High-Order Output-Based Adaptive Methods for Steady and Unsteady Aerodynamics

Krzysztof J. Fidkowski *
University of Michigan, USA

December 12, 2013

Contents

*Assistant Professor, Aerospace Engineering Department, kfid@umich.edu

1 Introduction

1.1 Adaptation Motivation

This compilation of notes presents an in-depth look at the development and application of output-based adaptive methods to large, complex problems in aerodynamics. We choose aerodynamics applications because the convection-dominated nature of these flows is such that outputs of engineering interest can be sensitive to small perturbations in the flow. The problem of generating adequate meshes is thus not very forgiving, as inadequate resolution in a seemingly uninteresting area of the domain can have repercussions in important areas that affect outputs of interest. It is in these situations that output-based adaptive methods, which specifically take into account such "propagation effects," can pay off, often by a margin that thankfully outweighs their nontrivial cost.

When discussing adaptation, we are concerned with minimizing errors from the numerical discretization, which includes effects of mesh size, polynomial order, and time step. Such error is inherent to computational simulations because of the finite nature of discrete numerical calculations. As illustrated in Figure 1, discretization error is not the only source of error in computation, but its management tends to be the most elusive. In practice, with limited resources, numerical errors are managed by practitioners who are knowledgeable about the assumptions and limitations of the models, and/or by best-practice guidelines for mesh generation. However, even very experienced practitioners cannot reliably quantify the error in a discrete approximation of a complex flowfield – see for example Figure 3. In addition, reliance on best-practice guidelines is an open-loop solution that leaves unchecked the possibility of large amounts of numerical error for computations on novel configurations.

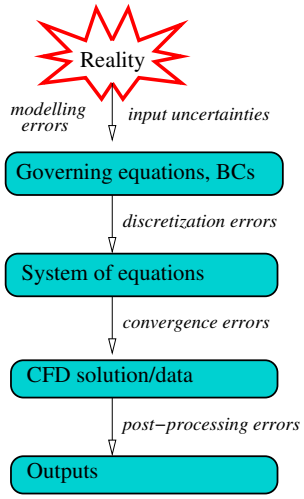


Figure 1: Various sources of error affecting computation.

Methods based on an *adjoint* analysis, where an adjoint associated with an output is the sensitivity of that output to residuals in the governing equations, help to close the loop in the control of numerical error via two ways. First, these methods provide output error estimates, which can be used as types of numerical "error bars" to determine if the engineering output has been computed to sufficient accuracy. Second, the error estimates can be localized, e.g. to elements of the computational mesh, in order to drive an adaptive method that reduces the error, as illustrated schematically in Figure 2. We view both of these uses of output-based analysis as improvements in the *robustness* of numerical simulations. Furthermore, note that the a posteriori nature of the error estimation and adaptation elicits a new paradigm for what constitutes a numerical solution – rather than treating properties of the discretization, such as the mesh, as givens, we propose to treat these properties as unknowns in the solution process.

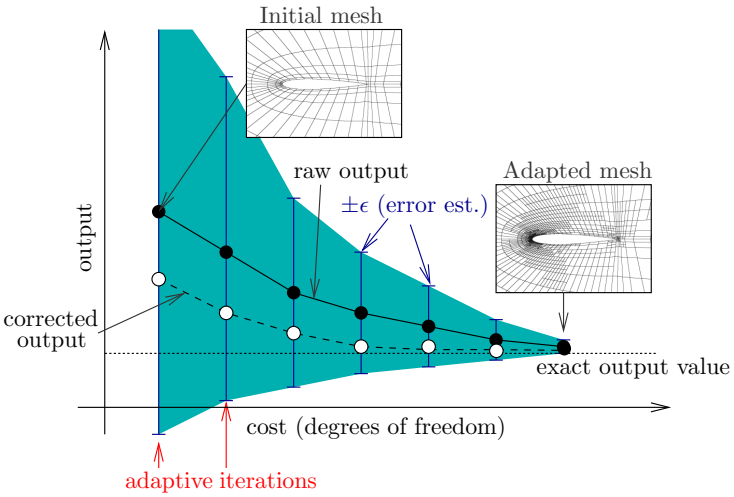


Figure 2: Typical adaptive convergence result obtained using output-based error estimates. The task of allocating degrees of freedom to regions of the domain is incorporated into the solver. Note, in many cases we can compute directional error estimates, i.e. corrections, and the resulting corrected outputs are illustrated by a dashed line. We can also often obtain conservative error estimates (though not strict bounds), and these are illustrated by the shaded regions at $\pm\epsilon$.

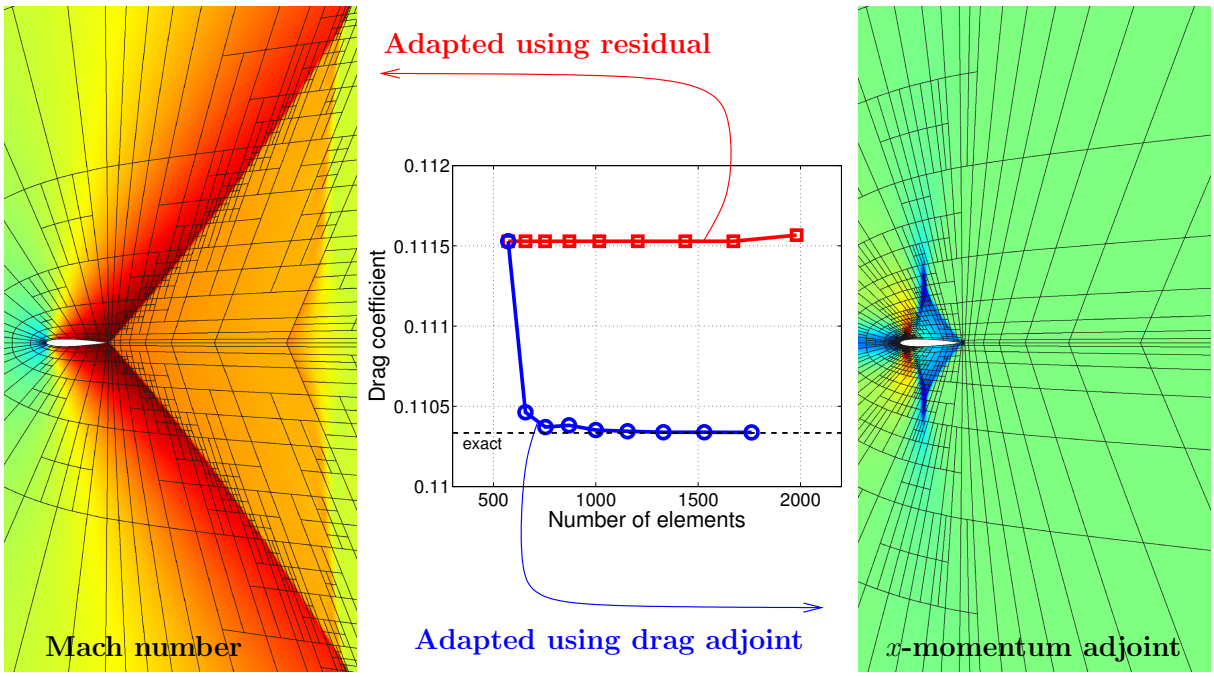


Figure 3: Two ways of adapting the mesh for a transonic, $M_\infty = 0.95$, Euler simulation around a NACA 0012 airfoil. The "fishtail" shock structure aft of the airfoil is a prominent feature, one targeted by a typical indicator such as an unweighted residual, which is actually not necessary for the prediction of drag. An indicator that weights the residual by the drag adjoint is not distracted by this feature and quickly hones in on the correct drag value.

1.2 Why "High Order"?

The title of these notes indicates that we employ a "high-order" discretization. In many scientific and engineering disciplines a debate continues on the merits and deficiencies of low-order versus high-order algorithms, in particular for Computational Fluid Dynamics (CFD). A recent example is the International Workshop on High-Order CFD Methods [108], in which "high-order" discretizations were compared against their "low-order" counterparts for a suite of over a dozen test cases. The quotation marks indicate that that demarcation of low/high order is also debatable. A typical answer is that the choice of order depends on the problem: smooth/discontinuous flows are efficiently resolved with high/low order methods. However, most practical applications of CFD have both smooth and singular features, the locations of which are generally not known a priori, so that neither low- nor high-order algorithms ubiquitously dominate.

