An Output-Based Adaptive Hybridized Discontinuous Galerkin Method on Deforming Domains

Krzysztof J. Fidkowski*

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

In this paper we present an output-based adaptive method for unsteady simulations of convection-dominated flows on deformable domains. The target discretization is the hybridized discontinuous Galerkin method (HDG), which offers potential computational savings at high order compared to the discontinuous Galerkin (DG) method. 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. The temporal discretization for adaptive runs is DG in time, which lends itself to rigorous a posteriori error estimation using a fine-space discrete adjoint solution. We present modifications to the approximate factorization technique that enables an efficient DG-in-time solution for HDG. Space-time error estimates drive metric-based static spatial refinement on unstructured spatial meshes and slab-based temporal nodalizations. We show accuracy and cost comparisons between adaptive DG and HDG simulations of two-dimensional flows governed by the compressible Navier-Stokes equations on deforming domains.

I. 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),[1–5] which under certain circumstances, notably at high order, can be computationally cheaper than the popular discontinuous Galerkin (DG) method.[6–12]

A relatively fair way to compare 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,[13–19] and recently with HDG discretizations.[20–22] 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.[23–31]

In this work, we extend unsteady output-based adaptation techniques to high-order compressible Navier-Stokes simulations on deforming domains, discretized with HDG. We build on our previous work with DG,[31] and we note requisite discretization modifications for HDG in an ALE

*Associate Professor, AIAA Senior Member

formulation. We also extend our spatial mesh adaptation mechanics from hanging-node refinement to unstructured metric-based refinement. We compare HDG and DG results in an adaptive setting for various unsteady simulations, and we show the benefits of output-based adaptation compared to uniform mesh refinement.

II. Hybridized Discontinuous Galerkin Discretization

II.A. Primal

We use an adaptive hybridized discontinuous Galerkin (HDG) method to solve the systems of PDEs governing the flow. Figure 1 illustrates the primary differences between HDG and the standard discontinuous Galerkin method (DG), namely the introduction of additional degrees of freedom that decouple elements from each other in the global system and yield a global system of smaller size at high order compared to DG.

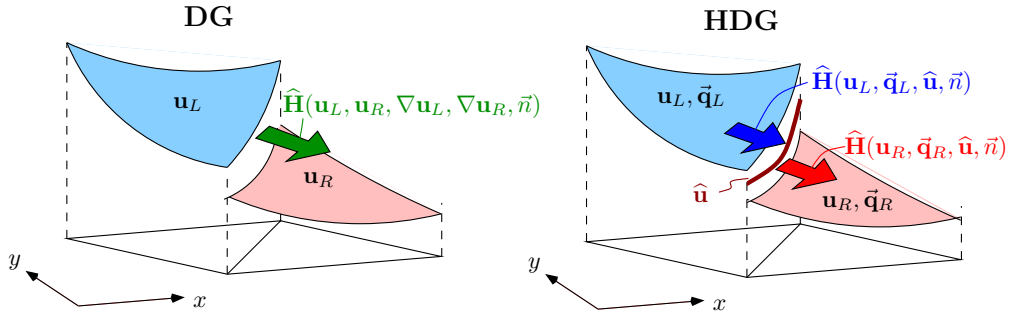


Figure 1. In the HDG method, additional unknowns on element interfaces allow elimination of the element-interior unknowns. This results in a global system in which the number of unknowns scales as $p^{\text{dim}-1}$ instead of p^{dim} for DG.

The HDG method applies to a first-order system of PDEs in conservative form,

$$\vec{\mathbf{q}} - \nabla\mathbf{u} = \vec{\mathbf{0}}, \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{H}} = \mathbf{0}, \tag{2}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the state vector, $\vec{\mathbf{q}} \in [\mathbb{R}^s]^{\text{dim}}$ is the state gradient, and $\vec{\mathbf{H}} = \vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \vec{\mathbf{q}})$ is the total flux consisting of inviscid ($\vec{\mathbf{F}}$) and viscous ($\vec{\mathbf{G}}$) contributions. The i^{th} spatial component of the viscous flux is linear in $\vec{\mathbf{q}}$, $\mathbf{G}_i = -\mathbf{K}_{ij}\mathbf{q}_j$ (summation implied on $j \in 1 \ldots \text{dim}$).

Weighting the above equations with appropriate test functions, integrating by parts, and introducing the interface variable $\widehat{\mathbf{u}}$, yields, for one element Ω_e of the mesh,

$$\int_{\Omega_e} \vec{\mathbf{v}}^T \cdot \vec{\mathbf{q}}\, d\Omega + \int_{\Omega_e} \nabla \cdot \vec{\mathbf{v}}^T \mathbf{u}\, d\Omega - \int_{\partial\Omega_e} \vec{\mathbf{v}}^T \cdot \vec{n}\, \widehat{\mathbf{u}}\, ds = 0 \qquad \forall \vec{\mathbf{v}} \in [\mathcal{V}]^{\text{dim}}, \tag{3}$$

$$\int_{\Omega_e} \mathbf{w}^T \frac{\partial \mathbf{u}}{\partial t}\, d\Omega - \int_{\Omega_e} \nabla\mathbf{w}^T \cdot \vec{\mathbf{H}}\, d\Omega + \int_{\partial\Omega_e} \mathbf{w}^T \widehat{\mathbf{H}} \cdot \vec{n}\, ds = 0 \qquad \forall \mathbf{w} \in \mathcal{V}, \tag{4}$$

$$\int_{\sigma_f} \boldsymbol{\mu}^T \left\{ \widehat{\mathbf{H}} \cdot \vec{n}\big|_L + \widehat{\mathbf{H}} \cdot \vec{n}\big|_R \right\} ds = 0 \qquad \forall \boldsymbol{\mu} \in \mathcal{M}, \tag{5}$$

where the spaces of test functions are polynomials of a certain degree on the elements and interfaces. The third equation weakly imposes the continuity of the flux across interfaces and is required to

close the system. The "one-sided" fluxes are defined as

$$\widehat{\mathbf{H}} \cdot \vec{n} \;=\; \vec{\mathbf{H}}(\hat{\mathbf{u}}, \vec{\mathbf{q}}) \cdot \vec{n} + \mathbf{S}(\hat{\mathbf{u}})(\mathbf{u} - \hat{\mathbf{u}}) \,, \tag{6}$$

where $\mathbf{S} \in \mathbb{R}^{s \times s}$ is a stabilization tensor which, in this work, is chosen to yield a Roe-like flux with a penalty jump term,

$$\mathbf{S} = \mathbf{R} \left| \mathbf{\Lambda} \right| \mathbf{L} + \frac{n_i \mathbf{K}_{ij} n_j}{\ell_{\text{visc}}}. \tag{7}$$

Here, ℓ_{visc} is an $\mathcal{O}(1)$ user-defined viscous length scale, and the matrices $\mathbf{R}, \mathbf{\Lambda}$, and $\mathbf{L}$ come from an eigen-decomposition of the convective flux Jacobian evaluated about $\hat{\mathbf{u}}$. The above one-sided fluxes and $\hat{\mathbf{u}}$ variables are defined on all interior faces, while a Roe (or analytical) convective flux is used on boundary faces. In the present work we also consider HDG viscous stabilization via the method of Bassi and Rebay (BR2),[32] with an element and face-specific scaling that accounts for element anisotropy but keeps the equivalent viscous length scale $\mathcal{O}(1)$.

