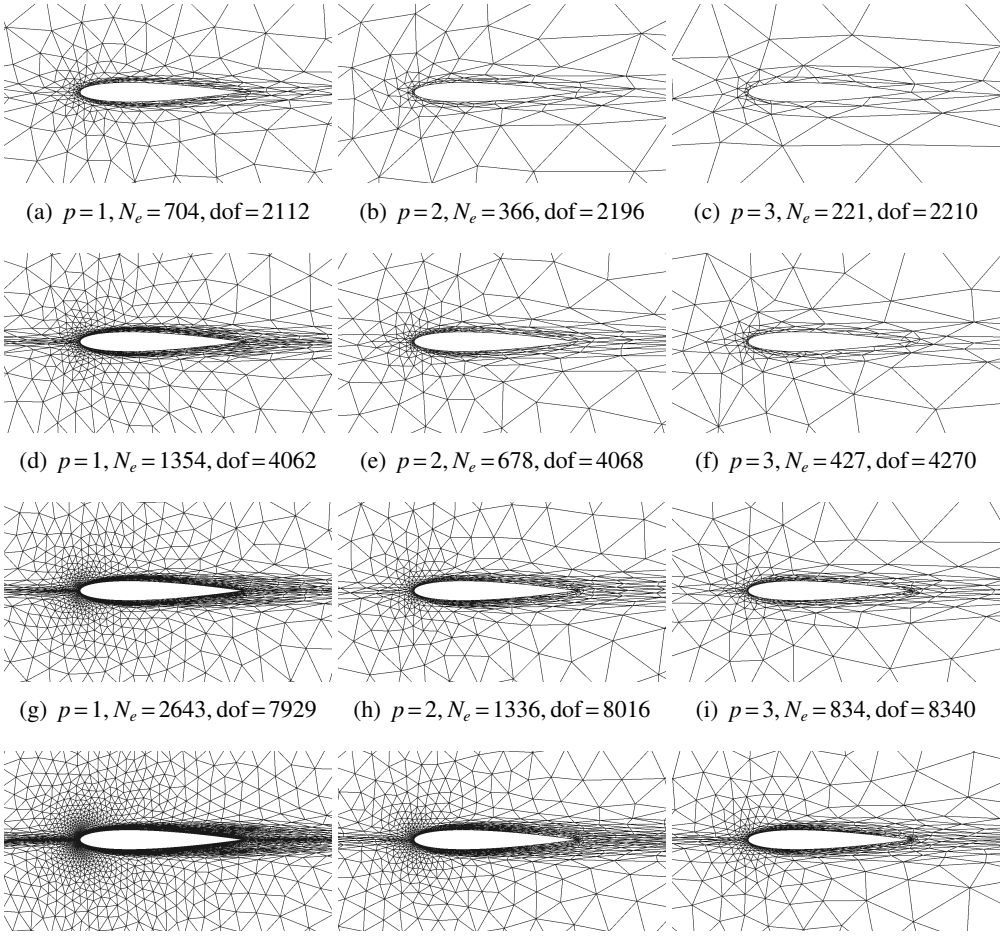


(a) Linear scale (b) Log scale

Figure 11: Compressible Navier-Stokes flow over a pitching and plunging NACA 0012 airfoil: convergence of the final-time flow energy integral output relative to a truth solution on a fine mesh. The sequence of meshes is obtained by quasi-uniform refinement of an initial mesh.

for $p = 2$), and one may reasonably ask whether uniform refinement is the most efficient choice for this case. Note that while the unsteady simulation is relatively short, mesh resolution is required further away from the airfoil during the steady-state initialization, and hence the task of generating an optimal mesh is not trivial. In the present study we address this question by a simple experiment: we generate a sequence of adapted spatial meshes using steady-state mesh optimization [38, 39] on the drag output. These are constructed specific to each order by prescribing a fixed number of degrees of freedom (dof). Figure 12 shows the first four meshes of the sequences for each order. Note the decrease in the number of elements with increasing p for a fixed dof.