This order debate masks a more important observation: without solution-based error estimation and adaptivity, neither low nor high-order methods are robust. Unquantified discretization errors can and do pollute results. Over-resolution, while less typical, is also problematic because it is inefficient: slow simulations mean fewer high-fidelity runs for uncertainty quantification or design optimization applications. Moreover, determining under- versus over-resolution can be extremely difficult for complex aerodynamic problems, and hence in these cases output-based methods are paramount for controlling numerical error and appropriately distributing resolution for efficient calculations.

In an adaptive setting, high order is applied selectively and becomes just another tool for efficient accuracy improvement. Other tools include mesh size and stretching optimization and mesh motion; one can imagine that more adaptive tools translate into a better use of degrees of freedom, and hence more efficient adaptation, as indicated in Figure 4. For our work, we focus on *hp*-refinement (refinement of both the mesh and

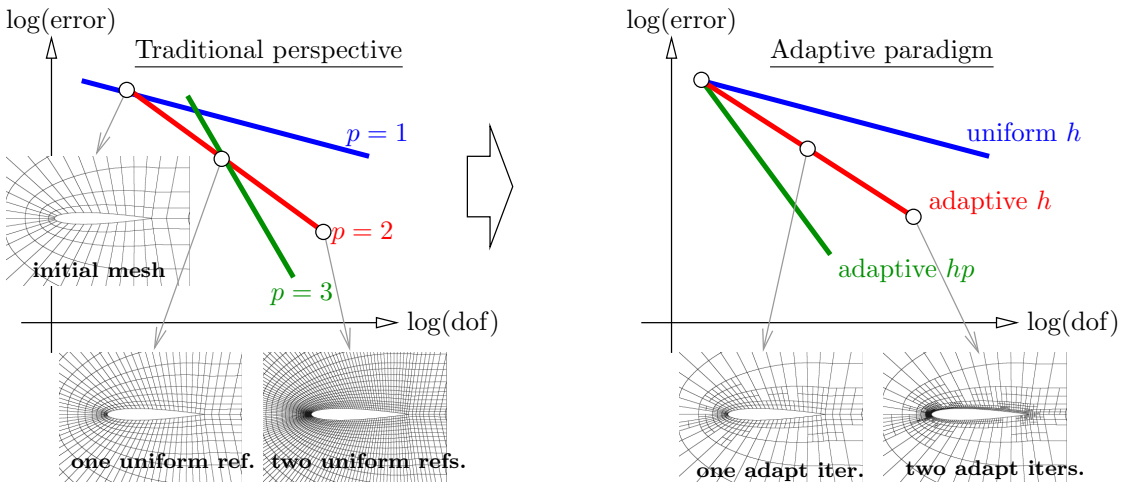


Figure 4: Schematic comparison of output convergence for uniform mesh refinement and for adaptive refinement. In aerodynamics, *hp*-refinement provides an important capability for addressing errors in flows with both smooth and discontinuous regions, specifically when targeting engineering outputs. Note that in the presence of singular features, the convergence rate of high-order approximations with uniform refinement (left) will generally be no better than that of low-order approximations, whereas this is not the case with *hp*-refinement.

approximation order), which is important because aerodynamic flows generally exhibit both smooth and discontinuous features. *hp*-refinement is then a big step in maximizing efficiency and robustness of CFD calculations, and the ability to locally enrich the order of the method is hence particularly important.

1.3 Background

We approach high order not from the point of view that these methods are categorically better than their low-order counterparts, but rather from the perspective that high order is just another refinement option for enrichment and optimization of a solution approximation space. Of course this means that we need to work with a discretization that admits high order, preferably in a locally-adaptive manner. Several such choices exist, yet we turn to finite elements for their built-in variational framework, and to discontinuous approximation spaces for convective stability. Previous works have demonstrated the realizability of high-order accuracy [**?** 37], error estimation [14, 57, 31], *hp*-adaptation [61], stable viscous discretization [9, 10, 3, 16, 85], and extension to variational space-time algorithms [5, 67, 105, 106, 65, 8, 73, 97, 36] using discontinuous Galerkin (DG) finite element methods.

DG methods for CFD have received a great deal of attention, with a common conclusion that while accurate, they are expensive. A class of hybrid methods, collectively termed hybrid discontinuous Galerkin (HDG), has been recently developed to address the expense of traditional DG [78, 77, 91]. In HDG, the element-interior approximation remains the same as in DG, but additional unknowns are introduced: the traces of the state on element interfaces. Stabilized fluxes couple interior DOFs to these traces only, so that interior DOFs can be statically condensed out of the linear system, leaving the trace DOFs, which scale with order to a one-lower dimension, as the only globally-coupled unknowns. Viscous discretizations are handled naturally via conversion to a first-order system with additional unknowns. We will apply output-based adaptation to both DG and HDG discretizations.

Output-based methods [88, 12, 54, 107, 75, 35] seek to quantify and minimize errors in scalar outputs of interest. Their robustness and efficiency arise from the fact that they specifically target for refinement those and only those areas of the computational domain that are important for predicting the output. They are well-suited for convection-dominated flows because through the use of adjoint solutions they properly account for error propagation effects, which can be troublesome for many heuristic adaptive indicators. Application to unsteady problems has been more limited [69, 70, 8, 73, 36, 33] due largely to implementation challenges and computational expense associated with the solution of a fine-space adjoint equation, especially for nonlinear problems.

2 Discretization

2.1 The Discontinuous Galerkin Method

2.1.1 Conservation Equations

Consider a conservation law given by the partial differential equation (PDE)[1].

$$\partial_t \mathbf{u} + \partial_i \mathbf{H}_i(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \tag{1}$$

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

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

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

where $\mathbf{F}_i \in \mathbb{R}^s$ is the i^{th} component of the inviscid/convective flux, $\mathbf{G}_i \in \mathbb{R}^s$ is the i^{th} component of the viscous flux, and $\mathbf{K}_{ij} \in \mathbb{R}^{s \times s}$ is the (i, j) component of the viscous diffusivity tensor. We will initially focus on steady problems, so that $\partial_t \mathbf{u} = \mathbf{0}$.

2.1.2 Solution Approximation

DG is a finite element method in which the state $\mathbf{u}$ is spatially approximated in functional form, using linear combinations of basis functions, usually polynomials, on each element. No continuity constraints are imposed on the approximations on adjacent elements. Denote by T_h the set of N_e elements in a non-overlapping tessellation of the domain Ω, as illustrated in Figure 5.

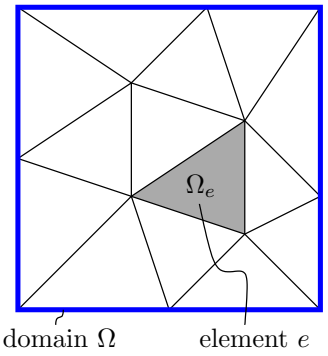


Figure 5: Partition of a square domain into 14 triangular elements.

$$\mathbf{u}_h(\vec{x}) \approx \sum_{e=1}^{N_e} \sum_{n=1}^{N_{p_e}} \mathbf{U}_{en} \phi_{en}(\vec{x}), \tag{4}$$

where

N_{p_e}	=	number of basis functions needed for an order p_e approximation
$\phi_{en}(\vec{x})$	=	n^{th} order p_e basis function on element e (zero on all other elements)
p_e	=	order of spatial basis functions on element e
N_e	=	number of elements
$\mathbf{U}_{en}$	=	vector of s coefficients on n^{th} basis function on element e

Formally, we can write that $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where

$$\mathcal{V}_h = \left\{ u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^{p_e} \ \forall \Omega_e \in T_h \right\},$$

and $\mathcal{P}^{p_e}$ denotes polynomials of order p_e on element e. [2] In many cases we use the same approximation order on all elements, so that $\forall e, p_e = p$.

[1] See Appendix A for the PDEs used in this work

[2] A caveat here is that for elements that are curved, the polynomial approximation is usually performed on a master reference element, so that following the reference-to-global mapping, the state approximation on curved elements is not strictly of order p_e.

2.1.3 Weak Form

We obtain a weak form of (1) by multiplying the PDE by test functions $\mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h$ and integrating by parts to couple elements via fluxes. The convective fluxes on element faces are handled via a traditional finite-volume (approximate) Riemann solver, but the diffusive treatment is trickier and requires stabilization. Multiple diffusion formulations exist [3], and we employ the second form of Bassi and Rebay (BR2) [9].