Choosing bases for the trial/test spaces turns Eqns. 3–5 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

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta\vec{\mathbf{Q}} \\ \Delta\mathbf{U} \\ \Delta\mathbf{\Lambda} \end{bmatrix} + \begin{bmatrix} \vec{\mathbf{R}}^Q \\ \mathbf{R}^U \\ \mathbf{R}^\Lambda \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{0}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{8}$$

where $\vec{\mathbf{Q}}$, $\mathbf{U}$, and $\mathbf{\Lambda}$ are the discrete unknowns in the approximation of $\vec{\mathbf{q}}$, $\mathbf{u}$, and $\hat{\mathbf{u}}$, 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. Statically condensing out the element-interior states gives a smaller system,

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

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

II.B. Adjoint

We use discrete adjoint solutions for error estimation and mesh adaptation. The discrete adjoint is obtained by solving the transposed linear system, driven by the linearization of a scalar output of interest, J. For HDG, this system reads

$$\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}^Q \\ \mathbf{\Psi}^U \\ \mathbf{\Psi}^\Lambda \end{bmatrix} + \begin{bmatrix} \frac{\partial J}{\partial \vec{\mathbf{Q}}}^T \\ \frac{\partial J}{\partial \mathbf{U}}^T \\ \frac{\partial J}{\partial \mathbf{\Lambda}}^T \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{0}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{9}$$

Statically condensing the element-interior adjoints gives the smaller system,

$$\boldsymbol{\mathcal{K}}^T \mathbf{\Psi}^\Lambda + \left(\frac{\partial J}{\partial \mathbf{\Lambda}}^T - \mathbf{B}^T \mathbf{A}^{-T} \left[\frac{\partial J}{\partial \vec{\mathbf{Q}}}, \frac{\partial J}{\partial \mathbf{U}} \right]^T \right) = \mathbf{0},$$

where the matrix $\boldsymbol{\mathcal{K}}$ is the same as that resulting from the primal system.

II.C. Unsteady

The above primal and adjoint discretizations extend to implicit unsteady formulations. The unsteady term appears in Eqn. 4, and when discretized in time it contributes to the residual and the linearization via terms multiplied by the spatial mass matrix. In the primal case, we are then solving for the state at one time node (e.g. backwards-difference multi-step or implicit Runge-Kutta) or on one time slab (e.g. DG in time[33]). In the adjoint case, we are solving for the adjoint at the time node, and the transpose on the unsteady residual Jacobian matrix reverses the temporal dependence, so that the adjoint is marched backwards in time. For error estimation and mesh adaptation, we use a slab-based discontinuous Galerkin temporal solver ("DG in time") based on an approximate Newton iterative formulation, as described below. For runs without error estimation, we use diagonally-implicit Runge-Kutta (DIRK).

Choosing spatial basis functions for the trial/test spaces and keeping the unsteady term separate turns Eqns. 3–5 into the following nonlinear system of ordinary differential equations:

$$\begin{aligned} \vec{\mathbf{R}}^Q &= \vec{\mathbf{0}}, \\ \mathbf{M}^U \tfrac{d\mathbf{U}}{dt} + \mathbf{R}^U &= \mathbf{0}, \\ \mathbf{R}^\Lambda &= \mathbf{0}, \end{aligned} \tag{10}$$

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 Eqn. 10 can be written compactly as

$$\mathbf{M}\frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \tag{11}$$

where the general, non-invertible, mass matrix is

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

II.C.1. Diagonally-Implicit Runge Kutta

When time-marching Eqn. 11 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:

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

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. 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

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

II.C.2. DG in Time

In a discontinuous Galerkin temporal discretization we divide the temporal domain into time slabs. Consider one time slab of size Δt between t_0 and $t_1 = t_0 + \Delta t$. Let $\varphi^n(t), n = 1 \ldots r+1$ be a set of temporal basis functions for order r polynomial approximation in time. We expand the solution vector as

$$\mathbf{W}(t) = \sum_{n=1}^{r+1} \mathbf{W}^n \varphi^n(t). \tag{12}$$

Multiplying Eqn. 11 by test functions $\varphi^m(t)$ and integrating by parts from t_0 to t_1 yields $r+1$ unsteady residuals,

$$\bar{\mathbf{R}}^m \equiv a^{mn}\mathbf{M}\mathbf{W}^n - \varphi^m(t_0)\mathbf{M}\mathbf{W}^0 + \int_{t_0}^{t_1} \varphi^m \mathbf{R}(\mathbf{W})\, dt \quad = \quad \mathbf{0}, \tag{13}$$

where summation is implied on the repeated index m, $\mathbf{W}^0$ is the initial condition for this slab, i.e. the solution at the end of the previous slab, and

$$a^{mn} = -\int_{t_0}^{t_1} \frac{d\varphi^m}{dt}\varphi^n\, dt + \varphi^m\varphi^n\Big|_{t_1}. \tag{14}$$

Eqn. 13 constitutes a large coupled system for the unknowns within the time slab. To make the solution tractable, we implement an approximate iterative solution approach,[29,34] with a modification that handles the present case of a non-invertible $\mathbf{M}$.

The approximate iterative solver is a quasi-Newton method for Eqn. 13. We approximate the residual linearization as

$$\frac{\partial \bar{\mathbf{R}}^m}{\partial \mathbf{W}^n} = a^{mn}\mathbf{M} + \int_{t_0}^{t_1} \varphi^m \frac{\partial \mathbf{R}}{\partial \mathbf{W}}\varphi^n\, dt \approx a^{mn}\mathbf{M} + \frac{\partial \mathbf{R}}{\partial \mathbf{W}}\Big|_{t_{1/2}} \int_{t_0}^{t_1} \varphi^m\varphi^n\, dt = a^{mn}\mathbf{M} + b^{mn}\Delta t\mathbf{A}, \tag{15}$$

where we have assumed that the spatial residual Jacobian matrix, $\mathbf{A}$, is constant over the slab and equal to the value at the slab midpoint, $t_{1/2}$. The normalized temporal mass matrix is

$$b^{mn} = \frac{1}{\Delta t}\int_{t_0}^{t_1} \varphi^m\varphi^n\, dt. \tag{16}$$

With this approximation, the Newton update for Eqn. 13 reads

$$\left(a^{mn}\mathbf{M} + b^{mn}\Delta t\mathbf{A}\right)\Delta\mathbf{W}^n + \bar{\mathbf{R}}^m = \mathbf{0}. \tag{17}$$