We now run the unsteady simulations with DG and HDG, at orders $p = 1, 2, 3$, using the mesh sequence specific to each order p. Figure 13 shows the convergence of the final-time total energy output. Compared to uniform refinement of a single mesh, we see much larger values of the output error at comparable mesh sizes, and large oscillations in the error that do not appear to diminish with mesh refinement. This result demonstrates that, even though a steady-solve initialization is used for the unsteady runs, meshes optimized for this steady solve can be severely sub-optimal for the unsteady sim-



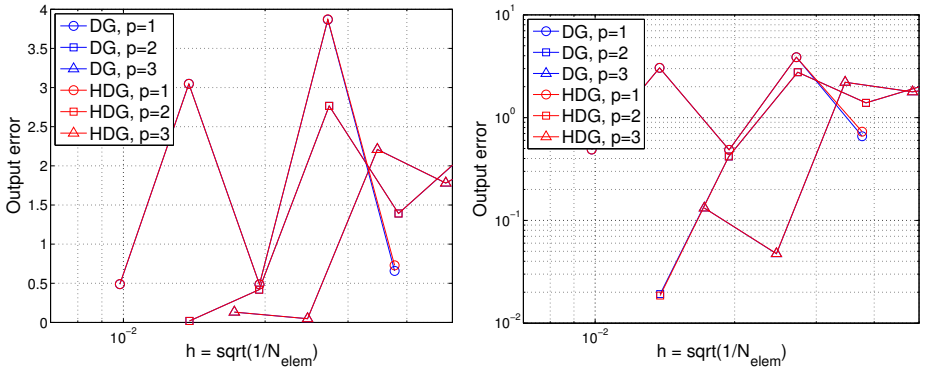
(a) $p=1, N_e=704, \text{dof}=2112$ (b) $p=2, N_e=366, \text{dof}=2196$ (c) $p=3, N_e=221, \text{dof}=2210$

(d) $p=1, N_e=1354, \text{dof}=4062$ (e) $p=2, N_e=678, \text{dof}=4068$ (f) $p=3, N_e=427, \text{dof}=4270$

(g) $p=1, N_e=2643, \text{dof}=7929$ (h) $p=2, N_e=1336, \text{dof}=8016$ (i) $p=3, N_e=834, \text{dof}=8340$

(j) $p=1, N_e=5274, \text{dof}=15822$ (k) $p=2, N_e=2650, \text{dof}=15900$ (l) $p=3, N_e=1650, \text{dof}=16500$

Figure 12: Compressible Navier-Stokes flow over a pitching and plunging NACA 0012 airfoil: first four meshes in order-specific adapted meshes, generated by steady-state mesh optimization for drag prediction.



(a) Linear scale (b) Log scale

Figure 13: Compressible Navier-Stokes flow over a pitching and plunging NACA 0012 airfoil: convergence of the final-time flow energy integral output relative to a truth solution on a fine mesh. The sequence of meshes is obtained by steady-state output-based adaptive mesh refinement on drag.

ulations. The steady-state optimized meshes are indeed tailored for the prediction of the single output, steady-state drag. While the near-field integration of the drag force resembles the integration required for a power calculation, in the unsteady simulation, mesh resolution further away from the airfoil, in areas dictated by the mesh motion, affects the unsteady output as well. Uniform refinement blindly adds such resolution, whereas the steady-state optimization may leave certain areas with consistently-low resolution. This example motivates the use of unsteady output-based adaptive techniques, which we have already investigated for DG [29, 31], and which are the subject of ongoing work for HDG.

5. Conclusions

In this paper we present a hybrid discontinuous Galerkin (HDG) discretization for unsteady simulations on deformable domains. We present details of the ALE formulation for HDG, including the transformation of the gradient equation. Compared to traditional DG, HDG offers computational advantages at high-order approximation due to static condensation of element-interior degrees of freedom. Such computational sav-

ings extend to implicit unsteady simulations, as the sizes of the systems to be solved at each time step are reduced. In addition, through a separate approximation of the gradient, HDG can yield improved (optimal) convergence of outputs computed from the gradient of the state.

Results from a one-dimensional advection-diffusion verification study on a non-deforming periodic domain show that solutions on moving meshes are very close to those on non-moving meshes. Specifically, the differences in scalar output errors between the two are often negligible relative to the size of the discretization error. For the Euler verification example, the differences are larger, most likely due to the larger magnitude deformation in this case. However, the differences decrease with increasing mesh resolution, giving credence to the implementation of the ALE formulation.

The one-dimensional verification tests also highlight the importance of the initial condition on output convergence rates. Optimal rates are not observed for all initial condition interpolations, the pointwise errors of which are all $O(h^{p+1})$. To attain optimal output error rates, the initial conditions must come from a projection that does not introduce output errors higher than the expected optimal rate, e.g. $2p+1$ for advection flows. A least-squares projection, with initial condition errors in integral outputs of $O(h^{2p+2})$ is sufficient in this regard.

Two viscous stabilization methods were tested for the HDG discretization: one based on a constant viscous length scale, and one based on the second form of Bassi and Rebay (BR2), which yields a viscous length scale that scales inversely with the mesh size, $1/h$. The constant viscous length scale stabilization yields optimal convergence rates for diffusion-dominated problems, but the BR2 form (with its higher jump penalization for smaller grid sizes) was observed to be more robust for the compressible Navier-Stokes simulations. For advection-dominated flows, the differences in the accuracy between these two methods is not large.