We can write the final semilinear weak form as

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h) = 0, \quad \forall \mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h, \tag{5}$$

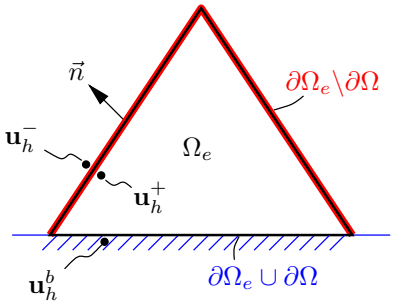
which, by linearity of the second argument, we can decompose into contributions from each element,

$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h) = \sum_{e=1}^{N_e} \mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) = 0, \quad \forall \mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h. \tag{6}$$

Integrating by parts and applying the BR2 diffusion treatment, we find that the semilinear form associated with each element is

$$\begin{aligned}
\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) \;=\; & \int_{\Omega_e} \mathbf{v}_h^T \partial_t \mathbf{u}_h \, d\Omega - \int_{\Omega_e} \partial_i \mathbf{v}_h^T \mathbf{H}_i \, d\Omega \\
& + \int_{\partial\Omega_e \setminus \partial\Omega} \mathbf{v}_h^{+T} \left(\widehat{\mathbf{F}} + \widehat{\mathbf{G}} \right) ds + \int_{\partial\Omega_e \cup \partial\Omega} \mathbf{v}_h^{+T} \left(\widehat{\mathbf{F}}^b + \widehat{\mathbf{G}}^b \right) ds \\
& - \int_{\partial\Omega_e \setminus \partial\Omega} \partial_i \mathbf{v}_h^{+T} \mathbf{K}_{ij}^{+} \left(\mathbf{u}_h^{+} - \widehat{\mathbf{u}}_h \right) n_j \, ds \\
& - \int_{\partial\Omega_e \cup \partial\Omega} \partial_i \mathbf{v}_h^{+T} \mathbf{K}_{ij}^{b} \left(\mathbf{u}_h^{+} - \mathbf{u}_h^{b} \right) n_j \, ds
\end{aligned} \tag{7}$$

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 two terms symmetrize the semilinear form for adjoint consistency. A schematic showing key quantities and the variables affecting the fluxes is given below. The unique state on an interior face is $\widehat{\mathbf{u}}_h = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$.



$$\begin{aligned}
\widehat{\mathbf{F}} &= \widehat{\mathbf{F}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}) \\
\widehat{\mathbf{G}} &= \widehat{\mathbf{G}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla\mathbf{u}_h^+, \nabla\mathbf{u}_h^-, \vec{n}) \\
\widehat{\mathbf{F}}^b &= \widehat{\mathbf{F}}^b(\mathbf{u}_h^b, \mathrm{BC}, \vec{n}) \\
\widehat{\mathbf{G}}^b &= \widehat{\mathbf{G}}^b(\mathbf{u}_h^b, \nabla\mathbf{u}_h^+, \mathrm{BC}, \vec{n})
\end{aligned}$$

In particular, on an interior face σ^f, the convective flux $\widehat{\mathbf{F}}$ is computed using the Roe approximate Riemann solver [93], and the BR2-stabilized viscous flux is

$$\widehat{\mathbf{G}} = \frac{1}{2} \left(\mathbf{G}_i^+ + \mathbf{G}_i^- \right) n_i^+ + \eta \frac{1}{2} \left(\boldsymbol{\delta}_i^+ + \boldsymbol{\delta}_i^- \right) n_i^+, \tag{8}$$

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 $\forall \boldsymbol{\tau}_{hi} \in \boldsymbol{\mathcal{V}}_h$

$$\int_{\Omega_e^+} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i^+ \, d\Omega + \int_{\Omega_e^-} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i^- \, d\Omega = \int_{\sigma^f} \frac{1}{2} \left(\boldsymbol{\tau}_{hi}^{+T} \boldsymbol{K}_{ij}^+ + \boldsymbol{\tau}_{hi}^{-T} \boldsymbol{K}_{ij}^- \right) (\mathbf{u}_h^+ - \mathbf{u}_h^-) n_j^+ \, ds. \quad (9)$$

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.

On a boundary face σ^b, fluxes are typically computed directly from the boundary state, $\mathbf{u}^b$, which is a function (projection) of the interior state and the boundary-condition data, $\mathbf{u}_h^b = \mathbf{u}_h^b(\mathbf{u}_h^+, \text{BC})$. 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

$$\widehat{\mathbf{G}}^b = \Pi_G^{\text{BC}} \left[\mathbf{G}_i(\mathbf{u}_h^b, \nabla \mathbf{u}_h^+) n_i + \eta \boldsymbol{\delta}_i n_i \right], \quad (10)$$

where the auxiliary variable $\vec{\boldsymbol{\delta}} \in [\boldsymbol{\mathcal{V}}_h]^d$ has support on the element adjacent to the boundary face σ^b and is obtained by solving $\forall \boldsymbol{\tau}_{hi} \in \boldsymbol{\mathcal{V}}_h$

$$\int_{\Omega_e} \boldsymbol{\tau}_{hi}^T \boldsymbol{\delta}_i \, d\Omega \;\; = \;\; \int_{\sigma^b} \boldsymbol{\tau}_{hi}^{+T} \boldsymbol{K}_{ij}^b (\mathbf{u}_h^+ - \mathbf{u}_h^b) n_j^+ \, ds. \quad (11)$$

The projection Π_G^{BC} in (10) incorporates boundary conditions on the viscous flux, such as a prescribed heat flux in the compressible Navier-Stokes equations.

2.1.4 Discrete System

For efficient implementation, the various contributions from all elements to the final semi-linear form in (5) are computed in three logical loops: one loop over all element interiors, one loop over all interior faces, and one loop over all boundary faces. When computing these contributions, we choose as test functions the trial basis functions introduced in (4), ϕ_{en}, where

$$\text{span}\,\{\phi_{en}\} = \mathcal{V}_h. \quad (12)$$

Recall that e is the element number and n indexes the polynomial basis functions inside element e. Each state equation is tested with the same set of functions, which means that we can define a size-s residual vector for the n^{th} test function in element e by

$$\mathbf{R}_{en} \equiv \{\mathcal{R}_h(\mathbf{u}_h, \phi_{en}\mathbf{e}_r)\}_{r=1\ldots s} \in \mathbb{R}^s, \quad (13)$$

where $\mathbf{e}_r \in \mathbb{R}^s$, $r = 1 \ldots s$, is a vector of all zeros except a 1 in position r. We now use the convention that dropping a subscript means considering the set of values over the entire range of that subscript. So $\mathbf{R}_e$ is the set of residuals over all states and basis functions inside element e, and $\mathbf{R}$ is the set of all residuals for all elements in the domain. Using the same convention for the state, $\mathbf{U}$, we can write the discrete system of equations (i.e. residuals) compactly as

$$\mathbf{R}(\mathbf{U}) = \mathbf{0}. \quad (14)$$

Note that both $\mathbf{U}$ and $\mathbf{R}$ lie in $\mathbb{R}^N$, where the total number of degrees of freedom including equation states is

$$N = \sum_{e=1}^{N_e} N_{p_e} s. \tag{15}$$

When considering different discretization spaces, we will append a subscript h or H to the variables $\mathbf{R}$, $\mathbf{U}$, and N.

In practice, we store the state and residual vectors unrolled, as illustrated in Figure 6. Finally, the number of basis functions per element depends on the approximation space

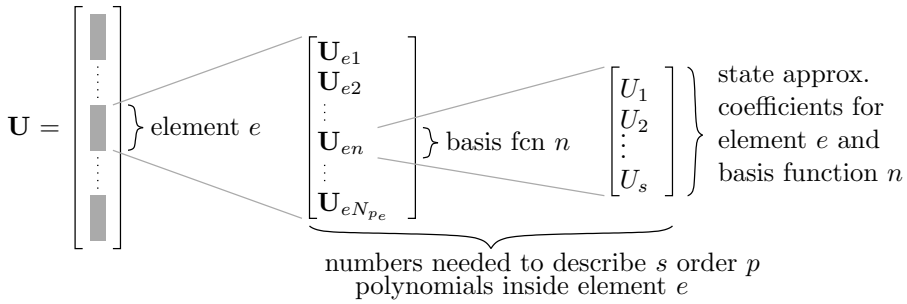


Figure 6: Unrolled storage of the state vector $\mathbf{U} \in \mathbb{R}^N$.

and order p. We consider two approximation spaces that are given by the span of monomials in reference space coordinates ξ_i, $\Pi_{i=1}^d \xi_i^{p_i}$: a full-order space in which $\sum_{i=1}^d p_i \leq p$, and a tensor-product space in which $\forall i, p_i \leq p$. The resulting dimensions of these spaces are

$$\text{full-order basis set: } N_p = \binom{p}{d} \qquad \text{tensor-product basis set: } N_p = (p+1)^d.$$

2.1.5 Nonlinear Solver

To solve (14), we use a preconditioned Newton-Krylov method augmented with pseudo-transient continuation. Starting from an initial guess, usually the free-stream condition, and a conservative time step set using a Courant-Friedrichs-Lewy (CFL) number of approximately 1, the Newton solver iterates until steady state. The linear system at each Newton iteration is solved using an element-line preconditioned General Minimal Residual Krylov subspace method [95, 94]. As successful Newton updates are taken, the time step is increased according to an exponential growth formula for the CFL number. Details on the solver, including the incorporation of physical constraints on some of the variables, can be found in [22].