However, Eqn. 17 is still a large linear system: the updates $\Delta\mathbf{W}^n$ for all slab unknowns are coupled. By appropriate choice of temporal basis functions we can make some off-diagonal terms of a^{mn} and/or b^{mn} zero, but we can never fully decouple the equations. We thus resort to an approximate solution of Eqn. 17. Previous approaches[29,34] began by multiplying the system by $\mathbf{M}^{-1}$, which is not possible in the HDG formulation since $\mathbf{M}$ is not invertible. Instead, we begin by multiplying Eqn. 17 by $(\Delta t\mathbf{A})^{-1}$, with the result

$$\underbrace{\left(a^{mn}\mathbf{B} + b^{mn}\mathbf{I}\right)}_{\mathbf{C}^{mn}}\Delta\mathbf{W}^n + \mathbf{F}^m \quad = \quad \mathbf{0}, \tag{18}$$

where $\mathbf{B} = \frac{1}{\Delta t}\mathbf{A}^{-1}\mathbf{M}$ and $\mathbf{F}^m = \frac{1}{\Delta t}\mathbf{A}^{-1}\bar{\mathbf{R}}^m$. Note that this multiplication by the inverse of $\mathbf{A}$ is necessary to create a system where the coefficient matrices, $\mathbf{C}^{mn}$ commute, as $\mathbf{M}$ and $\mathbf{A}$ do not.

For $r = 1$ and with Lagrange basis functions, $\varphi^1(t) = (t_1 - t)/\Delta t$, $\varphi^2(t) = (t - t_0)/\Delta t$, Eqn. 18 becomes

$$\begin{array}{rcrcrcl} \left[\frac{\mathbf{B}}{2} + \frac{1}{3}\mathbf{I}\right]\Delta\mathbf{W}^1 & + & \left[\frac{\mathbf{B}}{2} + \frac{1}{6}\mathbf{I}\right]\Delta\mathbf{W}^2 & + & \mathbf{F}^1 & = & \mathbf{0}, \\ \left[-\frac{\mathbf{B}}{2} + \frac{1}{6}\mathbf{I}\right]\Delta\mathbf{W}^1 & + & \left[\frac{\mathbf{B}}{2} + \frac{1}{3}\mathbf{I}\right]\Delta\mathbf{W}^2 & + & \mathbf{F}^2 & = & \mathbf{0}. \end{array} \tag{19}$$

Note, that these equations are still coupled. Formally solving the first one for $\Delta\mathbf{W}^1$ and substituting into the second one gives

$$\left[\mathbf{I} + 4\mathbf{B} + 6(\mathbf{B})^2\right]\Delta\mathbf{W}^2 + \left[6\mathbf{B} - 2\mathbf{I}\right]\mathbf{F}^1 + \left[6\mathbf{B} + 4\mathbf{I}\right]\mathbf{F}^2 = \mathbf{0}. \tag{20}$$

Solving Eqn. 20 is complicated by the presence of $(\mathbf{B})^2 = \mathbf{B}\mathbf{B}$, which requires forming or working with the square of the residual Jacobian matrix. We therefore make an enabling approximation:

$$\mathbf{I} + 4\mathbf{B} + 6(\mathbf{B})^2 \approx \mathbf{I} + 2\sqrt{6}\mathbf{B} + 6(\mathbf{B})^2 = \left[\mathbf{I} + \sqrt{6}\mathbf{B}\right]^2. \tag{21}$$

Note that this approximation becomes more valid for $\Delta t \to 0$ as then $(\mathbf{B})^2$ dominates $\mathbf{B}$. Using Eqn. 21 and introducing a temporary variable $\mathbf{Y}$, Eqn. 20 becomes a system,

$$\left[\mathbf{I} + \sqrt{6}\mathbf{B}\right]\mathbf{Y} + \left[6\mathbf{B} - 2\mathbf{I}\right]\mathbf{F}^1 + \left[6\mathbf{B} + 4\mathbf{I}\right]\mathbf{F}^2 = \mathbf{0}. \tag{22}$$

$$\left[\mathbf{I} + \sqrt{6}\mathbf{B}\right]\Delta\mathbf{W}^2 - \mathbf{Y} = \mathbf{0}. \tag{23}$$

Multiplying both equations by $\mathbf{A}$ and substituting the definitions of $\mathbf{B}$ and $\mathbf{F}^m$ gives

$$\left[\mathbf{A} + \frac{\sqrt{6}}{\Delta t}\mathbf{M}\right]\mathbf{Y} + \frac{1}{\Delta t}\left[4\mathbf{R}^2 - 2\mathbf{R}^1\right] + \frac{6\mathbf{M}\mathbf{A}^{-1}}{\Delta t^2}\left[\mathbf{R}^1 + \mathbf{R}^2\right] = \mathbf{0}, \tag{24}$$

$$\left[\mathbf{A} + \frac{\sqrt{6}}{\Delta t}\mathbf{M}\right]\Delta\mathbf{W}^2 - \mathbf{Y} = \mathbf{0}. \tag{25}$$

Solving this system requires two inversions with the matrix $\left[\mathbf{A} + \frac{\sqrt{6}}{\Delta t}\mathbf{M}\right]$ and one inversion with the matrix $\mathbf{A}$ (to form the forcing vector). However, each of these solves is of the same size and sparsity pattern as used in steady-state calculations.

With $\Delta\mathbf{W}^2$ known, we solve for $\Delta\mathbf{W}^1$ from Eqn. 19. The resulting system is (after multiplication by $\mathbf{A}$),

$$\left[\mathbf{A} + \frac{3\mathbf{M}}{2\Delta t}\right]\Delta\mathbf{W}^1 + \left[\frac{\mathbf{A}}{2} + \frac{3\mathbf{M}}{2\Delta t}\right]\Delta\mathbf{W}^2 + \frac{3}{\Delta t}\mathbf{R}^1 = \mathbf{0}. \tag{26}$$

We thus require one more linear solve to obtain $\Delta\mathbf{W}^1$. In summary, at each Newton iteration for computing an update $\Delta\mathbf{W}^n$ via Eqn. 17, we solve three equations: 24–26. These approximations, and the resulting iterates, turn out to be equivalent to that presented in previous work[29] when the mass matrix is invertible, as is the case with DG. Similar steps can be taken for $r = 2$ in Eqn. 18: formally solving the first equation for $\Delta\mathbf{W}^1$, substituting into the remaining equations, solving for $\Delta\mathbf{W}^2$, and substituting into the last equation to obtain one equation for $\Delta\mathbf{W}^3$, whence an approximate factorization is applied. The resulting sequence of steps again produces the same answers as the inverse mass-matrix method[29] but can be applied in the case of HDG.