A test with the full compressible Navier-Stokes equations at moderate Reynolds number on a pitching and plunging airfoil revealed that both discretizations, DG and

HDG, produce nearly the same output on a given mesh. Under uniform refinement, output accuracy improved, and high order spatial approximation yielded higher accuracy for a given mesh, as expected. To test whether the convergence of the output could be improved, an adaptive sequence of meshes was generated, one sequence for each approximation order, using steady-state drag-based mesh optimization. However, the unsteady, deforming simulations with these meshes resulted in high errors in the final flow energy output and did not exhibit convergence with increasing degrees of freedom. This behavior motivates the need for unsteady output-based mesh adaptation, which is the subject of ongoing work. This will include not only spatial refinement, but also temporal refinement, so that both errors can be kept appropriately in check. We note that no geometric conservation law was implemented in this study, and while geometric conservation errors are present in the ALE runs, their effect on output accuracy is comparable to that of discretization error and diminishes with high order and especially adaptive mesh refinement.

Finally, another topic of future work is a computational time comparison of HDG and DG for cases similar to those considered here. Presently, neither solver is optimized for simulations with mesh motion, and while wall clock time advantages were observed for HDG starting at $p = 3$, detailed and reliable timings are not yet available. Furthermore, such timings will likely have to incorporate parallel efficiency considerations on distributed memory architectures, for large-scale simulations.

Acknowledgments

The author acknowledges support from the Air Force Office of Scientific Research under grant FA9550-11-1-0081, and from the Department of Energy under grant DE-FG02-13ER26146/DE-SC0010341.

References

[1] N. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations, Journal of Computational Physics 228 (2009) 3232–3254.

[2] B. Cockburn, J. Gopalakrishnan, R. Lazarov, Unified hybridization of discontinuous Galerkin, mixed, and continuous galerkin methods for second order elliptic problems, SIAM Journal on Numerical Analysis 47 (2) (2009) 1319–1365.

[3] N. C. Nguyen, J. Peraire, B. Cockburn, Hybridizable discontinuous Galerkin methods, in: J. S. Hesthaven, E. M. Rnquist, T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick (Eds.), Spectral and High Order Methods for Partial Differential Equations, Vol. 76 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2011, pp. 63–84, 10.1007/978-3-642-15337-2_4.
URL http://dx.doi.org/10.1007/978-3-642-15337-2_4

[4] J. Peraire, N. C. Nguyen, B. Cockburn, An embedded discontinuous galerkin method for the compressible euler and navier-stokes equations, AIAA Paper 2011-3228 (2011).

[5] S. Rhebergen, B. Cockburn, Space-time hybridizable discontinuous Galerkin method for the advectiondiffusion equation on moving and deforming meshes, in: C. A. de Moura, C. S. Kubrusly (Eds.), The CourantFriedrichsLewy (CFL) Condition, Birkhäuser Boston, 2013, pp. 45–63. doi:10.1007/978-0-8176-8394-8_4.
URL http://dx.doi.org/10.1007/978-0-8176-8394-8_4

[6] W. Reed, T. Hill, Triangular mesh methods for the neutron transport equation, Los Alamos Scientific Laboratory Technical Report LA-UR-73-479 (1973).

[7] B. Cockburn, C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (3) (2001) 173–261.

[8] K. J. Fidkowski, T. A. Oliver, J. Lu, D. L. Darmofal, *p*-Multigrid solution of high–order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, Journal of Computational Physics 207 (2005) 92–113. doi:10.1016/j.jcp.2005.01.005.

[9] K. S. Bey, J. T. Oden, *hp*-version discontinuous Galerkin methods for hyperbolic conservation laws, Computer Methods in Applied Mechanics and Engineering 133 (1996) 259–286.

[10] P. Houston, R. Hartmann, E. Süli, Adaptive discontinuous Galerkin finite element methods for compressible fluid flows, in: M. Baines (Ed.), Numerical Methods for Fluid Dynamics VII, ICFD, 2001, pp. 347–353.

[11] F. Bassi, S. Rebay, Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes equations, International Journal for Numerical Methods in Fluids 40 (2002) 197–207.

[12] F. Brezzi, L. Marini, E. Süli, Discontinuous Galerkin methods for first-order hyperbolic problems, Mathematical Models and Methods in Applied Sciences 14 (2004) 1893–1903.

[13] N. A. Pierce, M. B. Giles, Adjoint recovery of superconvergent functionals from PDE approximations, SIAM Review 42 (2) (2000) 247–264.