2.2 Discrete Adjoint

Suppose that we are interested in a scalar output computed from the solution to our PDE. Using the discrete state vector, we write

$$\text{output } J \;=\; J(\mathbf{U}). \tag{16}$$

The discrete adjoint, $\boldsymbol{\Psi} \in \mathbb{R}^N$, is a vector of sensitivities of the output to the N residuals. That is, each entry of the adjoint tells us the effect that a perturbation in the same entry in the residual vector would have on the output J. One common source of residual perturbations is changes in input parameters for a problem, and so we ground the adjoint presentation in the context of a local sensitivity analysis.

2.2.1 Local Sensitivity Analysis

Consider a situation in which N_μ parameters, $\boldsymbol{\mu} \in \mathbb{R}^{N_\mu}$, affect the PDE in (14). For example, in aerodynamics, parameters could be used to set boundary conditions or the geometry. We can then write the following chain of dependence,

$$\underbrace{\boldsymbol{\mu}}_{\text{inputs} \in \mathbb{R}^{N_\mu}} \rightarrow \underbrace{\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) = 0}_{N \text{ equations}} \rightarrow \underbrace{\mathbf{U}}_{\text{state} \in \mathbb{R}^N} \rightarrow \underbrace{J(\mathbf{U})}_{\text{output (scalar)}} . \tag{17}$$

We are interested in how J changes (locally for nonlinear problems) with $\boldsymbol{\mu}$,

$$\frac{dJ}{d\boldsymbol{\mu}} \in \mathbb{R}^{1 \times N_\mu} = N_\mu \text{ sensitivities.} \tag{18}$$

Note, if J depends directly on $\boldsymbol{\mu}$ we would add $\frac{\partial J}{\partial \boldsymbol{\mu}}$ to the above sensitivity, but for clarity of presentation we consider only the case when $J = J(\mathbf{U})$. Several options exist for computing these sensitivities. Two direct ones are finite differencing, in which the input parameters are perturbed one at a time, and forward linearization, in which the sequence of operations in (17) is linearized. Both of these become expensive when N_μ is moderate or large, because of the need to re-solve the system $\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) = 0$ for each parameter. A third choice is the adjoint approach, which requires an inexpensive residual perturbation calculation followed by an adjoint weighting to compute the effect on the output. That is, we write

$$\frac{dJ}{d\boldsymbol{\mu}} = \boldsymbol{\Psi}^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}. \tag{19}$$

This approach is efficient for computing a large number of sensitivities for one output, as the cost is one residual perturbation calculation and one vector product per sensitivity.

The central idea in the adjoint approach is that we do not need to solve the forward problem each time we want a sensitivity. Suppose that for a given $\boldsymbol{\mu}$ we solve our discrete system and find $\mathbf{U}$ such that $\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) = \mathbf{0}$. Now perturb $\boldsymbol{\mu} \rightarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}$ ... what is the effect on J? We can re-solve for a perturbed state, but this is expensive. Instead, we can separate the effects of $\boldsymbol{\mu}$ on $\mathbf{R}$, and $\mathbf{R}$ on J, as illustrated on the right. The adjoint method precomputes the effect of $\mathbf{R}$ on J, which is the expensive step. The resulting N sensitivities are stored in the vector $\boldsymbol{\Psi}$.

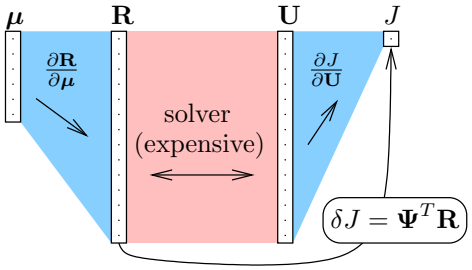


Figure 7: Bypassing the forward solve via an adjoint approach to sensitivity calculation.

2.2.2 The Adjoint System

To derive an equation for the adjoint, we consider the chain of operations we would take in computing the sensitivities via a direct approach. In the following steps, we assume small perturbations.

1. Input $\quad \boldsymbol{\mu} \rightarrow \boldsymbol{\mu} + \delta\boldsymbol{\mu}$
2. Residual $\quad \mathbf{R}(\mathbf{U}, \boldsymbol{\mu} + \delta\boldsymbol{\mu}) = \delta\mathbf{R} \neq 0 \quad \rightarrow \quad \mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\Big|_{\mathbf{U},\boldsymbol{\mu}} \delta\boldsymbol{\mu} = \delta\mathbf{R}$
3. State $\quad \mathbf{R}(\mathbf{U} + \delta\mathbf{U}, \boldsymbol{\mu} + \delta\boldsymbol{\mu}) = 0 \quad \rightarrow \quad \mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\Big|_{\mathbf{U},\boldsymbol{\mu}} \delta\boldsymbol{\mu} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U},\boldsymbol{\mu}} \delta\mathbf{U} = 0$
4. Output $\quad J(\mathbf{U} + \delta\mathbf{U}) = J(\mathbf{U}) + \delta J \quad \rightarrow \quad \delta J = \frac{\partial J}{\partial \mathbf{U}} \delta\mathbf{U}$

Subtracting step 2 from step 3, we obtain

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\Big|_{\mathbf{U},\boldsymbol{\mu}} \delta\mathbf{U} = -\delta\mathbf{R} \quad \Rightarrow \quad \delta\mathbf{U} = -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1} \delta\mathbf{R}. \tag{20}$$

Combining this result with the output linearization in step 4 gives the output perturbation in terms of the residual perturbation,

$$\delta J = \frac{\partial J}{\partial \mathbf{U}} \delta\mathbf{U} = \underbrace{-\frac{\partial J}{\partial \mathbf{U}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1}}_{\boldsymbol{\Psi}^T \in \mathbb{R}^N} \delta\mathbf{R}. \tag{21}$$

Taking the transpose of the equation $\boldsymbol{\Psi}^T = -\frac{\partial J}{\partial \mathbf{U}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1}$ and moving everything to the left-hand side gives the *adjoint equation*,

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)^T \boldsymbol{\Psi} + \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T = 0. \tag{22}$$

The n^{th} component of $\boldsymbol{\Psi}$ is the sensitivity of J to changes in the n^{th} residual.

Since $\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) = 0$, from step 2 above we have $\delta\mathbf{R} = \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\Big|_{\mathbf{U},\boldsymbol{\mu}} \delta\boldsymbol{\mu}$ and (21) becomes

$$\delta J = \boldsymbol{\Psi}^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\Big|_{\mathbf{U},\boldsymbol{\mu}} \delta\boldsymbol{\mu} \quad \Rightarrow \quad \frac{dJ}{d\boldsymbol{\mu}} = \boldsymbol{\Psi}^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}\Big|_{\mathbf{U},\boldsymbol{\mu}}. \tag{23}$$

Therefore, once we have $\boldsymbol{\Psi}$, no more solves are required for new sensitivities for the same output. Note that the calculation of $\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}$ is typically very cheap compared to a forward solve.

Although we have presented the adjoint in the context of a parameter sensitivity analysis, we will see in the next section that residual perturbations also arise when discrete solutions are viewed from an enriched space. This will be the motivation for using adjoint solutions in output error estimation.

2.2.3 Adjoint Consistency

The solution to (22) is a *discrete* adjoint, which at the simplest level we can think of as a vector of N numbers. However, the adjoint also has a continuous counterpart, call it $\boldsymbol{\psi}(\vec{x})$,

and we can think of the N numbers in $\mathbf{\Psi}$ as expansion coefficients in an approximation of $\boldsymbol{\psi}$ using the same basis functions used for the primal problem. The accuracy of this approximation is of interest for various reasons, including error estimation.