III. Arbitrary Lagrangian-Eulerian Formulation

III.A. 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.[35] This transformation is illustrated graphically in Figure 2, and Table 1 defines key quantities.

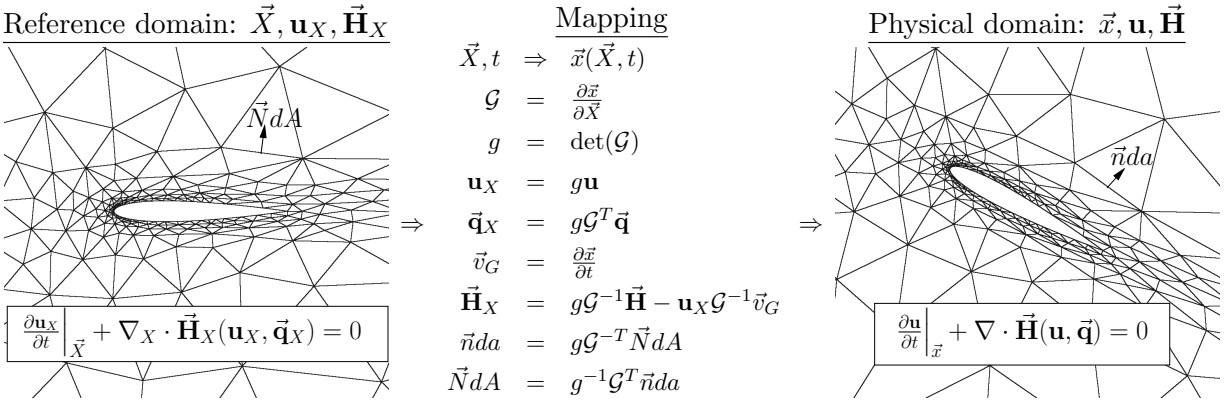


Figure 2. Summary of the mapping between reference and physical domains. The equations are solved on the reference domain, which remains fixed for all time. When denoting reference-domain quantities, we use a subscript X.

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			

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 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.[35] 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.[35] However, as these errors become smaller with higher-order approximation and adaptation, in this work we forgo a GCL, the implementation of which in an output-based adaptive setting is not trivial.[31]

III.B. Transformed Equations

To obtain the transformed conservation laws in reference space, we integrate the evolution PDE in Eqn. 2 over a time-varying volume $v(t)$,

$$\int_{v(t)} \frac{\partial \mathbf{u}}{\partial t} dv + \int_{\partial v(t)} \vec{\mathbf{H}} \cdot \vec{n}\, da = \mathbf{0}. \tag{27}$$

The two integrals in Eqn. 27 transform as

$$\int_{v(t)} \frac{\partial \mathbf{u}}{\partial t} = \frac{d}{dt}\int_{v(t)} \mathbf{u}\, dv - \int_{\partial v(t)} (\mathbf{u}\vec{v}_G)\cdot\vec{n}\, da = \int_V \frac{\partial (g\mathbf{u})}{\partial t}\, dV - \int_{\partial V} (g\mathbf{u}\mathcal{G}^{-1}\vec{v}_G)\cdot\vec{N}\, dA, \tag{28}$$

$$\int_{\partial v(t)} \vec{\mathbf{H}}\cdot\vec{n}\, da = \int_{\partial V} \vec{\mathbf{H}}\cdot(g\mathcal{G}^{-T}\vec{N})\, dA = \int_{\partial V} (g\mathcal{G}^{-1}\vec{\mathbf{H}})\cdot\vec{N}\, dA. \tag{29}$$

Substituting these expressions into (27) and applying the divergence theorem gives the PDE on the reference domain,

$$\left.\frac{\partial \mathbf{u}_X}{\partial t}\right|_{\vec{X}} + \nabla_X \cdot \vec{\mathbf{H}}_X(\mathbf{u}_X, \nabla_X \mathbf{u}_X) = \mathbf{0}, \tag{30}$$

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,

$$\vec{\mathbf{H}}_X = \vec{\mathbf{F}}_X + \vec{\mathbf{G}}_X, \qquad \vec{\mathbf{F}}_X = g\mathcal{G}^{-1}\vec{\mathbf{F}} + \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G, \qquad \vec{\mathbf{G}}_X = g\mathcal{G}^{-1}\vec{\mathbf{G}}. \tag{31}$$

To transform Eqn. 1, we write $\nabla\mathbf{u}$ in reference-space variables via the chain and product rules,

$$\nabla\mathbf{u} = \nabla_X(g^{-1}\mathbf{u}_X)\mathcal{G}^{-1} = g^{-1}\mathcal{G}^{-T}(\nabla_X\mathbf{u}_X - \mathbf{u}_X g^{-1}\nabla_X g), \tag{32}$$

Substituting into Eqn. 1, we have

$$\vec{\mathbf{q}} - g^{-1}\mathcal{G}^{-T}(\nabla_X\mathbf{u}_X - \mathbf{u}_X g^{-1}\nabla_X g) = \vec{\mathbf{0}}, \tag{33}$$

$$\underbrace{g\mathcal{G}^T\vec{\mathbf{q}}}_{\vec{\mathbf{q}}_X} - \nabla_X\mathbf{u}_X + \mathbf{u}_X g^{-1}\nabla_X g = \vec{\mathbf{0}}, \tag{34}$$

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

$$\vec{\mathbf{q}}_X - \nabla_X\mathbf{u}_X + \mathbf{u}_X g^{-1}\nabla_X g = \vec{\mathbf{0}}, \tag{35}$$

$$\frac{\partial \mathbf{u}_X}{\partial t} + \nabla_X\cdot\vec{\mathbf{H}}_X(\mathbf{u}_X, \nabla_X\mathbf{u}_X) = \mathbf{0}. \tag{36}$$

III.C. Implementation

An ALE solver for problems with mesh motion must operate on the reference-space equations in Eqns. 35–36. Equipping a DG or HDG code with such mesh motion capability should ideally not require a wholesale rewrite. Indeed, it is possible to "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. 35–36 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,

$$\vec{\mathbf{F}}_X = g\mathcal{G}^{-1}\vec{\mathbf{F}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G = g\mathcal{G}^{-1}\left(\vec{\mathbf{F}} - \mathbf{u}\vec{v}_G\right). \tag{37}$$

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

$$\vec{\mathbf{G}}_X = g\mathcal{G}^{-1}\vec{\mathbf{G}} = -g\mathcal{G}^{-1}\mathbf{K}\vec{\mathbf{q}} = -\underbrace{\mathcal{G}^{-1}\mathbf{K}\mathcal{G}^{-T}}_{\mathbf{K}_X}\vec{\mathbf{q}}_X. \tag{38}$$