[14] R. Becker, R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, in: A. Iserles (Ed.), Acta Numerica, Cambridge University Press, 2001, pp. 1–102.

[15] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, Journal of Computational Physics 183 (2) (2002) 508–532.

[16] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows, Journal of Computational Physics 187 (1) (2003) 22–46.

[17] S. Sen, K. Veroy, D. Huynh, S. Deparis, N. Nguyen, A. Patera, "Natural norm" a posteriori error estimators for reduced basis approximations, Journal of Computational Physics 217 (2006) 37–62.

[18] M. Nemec, M. J. Aftosmis, Error estimation and adpative refinement for embedded-boundary Cartesian meshes, AIAA Paper 2007-4187 (2007).

[19] K. J. Fidkowski, D. L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, American Institute of Aeronautics and Astronautics Journal 49 (4) (2011) 673–694. doi:10.2514/1.J050073.

[20] K. Fidkowski, High-order output-based adaptive methods for steady and unsteady aerodynamics, in: H. Deconinck, R. Abgrall (Eds.), 37^{th} Advanced CFD Lectures series; Von Karman Institute for Fluid Dynamics (December 9–12, 2013), von Karman Institute for Fluid Dynamics, 2013.

[21] J. P. Dahm, K. J. Fidkowski, Error estimation and adaptation in hybridized discontinous Galerkin methods, AIAA Paper 2014–0078 (2014). doi:10.2514/6.2014-0078.

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

[23] M. Schmich, B. Vexler, Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations, SIAM Journal on Scientific Computing 30 (1) (2008) 369–393.

[24] T. J. Barth, Space-time error representation and estimation in Navier-Stokes calculations, in: S. C. Kassinos, C. A. Langer, G. Iaccarino, P. Moin (Eds.), Complex Effects in Large Eddy Simulations, Springer Berlin Heidelberg, Lecture Notes in Computational Science and Engineering Vol 26, 2007, pp. 29–48.

[25] M. Besier, R. Rannacher, Goal-oriented space-time adaptivity in the finite element galerkin method for the compution of nonstationary incompressible flow, International Journal for Numerical Methods in Fluids 70 (2012) 1139–1166.

[26] K. Mani, D. J. Mavriplis, Error estimation and adaptation for functional outputs in time-dependent

flow problems, Journal of Computational Physics 229 (2010) 415–440.

[27] A. Belme, A. Dervieux, F. Alauzet, Error estimation and adaptation for functional outputs in time-dependent flow problems, Journal of Computational Physics 231 (2012) 6323–6348.

[28] B. T. Flynt, D. J. Mavriplis, Discrete adjoint based adaptive error control in unsteady flow problems, AIAA Paper 2012-0078 (2012).

[29] K. J. Fidkowski, Y. Luo, Output-based space-time mesh adaptation for the compressible Navier-Stokes equations, Journal of Computational Physics 230 (2011) 5753–5773. doi:10.1016/j.jcp.2011.03.059.

[30] K. J. Fidkowski, An output-based dynamic order refinement strategy for unsteady aerodynamics, AIAA Paper 2012-77 (2012). doi:10.2514/6.2012-77.

[31] S. M. Kast, K. J. Fidkowski, Output-based mesh adaptation for high order Navier-Stokes simulations on deformable domains, Journal of Computational Physics 252 (1) (2013) 468–494. doi:10.1016/j.jcp.2013.06.007.

[32] P.-O. Persson, J. Bonet, J. Peraire, Discontinuous Galerkin solution of the Navier-Stokes equations on deformable domains, Computer Methods in Applied Mechanics and Engineering 198 (2009) 1585–1595.

[33] S. Rhebergen, B. Cockburn, A spacetime hybridizable discontinuous galerkin method for incompressible flows on deforming domains, Journal of Computational Physics 231 (11) (2012) 4185 – 4204. doi:10.1016/j.jcp.2012.02.011.
URL http://www.sciencedirect.com/science/article/pii/S0021999112000903

[34] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations, in: K. Cockburn, Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Springer, Berlin, 2000, pp. 197–208.

[35] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (1981) 357–372.

[36] B. Cockburn, B. Dong, J. Guzman, M. Restelli, R. Sacco, A hybridizable discontinuous Galerkin method for steady-state convection-diffusion-reaction problems, SIAM Journal on Scientific Computing 31 (5) (2009) 3827–3846.

[37] J. Lu, An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (2005).

[38] M. Yano, An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (2012).

[39] K. J. Fidkowski, A local sampling approach to anisotropic metric-based mesh optimization, AIAA Paper SciTech (2016).