Suppose that the exact primal solution, $\mathbf{u} \in \boldsymbol{\mathcal{V}}$, satisfies

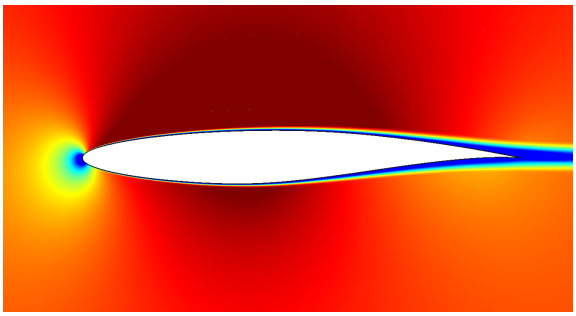
$$\mathcal{R}(\mathbf{u}, \mathbf{v}) = 0, \qquad \forall \mathbf{v} \in \boldsymbol{\mathcal{V}}, \tag{24}$$

for an appropriately defined space $\boldsymbol{\mathcal{V}}$. The exact adjoint $\boldsymbol{\psi} \in \boldsymbol{\mathcal{V}}$ then satisfies

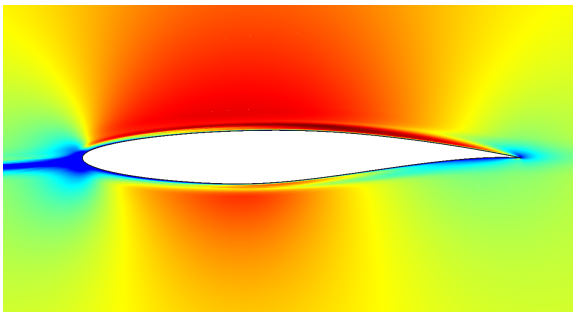
$$\mathcal{R}'[\mathbf{u}](\mathbf{v}, \boldsymbol{\psi}) + \mathcal{J}'[\mathbf{u}](\mathbf{v}) = 0, \qquad \forall \mathbf{v} \in \boldsymbol{\mathcal{V}}, \tag{25}$$

where the primes denote Fréchét linearization about the arguments in square brackets, and $\mathcal{R}$ and $\mathcal{J}$ are the continuous versions of the semilinear form and output functional, respectively. While we have assumed that both $\mathbf{u}$ and $\boldsymbol{\psi}$ lie in $\boldsymbol{\mathcal{V}}$, this may not always be the case [68].

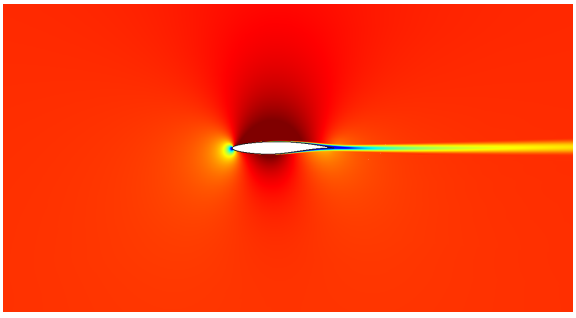
The exact adjoint can be regarded as a Green's function that relates source perturbations in the original partial differential equation to perturbations in the output[44, 46]. A sample adjoint solution is shown in Figure 8 for Reynolds-averaged compressible flow over an airfoil. While the adjoint solution often shares qualitative characteristics similar



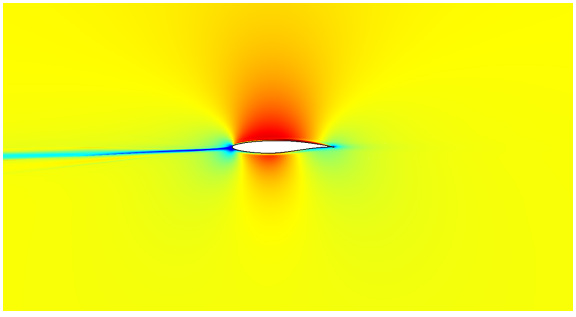
8.1: x-momentum state (near view)



8.2: x-momentum adjoint (near view)



8.3: x-momentum state (far view)



8.4: x-momentum adjoint (far view)

Figure 8: Comparison of the primal solution (x-momentum component) and the adjoint solution (conservation of x-momentum equation component) for a drag output in Reynolds-averaged turbulent flow over an RAE 2822 airfoil (see Section 6.8.2). The color scales are clipped to show the interesting features of each quantity – in the adjoint plots, yellow is near zero.

to the primal, such as the presence of a boundary layer in a high-Reynolds number flow, it also shows marked differences, such as the "wake reversal" seen in the far-field view in

Figure 8. In this case, the output is drag, and upstream of the airfoil, residual perturbations on/near the stagnation streamline (the flow that is going to hit or come closest to the airfoil) will have a larger magnitude impact on the drag than residual perturbations elsewhere; hence we see an adjoint "reversed" wake telling us that there are large sensitivities to perturbations in front of the airfoil. The figure shown tells us that this is the case for residual perturbations in the conservation of x-momentum equations, but plots of the other adjoint components show similar behavior.

The adjoint field depicted in Figure 8 is the discrete adjoint solution on a fine mesh. It can only be regarded as a faithful representation of the exact adjoint if the discretization is in some manner consistent with the exact adjoint problem. *Primal consistency* in the variational problem requires that the exact solution $\mathbf{u}$ satisfy the discrete variational statement,

$$\mathcal{R}_h(\mathbf{u}, \mathbf{v}) = 0, \qquad \forall \mathbf{v} \in \boldsymbol{\mathcal{W}}_h, \tag{26}$$

where $\boldsymbol{\mathcal{W}}_h = \boldsymbol{\mathcal{V}}_h + \boldsymbol{\mathcal{V}} = \{\mathbf{h} = \mathbf{f} + \mathbf{g} : \mathbf{f} \in \boldsymbol{\mathcal{V}}_h, \mathbf{g} \in \boldsymbol{\mathcal{V}}\}$. Similarly, the combination of the discrete semi-linear form $\mathcal{R}_h$ and the functional $\mathcal{J}_h$ is said to be *adjoint consistent* if [68, 53, 80]

$$\mathcal{R}'_h[\mathbf{u}](\mathbf{v}, \boldsymbol{\psi}) + \mathcal{J}'_h[\mathbf{u}](\mathbf{v}) = 0, \qquad \forall \mathbf{v} \in \boldsymbol{\mathcal{W}}_h. \tag{27}$$

Discretizations that are not adjoint consistent may still be *asymptotically adjoint consistent* if Eq. 27 holds in the limit $h \to 0$, by which we mean the limit of uniformly increasing resolution, over suitably normalized $\mathbf{v} \in \boldsymbol{\mathcal{W}}_h$. For non-variational discretizations, the definition of consistency must involve an approximation operator to map exact solutions into discrete spaces [27].

Adjoint consistency has an impact on the convergence of not only the adjoint approximation but also the primal approximation[3, 51, 45, 50, 68, 53, 80]. In error estimation, an adjoint-inconsistent discretization can lead to irregular or oscillatory adjoint solutions that pollute the error estimate with noise and lead to adaptation in incorrect areas [68]. Enforcing adjoint consistency imposes restrictions on the output definition and on the interior and boundary discretizations that enter into the semi-linear form. These restrictions have been studied by several authors in the context of the discontinuous Galerkin method [3, 68, 53]. In general, discretizations that are found to be adjoint inconsistent can often be made adjoint consistent by adding terms to either the semi-linear form or the output functional.