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 Eqn. 38 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 viscous stabilization in Eqn. 7, which adds the following term to the reference space weak form,

$$\begin{aligned}
\int_{\partial\Omega_{X,e}} \mathbf{w}^T \frac{\vec{N}\cdot\mathbf{K}_X\cdot\vec{N}}{\ell_{\text{X,visc}}}(\mathbf{u}_X - \hat{\mathbf{u}}_X)dA &= \int_{\partial\Omega_{X,e}} \mathbf{w}^T \frac{\mathcal{G}^{-1}\mathbf{K}\mathcal{G}^{-T}\vec{N}}{\ell_{\text{X,visc}}}(g\mathbf{u} - g\hat{\mathbf{u}})\vec{N}dA \\
&= \int_{\partial\Omega_e} \mathbf{w}^T \frac{\mathcal{G}^{-1}\mathbf{K}\mathcal{G}^{-T}\vec{N}}{\ell_{\text{X,visc}}}(\mathbf{u} - \hat{\mathbf{u}})\mathcal{G}^T\vec{n}da \\
&= \int_{\partial\Omega_e} \mathbf{w}^T \frac{\vec{n}\cdot\mathbf{K}\cdot\vec{n}}{\ell_{\text{visc}}}(\mathbf{u} - \hat{\mathbf{u}})da
\end{aligned} \tag{39}$$

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 Eqn. 39 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 need 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

$$\vec{\mathbf{H}} \cdot \vec{n} da = \left(g^{-1} \mathcal{G} \vec{\mathbf{H}}_X \right) \cdot \left(g \mathcal{G}^{-1} \vec{N} \right) dA = \vec{\mathbf{H}}_X \cdot \vec{N} dA. \tag{40}$$

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, Eqn. 35. In the weak form this source term is evaluated in a straightforward manner via an element-interior integration.

III.D. Outputs

We consider scalar outputs evaluated by a boundary integral of a linear combination of the fluxes. In physical space, the general form of such outputs, which include force and moment-type outputs, is

$$J = \int_{\partial\Omega} \mathbf{o}^T \widehat{\mathbf{H}} \cdot \vec{n} \, da, \tag{41}$$

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 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. The expression in Eqn. 41 is written in physical-space variables, and this is convenient since no transformations are required from the physical flux calculations.

In problems involving mesh motion, a common output of interest is a power or work (time-integrated power). Such an output can still be cast in the form of Eqn. 41 by including the grid velocity, $\vec{v}_G$, in the output test function $\mathbf{o}$. Specifically, 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 output is

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

where $\vec{f}_{\text{surf}}$ is the surface stress vector, i.e. the momentum flux components. 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

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

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. Finally, we note that in the calculation of boundary fluxes, stabilization terms from the discretization are retained for adjoint-consistency reasons.[36]

IV. Output-Based Adaptation

IV.A. Error Estimation

The adjoint solution lets us estimate the numerical error in the output J using a standard adjoint-weighted residual approach[14,19] with two discretization spaces: coarse (H) and fine (h). The

fine space could be uniform mesh refinement or approximation order increase – in this work we increment the order in space and time. Note, for error estimation, we use a DG-in-time temporal discretization. The output error is calculated by injecting the coarse state $\mathbf{U}_H$ into the fine space, evaluating the residual, and weighting it by the fine-space adjoint,

$$\delta J \equiv J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \approx \sum_{k=1}^{N_k} \boldsymbol{\Psi}_h^{k\,T} \bar{\mathbf{R}}_h^k(\mathbf{U}_h^H). \tag{42}$$

where $\bar{\mathbf{R}}_h^k(\mathbf{U}_h^H)$ is the fine-space unsteady residual evaluated using the coarse state injected into the fine space. For HDG, we only use the $\mathbf{U}$ state in error estimation, neglecting contributions from $\vec{\mathbf{Q}}$ and $\boldsymbol{\Lambda}$ – these will be investigated in future work. The fine-space adjoint vectors can be solved exactly or approximately, e.g. through iterative smoothing or reconstruction. In the present results we solve for the fine-space adjoint exactly.

IV.B. Localization

Localized to space-time elements, the error estimate in (42) produces an adaptive indicator for mesh adaptation,

$$\delta J \approx \sum_{k=1}^{N_k} \sum_{e=1}^{N_e} \varepsilon_e^k, \tag{43}$$

where N_k is the number of time slabs, N_e is the number of elements, and the error contribution of a given space-time element (e,k) is, assuming a DG-in time temporal discretization order of r,

$$\varepsilon_e^k = \sum_{m=1}^{r+1} \left(-\boldsymbol{\Psi}_{he}^{km}\right)^T \bar{\mathbf{R}}_{he}^{km}\left(\mathbf{U}_h^H\right), \tag{44}$$

This is just the adjoint-weighted residual restricted to the space-time element (e,k), with a sum taken over the intra-slab temporal degrees of freedom. The error indicator is then taken as the absolute value of this elemental contribution to the output error,

$$\text{error indicator for element } e \text{ of time slab } k = \epsilon_e^k = \left|\varepsilon_e^k\right|. \tag{45}$$

IV.C. Adaptation

Many choices exist for both spatial and temporal adaptation mechanics in unsteady simulations. In the present work we restrict our attention to static spatial adaptation, in which mesh resolution remains fixed (in reference space) over the course of the unsteady simulation. This spatial adaptation is in the form of metric-based unstructured refinement[37,38] driven by the error indicator in Eqn. 45 marginalized over the time nodes with a space-time anisotropy measure. Specifically, a simple error/mesh-size relationship is employed: an a priori estimate of the error convergence rate (h^{p+1}) dictates the relationship between the error indicator and the isotropic mesh size request, up to a multiplicative constant that is set based on a desired growth-rate of the total space-time degrees of freedom.

In time, the temporal discretization is adapted by optimizing the size of each time slab using a one-dimensional mesh optimization approach.[31] The error estimate driving this optimization is the indicator in Eqn. 45 marginalized over the spatial elements with the complement of the space-time anisotropy measure. The space-time anisotropy measure[29] estimates the portion of the local error indicator attributable to the spatial and temporal discretizations, using projections of the unsteady adjoint to semi-refined approximation spaces.

V. Results

V.A. Euler Vortex Verification

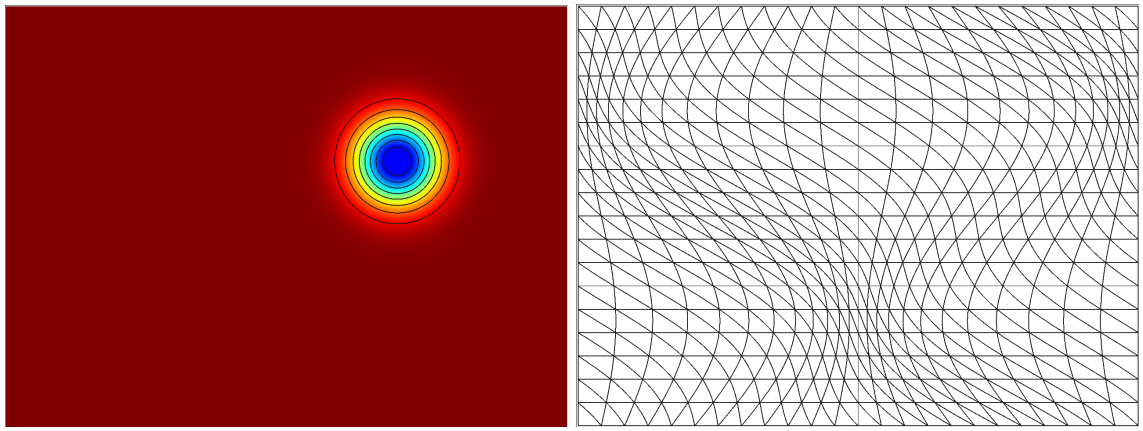
For the purpose of verification, we consider an analytical vortex solution to the Euler equations.[35] The state vector at point (x, y) and time t is obtained via

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

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



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

Figure 3. 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.

case using both DG and HDG, for various state approximation orders p and fourth-order DIRK time stepping, and we compute the L_2 norm of the state error at the final time, $t = 10$. Figure 4 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 negligible. 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.

V.B. Compressible Navier-Stokes Integral Output

To verify the HDG viscous discretization in the presence of mesh motion, we consider the same problem as in Section V.A but this time with nonzero constant viscosity, $\mu = 0.01$ (convenient

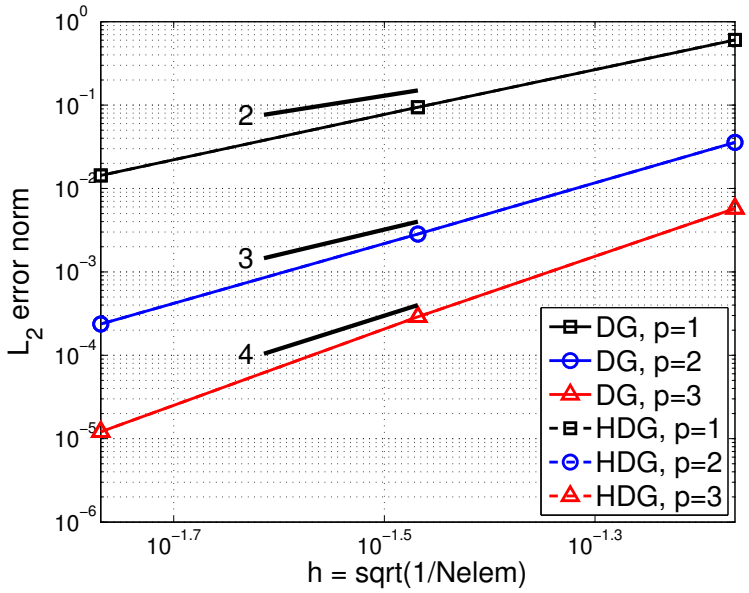


Figure 4. Euler vortex verification: convergence of the L_2 norm of the state error with uniform spatial refinement for DG and HDG (curves visually overlap). The temporal discretization is DIRK4 with a time step small enough to make the temporal error negligible.

units), and Prandtl number $Pr = 0.71$. In this case the given vortex solution is no longer exact, and instead of computing error norms in the state we initialize the flow with the $t = 0$ vortex profile and let the state evolve until $t = 10$. We then compare the results of DG and HDG by computing a scalar output,

$$J = \int_\Omega r^4 e^{-4r} p(\vec{x}, t = 10)\, dA, \qquad r \equiv \|\vec{x} - \vec{x}_0\|, \quad \vec{x}_0 = (15, 10), \tag{46}$$

where p is the pressure. Figure 5 shows the convergence of the error between J_{DG} and J_{HDG}, i.e. the outputs computed using the two discretizations, with spatial mesh refinement. Fourth-order DIRK is again used for the temporal discretization, with time step sufficiently small to keep the temporal errors negligible. Note that the errors decrease with uniform mesh refinement and order, indicating agreement between the results of the two discretizations.

V.C. NACA 0012 Airfoil in Pitch/Plunge Motion

We now present an output-based adaptive results for a NACA 0012 airfoil with zero trailing-edge thickness undergoing prescribed smooth plunge, $h(t)$, and pitch, $\theta(t)$, motions in the time range $0 \le t \le 1$ (we use convenient $O(1)$ units),

$$h(t) = \frac{1 - \cos \pi t}{2}, \quad \theta(t) = \frac{\pi}{6} \frac{1 - \cos 2\pi t}{2}.$$

The center of rotation for the pitch motion is at the airfoil 1/3 chord location. At $t = 0$, the state is initialized to a steady solution at zero angle of attack, Mach number of 0.2, and chord Reynolds number of $Re_c = 5000$ (constant viscosity). In our convenient units, the airfoil chord is 1 and the free-stream state is

$$\mathbf{u}_\infty = [\rho, \rho u, \rho v, \rho E]^T = \left[1, 1, 0, \frac{1}{2} + \frac{1}{M^2 \gamma(\gamma - 1)}\right]^T.$$

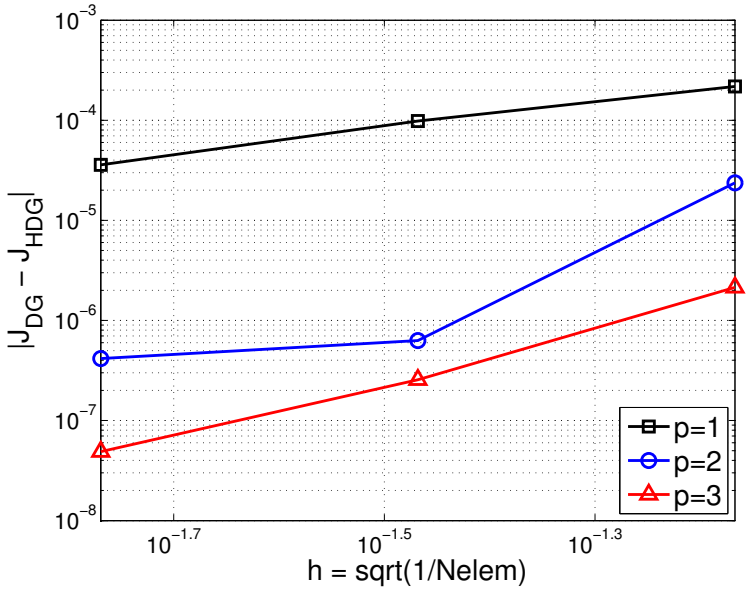


Figure 5. Compressible Navier-Stokes output verification: convergence of the difference between scalar outputs computed from DG and HDG discretizations, under uniform spatial mesh refinement. Time stepping is done via DIRK4.