2.3 Examples

2.3.1 Method of Manufactured Solutions

A natural question after implementing any discretization is: is the implementation correct? One can use analytical solutions to specific problems to verify parts of the implementation, but it is very difficult to find analytical test cases for the Navier-Stokes equations, especially with Reynolds-averaged turbulence, that test all or most components of the implementation. Another option is to use the method of *manufactured solutions*. True to its name, this method lets us "make up" a solution,

$$\mathbf{u}(\vec{x}) = \mathbf{u}^{\mathrm{MS}}(\vec{x}) = \text{chosen by the user.} \tag{28}$$

Now, a made-up solution such as this will generally not satisfy the PDE in (1). That is, substituting $\mathbf{u}^{\text{MS}}$ into (1) will result in a remainder of

$$\mathbf{s}^{\text{MS}} \equiv \partial_t \mathbf{u}^{\text{MS}} + \partial_i \mathbf{H}_i(\mathbf{u}^{\text{MS}}, \nabla \mathbf{u}^{\text{MS}}). \tag{29}$$

If we take this remainder and treat it as a (negative) source term, we can obtain a modified PDE that $\mathbf{u}^{\text{MS}}$ *does* satisfy,

$$\partial_t \mathbf{u}^{\text{MS}} + \partial_i \mathbf{H}_i(\mathbf{u}^{\text{MS}}, \nabla \mathbf{u}^{\text{MS}}) - \mathbf{s}^{\text{MS}} = \mathbf{0}. \tag{30}$$

So, to test our discretization of the PDE, we have to discretize an additional source term, $\mathbf{s}^{\text{MS}}$. Once we have this, we can run a simulation on an arbitrary domain with Dirichlet boundary conditions, $\mathbf{u}^{\text{MS}}(\vec{x})$, and test how close the resulting solutions are to the chosen exact solution. Based on interpolation theory, we expect the error between the discrete solution and $\mathbf{u}^{\text{MS}}$ to converge as h^{p+1} for a characteristic mesh size h (see Section 3.5). Note, a false alarm is possible if the original PDE discretization is correct and we botch the source term discretization, but source terms are typically easy to discretize so the risk of this happening is low.

One simple manufactured solution for the compressible Navier-Stokes equations is sinusoidally varying density, velocity, and pressure fields, given by

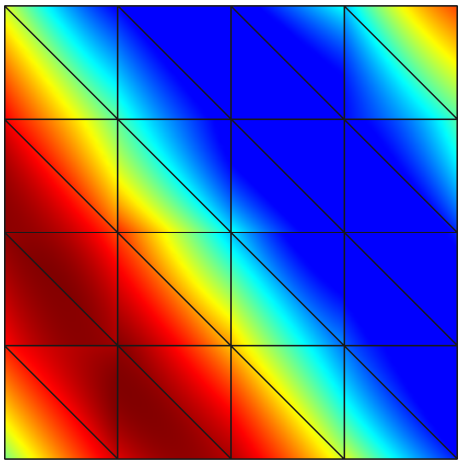
$$\begin{aligned} \rho^{\text{MS}} &= a_\rho + b_\rho \sin(c_\rho x + d_\rho y) \\ u^{\text{MS}} &= a_u + b_u \cos(c_u x + d_u y) \\ &\vdots \\ p^{\text{MS}} &= a_p + b_p \sin(c_p x + d_p y) \end{aligned} \tag{31}$$

We use the primitive variables because this makes the calculation of the source term $\mathbf{s}^{\text{MS}}$ easier. In the case of the Reynolds-averaged Navier-Stokes equations, we also make any additional variables sinusoidally varying, such as the working variable $\tilde{\nu}$ in the SA model.

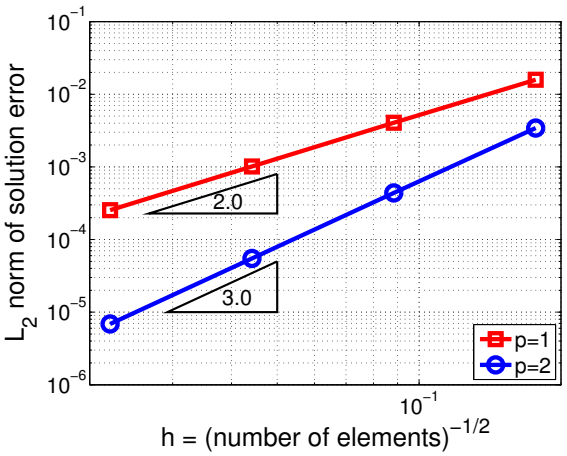
Figure 9 shows the results of manufactured solution tests for three equations: Euler, compressible Navier-Stokes, and Reynolds-averaged Navier-Stokes. The parameters used in defining $\mathbf{u}^{\text{MS}}$ are given in Table 1. In the convergence plots, we see that the continuous L_2 error norms in the discrete solutions decrease at the expected rates of $p + 1$. This demonstrates a successful verification of the discretization via a manufactured solution.

Table 1: Parameters entering (31) used in the manufactured solution test presented in Figure 9. Additional quantities are: gas constant, $R = 1.0$; Prandtl number, $Pr = 0.71$; specific heat ratio, $\gamma = 1.4$. For the viscous cases, a constant viscosity of $\mu = 0.01$ was used. For the RANS case, the wall distance was measured from the bottom boundary of the domain, which was a unit square.

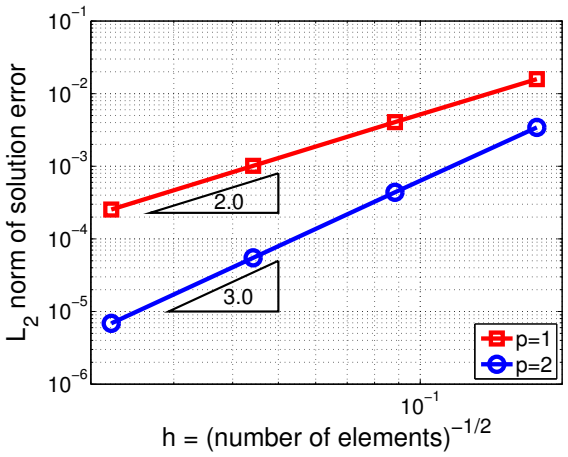
Quantity	a	b	c	d
ρ^{MS}	.9	.04	-2	1
u^{MS}	.1	.02	1	1
v^{MS}	.05	-.1	0.7	1.3
p^{MS}	1	.05	2	-1
$\tilde{\nu}^{\text{MS}}$	.02	.03	-1	2



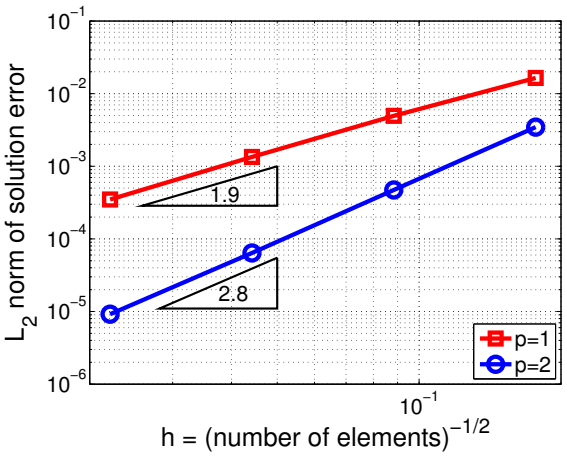
9.1: Manufactured solution, ρ



9.2: Euler discretization test



9.3: Compressible Navier-Stokes test



9.4: Spalart-Allmaras RANS test

Figure 9: Manufactured solution tests for DG discretizations of the Euler, compressible Navier-Stokes, and RANS-SA equations, each at orders $p = 1$ and $p = 2$, on a unit domain $\Omega = [0, 1]^2$. Errors shown are continuous L_2 norms between the discrete solution to (30) and the exact manufactured solution, $\mathbf{u}^{\text{MS}}$. Note that optimal order $p + 1$ convergence rates are (nearly) attained for all runs.