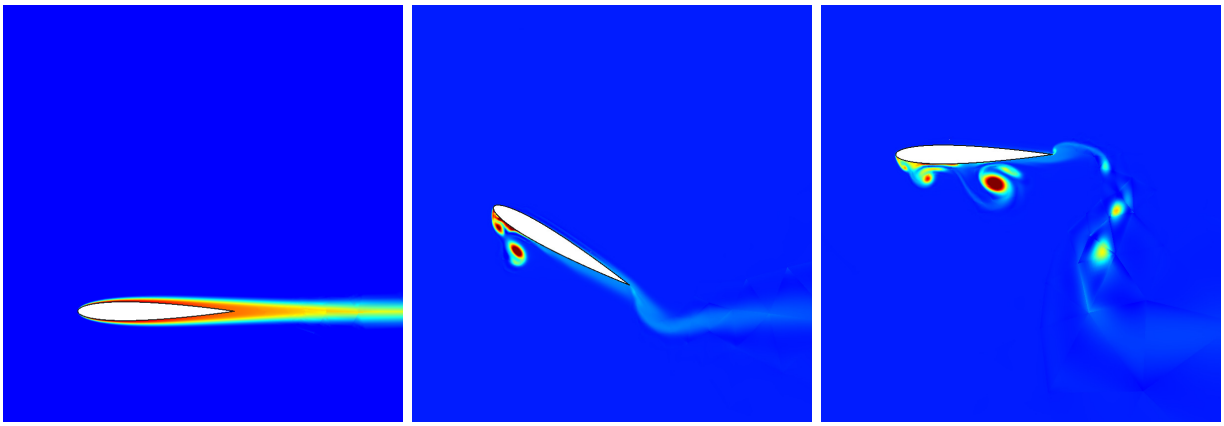


Figure 6. Entropy contours at $t = 0$, $t = 0.5$, $t = 1.0$ for a fine space-time mesh.

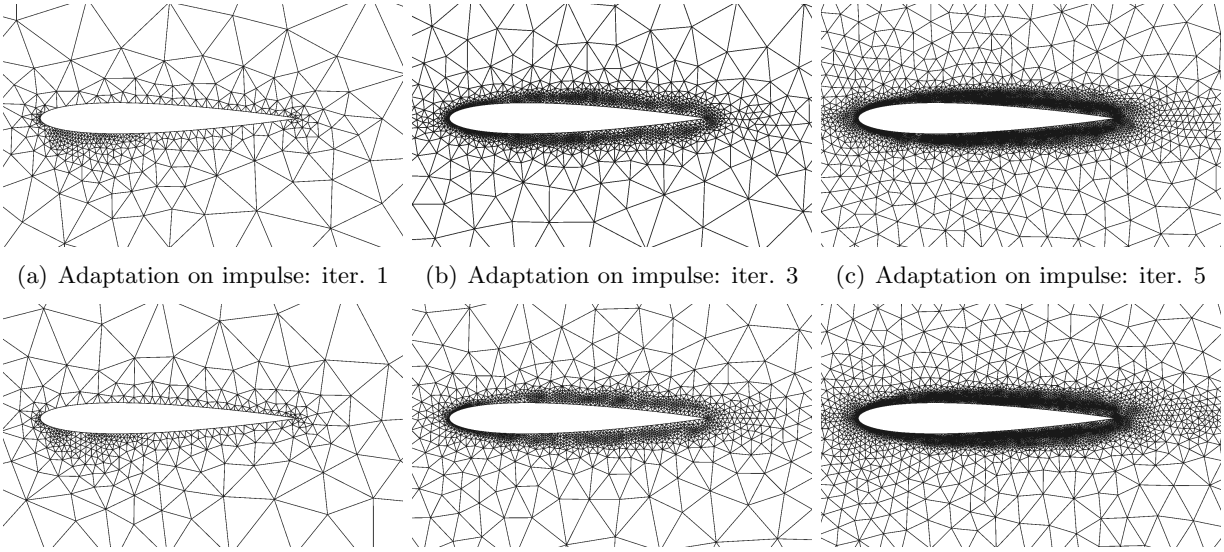
The Prandtl number is $Pr = 0.72$, and the ratio of specific heats is $\gamma = 1.4$. Figure 6 shows snapshots of the entropy contours for this unsteady simulation at three different times. Two outputs are of interest in this case: the vertical impulse imparted by the fluid on the airfoil, and the work done by the flow on the airfoil. These are defined as,

$$\text{Impulse} = \int_0^1 F_y \, dt = \int_0^1 \int_{\text{airfoil}} \hat{y} \cdot \vec{f}_{\text{surf}} \, ds \, dt, \tag{47}$$

$$\text{Work} = \int_0^1 \left[\vec{v}_0 \cdot \vec{F}_{\text{net}} + \vec{\omega} \cdot \vec{T}_{\text{net}} \right] dt = \int_0^1 \int_{\text{airfoil}} \vec{v}_G \cdot \vec{f}_{\text{surf}} \, ds \, dt, \tag{48}$$

where $\vec{v}_0$ is the plunge velocity, $\vec{\omega}$ is the pitch rate, F_y is the vertical force on the airfoil, $\vec{F}_{\text{net}}$ is the net force vector on the airfoil, and $\vec{T}_{\text{net}}$ is the net torque on the airfoil.

The initial spatial mesh contains 528 elements, and the initial temporal discretization consists of 40 equally-spaced time slabs. The spatial order is $p = 2$ and the temporal order for DG-in-time stepping is $r = 1$. Space-time adaptation proceeds using a prescribed factor of two growth rate in the degrees of freedom. Figure 7 shows the adapted spatial meshes. In this problem, most of



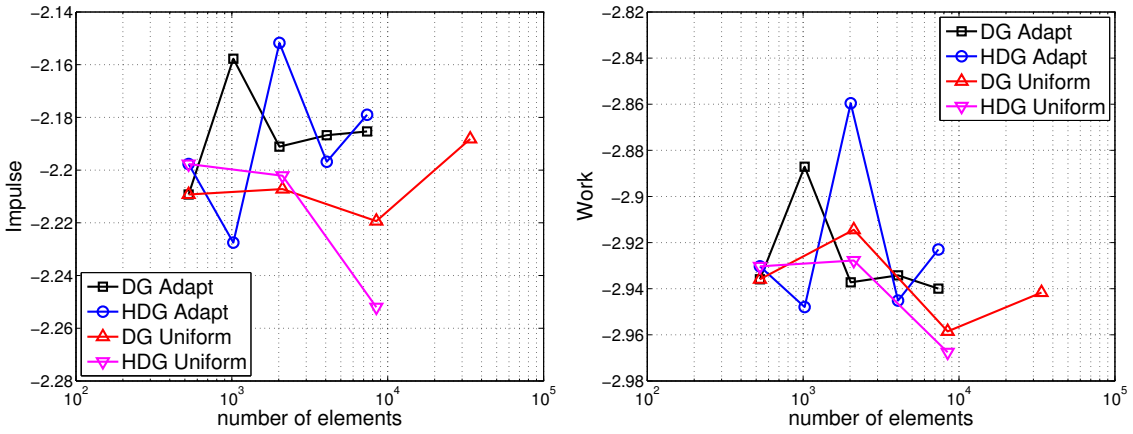
(a) Adaptation on impulse: iter. 1 (b) Adaptation on impulse: iter. 3 (c) Adaptation on impulse: iter. 5

(d) Adaptation on work: iter. 1 (e) Adaptation on work: iter. 3 (f) Adaptation on work: iter. 5

Figure 7. Reference-domain spatial meshes for unsteady output-based adaptation results using $p = 2$ spatial approximation order. Note the similarity in refinement patterns for the two unsteady outputs.

the initial error is due to lack of spatial refinement, and the adaptive scheme picks up on this by focusing the degree of freedom budget primarily on the spatial mesh. As such, the number of elements roughly doubles at every adaptive iteration. We note that the adapted meshes are quite similar between the two outputs used for adaptation. This is likely due to the fact that the two outputs, as integrals of weighted force components on the airfoil, are quite similar.

Due to similarities in the meshes between the two output drivers and between HDG and DG, we choose one adapted mesh sequence for presenting the convergence results: those generated by adapting on the work output with DG. Figure 8 shows the convergence of both outputs on this adapted mesh sequence, using DG and HDG spatial discretizations. For the impulse output, we see fairly good agreement between HDG and DG by the finer adapted meshes. HDG appears slightly more oscillatory but its results hover close to those of DG. For comparison, results are shown on



(a) Impulse output (b) Work output

Figure 8. Convergence of the impulse and work outputs versus number of spatial elements for DG and HDG discretizations on adapted and uniform mesh sequences.

meshes obtained from (quasi-)uniform refinement of the original mesh, via isotropic metric-based remeshing. This uniform refinement yields impulse values that are not relatively close to those of the adapted mesh sequence. In addition, the sequence of outputs does not exhibit convergence: the largest jumps in the output values occur between the finest meshes.

The work output in Figure 8(b) also shows fairly good agreement between HDG and DG. The HDG results are again more oscillatory, but the oscillations diminish on the fine meshes. As with the impulse output, the uniform refinement results still vary considerably even on the finest meshes.

VI. Conclusions

In this paper we present an extension of unsteady output-based adaptation to a hybrid discontinuous Galerkin (HDG) spatial discretization on deformable domains. Compared to traditional discontinuous Galerkin (DG), HDG offers computational advantages at high-order approximation due to static condensation of element-interior degrees of freedom. Such computational savings naturally 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.

We present details of the arbitrary Lagrangian-Eulerian formulation for HDG, including the transformation of the gradient equation. We also show how to modify a DG-in-time solver based on approximate factorization to the case of a non-invertible system mass matrix, as is the case for HDG. We apply unsteady output-based error estimation to the discretized problem in reference space and employ metric-based spatial mesh refinement in order to iteratively reduce the error.

Results show two verification studies that compare the performance of HDG and DG for problems with mesh deformation: one inviscid and one viscous. In both cases, HDG performs similarly to DG: differences diminish with mesh and order refinement. For a simulation of a plunging and pitching airfoil governed by the compressible Navier-Stokes equations, the two discretizations also show comparable results on a sequence of meshes generated using output-based adaptation. The adapted results converge more rapidly compared to uniform refinement as the adapted meshes target refinement in regions of high error to yield accurate outputs.

The present work considered spatial mesh resolution requirements for minimizing output error, while temporal errors were kept small by using small time steps. In future work we will consider concurrent temporal refinement and adaptation with other temporal discretizations.

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

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

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

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

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

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

[7]Cockburn, B. and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 173–261.

[8]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.

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

[10]Houston, P., Hartmann, R., and Süli, E., "Adaptive discontinuous Galerkin finite element methods for compressible fluid flows," *Numerical Methods for Fluid Dynamics VII, ICFD*, edited by M. Baines, 2001, pp. 347–353.

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

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

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

[14]Becker, R. and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.

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

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

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

[18]Nemec, M. and Aftosmis, M. J., "Error Estimation and Adpative Refinement for Embedded-Boundary Cartesian Meshes," AIAA Paper 2007-4187, 2007.

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

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

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

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

[23]Schmich, M. and Vexler, B., "Adaptivity with Dynamic Meshes for Space-Time Finite Element Discretizations of Parabolic Equations," *SIAM Journal on Scientific Computing*, Vol. 30, No. 1, 2008, pp. 369–393.

[24]Barth, T. J., "Space-Time Error Representation and Estimation in Navier-Stokes Calculations," *Complex Effects in Large Eddy Simulations*, edited by S. C. Kassinos, C. A. Langer, G. Iaccarino, and P. Moin, Springer Berlin Heidelberg, Lecture Notes in Computational Science and Engineering Vol 26, 2007, pp. 29–48.

[25]Besier, M. and Rannacher, R., "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*, Vol. 70, 2012, pp. 1139–1166.

[26]Mani, K. and Mavriplis, D. J., "Error Estimation and Adaptation for Functional Outputs in Time-Dependent Flow Problems," *Journal of Computational Physics*, Vol. 229, 2010, pp. 415–440.

[27]Belme, A., Dervieux, A., and Alauzet, F., "Error Estimation and Adaptation for Functional Outputs in Time-Dependent Flow Problems," *Journal of Computational Physics*, Vol. 231, 2012, pp. 6323–6348.

[28]Flynt, B. T. and Mavriplis, D. J., "Discrete Adjoint Based Adaptive Error Control in Unsteady Flow Problems," AIAA Paper 2012-0078, 2012.

[29]Fidkowski, K. J. and Luo, Y., "Output-based Space-Time Mesh Adaptation for the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 5753–5773.

[30]Fidkowski, K. J., "An Output-Based Dynamic Order Refinement Strategy for Unsteady Aerodynamics," AIAA Paper 2012-77, 2012.

[31]Kast, S. M. and Fidkowski, K. J., "Output-based Mesh Adaptation for High Order Navier-Stokes Simulations on Deformable Domains," *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494.

[32]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.

[33]Fidkowski, K. J., "Output error estimation strategies for discontinuous Galerkin discretizations of unsteady convection-dominated flows," *International Journal for Numerical Methods in Engineering*, Vol. 88, No. 12, 2011, pp. 1297–1322.

[34]Richter, T., "Discontinuous Galerkin as Time-Stepping Scheme for the Navier-Stokes Equations," *Fourth International Conference on High Performance Scientific Computing Modeling, Simulation and Optimization of Complex Processes*, Hanoi, Vietnam, 2009.

[35]Persson, P.-O., Bonet, J., and Peraire, J., "Discontinuous Galerkin Solution of the Navier-Stokes Equations on Deformable Domains," *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, 2009, pp. 1585–1595.

[36]Lu, J., *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.

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

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