2.3.2 Adjoint Sensitivity Tests

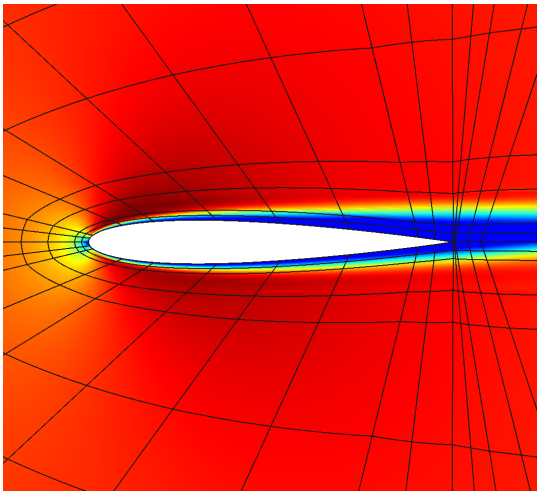
One way to test a discrete adjoint is to compare output sensitivities computed using the adjoint to those computed using finite differences. In this example we show this comparison for a viscous flow over a NACA 0012 airfoil at $M = 0.5$, $Re = 5000$, $\alpha = 2°$. We are interested in the sensitivity of the lift coefficient to angle of attack. Since the lift is defined as the force perpendicular to the free-stream direction, the output depends directly on the input parameters, so that we need to augment (19) with the partial derivative of J with respect to the input angle of attack, α,

$$\frac{dJ}{d\alpha} = \mathbf{\Psi}^T \frac{\partial \mathbf{R}}{\partial \alpha} + \frac{\partial J}{\partial \alpha}. \tag{32}$$

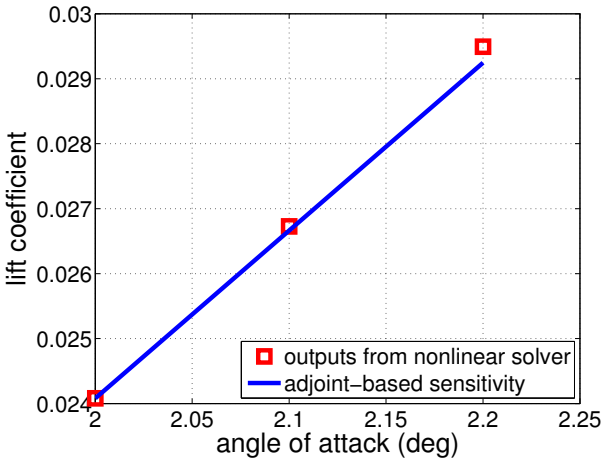
For the present test we compute $\frac{\partial \mathbf{R}}{\partial \alpha}$ using a simple forward difference,

$$\frac{\partial \mathbf{R}}{\partial \alpha} \approx \frac{\mathbf{R}(\mathbf{U}, \alpha + \Delta\alpha) - \mathbf{R}(\mathbf{U}, \alpha)}{\Delta\alpha}, \tag{33}$$

where $\Delta\alpha$ is a small value, e.g. 0.01rad. In Figure 10, we show a comparison of the sensitivity computed using (32), i.e. a local linearized sensitivity, to outputs obtained by actually perturbing the input parameter. The qualitative agreement is excellent, and a quantitative look shows convergence at the expected rate as the angle of attack perturbation goes to zero.



10.1: Mach number contours



10.2: lift coefficient sensitivity

Figure 10: Verification of the discrete adjoint solver using a parameter sensitivity test. The angle of attack is varied in a viscous flow over a NACA 0012 airfoil, and the resulting data points are overlaid on a line through the baseline, $\alpha = 2°$, case, with a slope computed from the discrete lift coefficient adjoint. The agreement is excellent for small α perturbations; deviations at larger α are due to the nonlinear nature of the compressible Navier-Stokes equations.

3 Output Error Estimation

Numerical errors due to insufficient mesh resolution can affect outputs, often in seemingly subtle but significant ways. The latter point is especially true for the convection-dominated flows common to aerodynamics. Our goals are to quantify the effect of these numerical errors and to reduce them through mesh adaptation. In this section, we consider the first of these: estimation of output error.

3.1 Two Discretization Levels

Without access to infinite resolution, estimating the true numerical error in an output is practically out of reach for general nonlinear problems. We thus resign ourselves to estimating the output error between two finite-dimensional spaces: a coarse approximation space ($\boldsymbol{\mathcal{V}}_H$) on which we calculate the state and output, and a fine space ($\boldsymbol{\mathcal{V}}_h$) relative to which we estimate the error. The equations and output representations on these spaces are

$$\text{coarse space:} \quad \rightarrow \underbrace{\mathbf{R}_H(\mathbf{U}_H) = 0}_{N_H \text{ equations}} \rightarrow \underbrace{\mathbf{U}_H}_{\text{state} \in \mathbb{R}^{N_H}} \rightarrow \underbrace{J_H(\mathbf{U}_H)}_{\text{output (scalar)}}$$

$$\text{fine space:} \quad \rightarrow \underbrace{\mathbf{R}_h(\mathbf{U}_h) = 0}_{N_h \text{ equations}} \rightarrow \underbrace{\mathbf{U}_h}_{\text{state} \in \mathbb{R}^{N_h}} \rightarrow \underbrace{J_h(\mathbf{U}_h)}_{\text{output (scalar)}}$$

We would like to measure the output error in the coarse solution relative to the fine space,

$$\text{output error:} \quad \delta J \equiv J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h). \tag{34}$$

The fine space is typically constructed by uniformly refining each element in the coarse space, or by increasing each element's approximation order. Figure 11 illustrates a fine space obtained by uniform refinement. We assume that the fine approximation space contains the coarse approximation space, so that the following lossless state injection, $\mathbf{U}_h^H$, is possible:

$$\mathbf{U}_h^H \equiv \mathbf{I}_h^H \mathbf{U}_H, \tag{35}$$

where $\mathbf{I}_h^H$ is the coarse-to-fine state injection (prolongation) operator.

3.2 The Adjoint-Weighted Residual

On the fine space, the exact solution $\mathbf{U}_h \in \mathbb{R}^{N_h}$ would give us zero fine-space residuals,

$$\mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}. \tag{36}$$

However, the state injected from the coarse space will generally not be a fine space solution and hence will not give us zero fine-space residuals,

$$\mathbf{R}_h(\mathbf{U}_h^H) \neq \mathbf{0}. \tag{37}$$

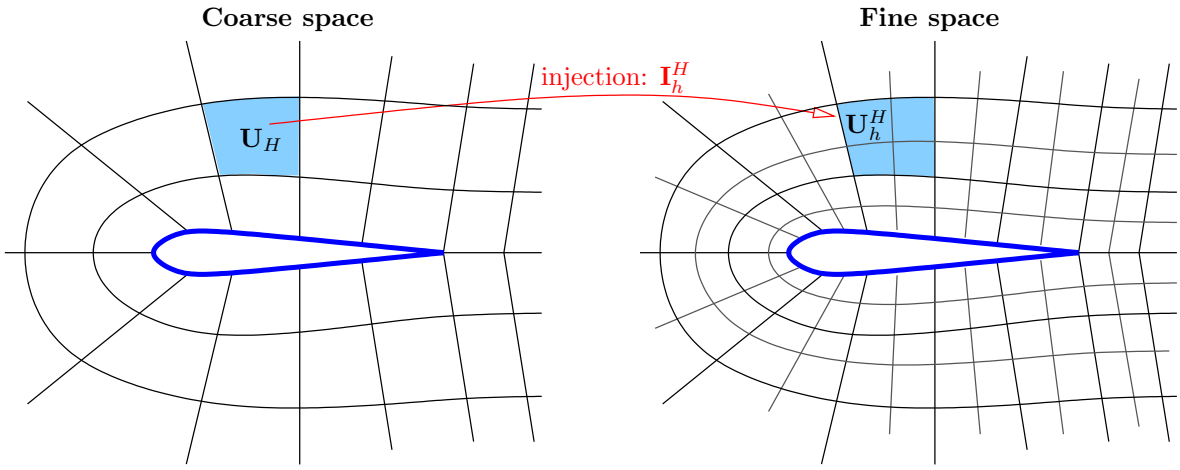


Figure 11: Sample fine approximation space obtained by uniform refinement of each element in the mesh of the coarse approximation space. An alternative fine space is obtained by incrementing the approximation order of each element.

Instead, the injected coarse state solves a *perturbed* fine-space problem,

$$\text{find } \mathbf{U}'_h \text{ such that: } \quad \mathbf{R}_h(\mathbf{U}'_h) \underbrace{-\mathbf{R}_h(\mathbf{U}_h^H)}_{\delta\mathbf{R}_h} = 0 \quad \Rightarrow \quad \text{answer is: } \mathbf{U}'_h = \mathbf{U}_h^H. \tag{38}$$

As this is just the fine-space problem with a residual perturbation, the fine-space adjoint, $\mathbf{\Psi}_h$, tells us to expect an output perturbation given by the inner product between the adjoint and the residual perturbation,

$$\underbrace{J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h)}_{\approx \delta J} = \mathbf{\Psi}_h^T \delta\mathbf{R}_h = -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H). \tag{39}$$

This derivation assumes small perturbations in $\mathbf{U}$ and $\mathbf{R}$ when the output or equations are nonlinear. Calling the left-hand side δJ assumes $J_H(\mathbf{U}_H) = J_h(\mathbf{U}_h^H)$, which is true if the output definition (e.g. geometry) does not change between the coarse and fine spaces.

In summary, we have

$$\boxed{\delta J \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)} \tag{40}$$

Note that this error estimate does not require the fine-space primal solution, $\mathbf{U}_h$.

3.3 Approximations

The error estimate in (40) uses the adjoint on the fine space, $\mathbf{\Psi}_h$. Obtaining $\mathbf{\Psi}_h$ requires solving a large, possibly expensive, linear system. Suppose we have the *coarse*-space adjoint, $\mathbf{\Psi}_H$, which injected into the fine space is $\mathbf{\Psi}_h^H \equiv \mathbf{I}_h^H \mathbf{\Psi}_H$. Define the adjoint perturbation as

$$\delta\mathbf{\Psi}_h \equiv \mathbf{\Psi}_h^H - \mathbf{\Psi}_h.$$

We then re-write (40) as

$$\delta J \approx \underbrace{-\left(\boldsymbol{\Psi}_h^H\right)^T \mathbf{R}_h(\mathbf{U}_h^H)}_{\text{computable correction}} + \underbrace{\left(\delta\boldsymbol{\Psi}_h\right)^T \mathbf{R}_h(\mathbf{U}_h^H)}_{\text{remaining error}}. \tag{41}$$

The computable correction is tempting to use as the sole error estimate. It does provide important information on δJ for many discretizations, including reconstructed finite volume. However, it performs poorly as an adaptive indicator because it does not incorporate any new information from the fine space. Moreover, it is zero for finite element discretizations with Galerkin orthogonality. Therefore we need some estimate of $\boldsymbol{\Psi}_h$. Several methods are used in practice,

1. Reconstruct $\boldsymbol{\Psi}_h$ from $\boldsymbol{\Psi}_H$ using information from neighboring elements.
2. Solve for $\boldsymbol{\Psi}_h$ approximately using a cheap iterative smoother on the fine space.
3. Solve for $\boldsymbol{\Psi}_h$ exactly on the fine space if the $N_h \times N_h$ linear system is tractable.

In this work we employ the third option when possible, as it gives the most accurate error estimates (albeit at the highest cost). When the fine-space solve becomes too expensive, we turn to one of the first two approximations; although we sacrifice some accuracy in the error estimate, the adaptive indicator obtained from the error estimate remains similar to that obtained from solving the fine-space adjoint exactly.

For nonlinear problems the leading term not present in (41) is quadratic in the state and adjoint errors. This "error in the error estimate" can be reduced to third-order in the state and adjoint errors by using [90]

$$\delta J \approx -\left(\boldsymbol{\Psi}_h^H\right)^T \mathbf{R}_h(\mathbf{U}_h^H) + \frac{1}{2}\left(\delta\boldsymbol{\Psi}_h\right)^T \mathbf{R}_h(\mathbf{U}_h^H) + \frac{1}{2}\left(\delta\mathbf{U}_h\right)^T \mathbf{R}_h^{\psi}(\boldsymbol{\Psi}_h^H), \tag{42}$$

where $\mathbf{R}_h^{\psi}(\boldsymbol{\Psi}_h^H)$ is the residual vector of the fine-space adjoint problem, (22), and $\delta\mathbf{U}_h = \mathbf{U}_h^H - \mathbf{U}_h$ is the state perturbation. In practice, both the state and adjoint perturbations could be approximated using one of the three approaches outlined above.

3.4 Error Effectivity

The error estimate calculated above is not a bound. We measure its accuracy by defining an *effectivity*,

$$\eta_H = \frac{J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h)}{J_H(\mathbf{U}_H) - J}, \tag{43}$$

where J is the exact output, i.e. calculated from the exact solution. An effectivity close to 1 is desirable. In practice this value will depend on the choice of fine space (order enrichment versus element subdivision), and on approximations made in the fine-space adjoint error estimation. However, we can make some rough a priori estimates. If the output converges as $J_H(\mathbf{U}_H) - J = CH^k$, uniform element subdivision for the fine space yields an effectivity of $\eta_H = 1 - (1/2)^k$ – this does not approach unity as $H \to 0$. On the other hand, if order enrichment is used for the fine space, then the effectivity converges as $\eta_H = 1 - C_1 H^{\delta k}$, where δk is the increase in convergence rate of the fine-space output relative to the coarse-space output. In this case, the effectivity does approach unity with mesh refinement [35].

3.5 Examples

We now present some examples that test the error estimates described in this section. We use uniform mesh refinement studies to determine the rate, k, at which certain errors, notably the output error, asymptotically converge,

$$\text{error} \propto h^k, \quad \text{valid as } h \to 0, \tag{44}$$

where h is a measure of the size of the mesh elements. Even though we have many elements, not all of the same size, we use a single h value that we can think of as indicating a refinement level. When we do a uniform refinement study, we are not concerned with the particular value of h; we just want to know how "uniform" changes in h for every element translate into changes in the error.

In the following studies we will uniformly subdivide mesh elements, by bisecting each edge [3], and measure the resulting effect on the error. In this case, a reasonable definition of h is $\sqrt{1/N_e}$, where N_e is the number of elements. So a uniform refinement of a two-dimensional mesh would increase N_e by a factor of 4, and decrease h by a factor of 2. Taking the logarithm of (44), we obtain

$$\log(\text{error}) \;=\; C + k \log\left(\sqrt{\frac{1}{N_e}}\right). \tag{45}$$

3.5.1 Drag Error for Euler Flow over a Bump

In this example, we consider inviscid subsonic flow inside a channel that has a smooth Gaussian perturbation on the bottom wall. The compressible Euler equations govern the flow, and the output J is the drag (horizontal) force on the bottom wall. The solution on a fine mesh is shown in Figure 12.1.

Figure 12.2 shows the initial coarse mesh for the uniform refinement study. On this mesh, we perform the following steps:

1. Solve for the flow using $p = 1$. This gives us $\mathbf{U}_H$, which is the state in the coarse approximation space. Also compute $J_H = J_H(\mathbf{U}_H)$.

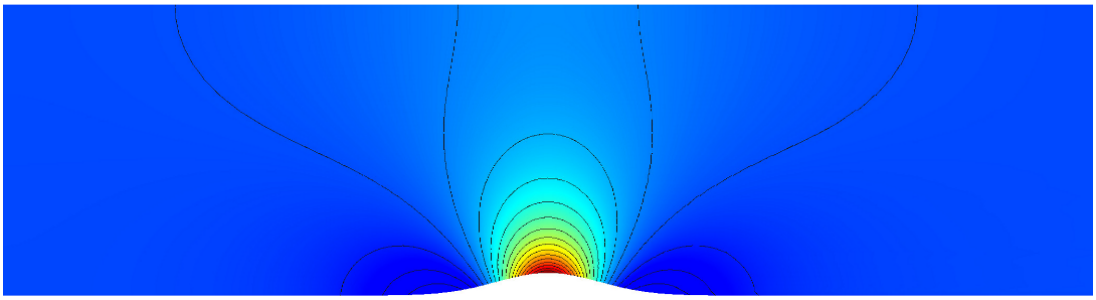
2. Solve the adjoint exactly using $p = 2$. This gives us $\mathbf{\Psi}_h$, which is in the fine approximation space.

3. Compute the error estimate using (40) and find the associated corrected output,

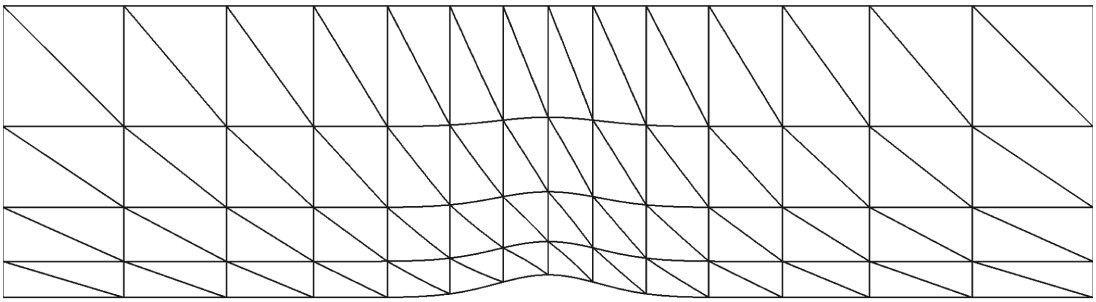
$$\text{corrected output} = J_H - \delta J. \tag{46}$$

We then repeat these steps on three successive refinements of the coarse mesh. We also obtain the "exact" output, J, by solving using $p = 3$ approximation on a mesh that is uniformly refined once more compared to the finest mesh in the study.

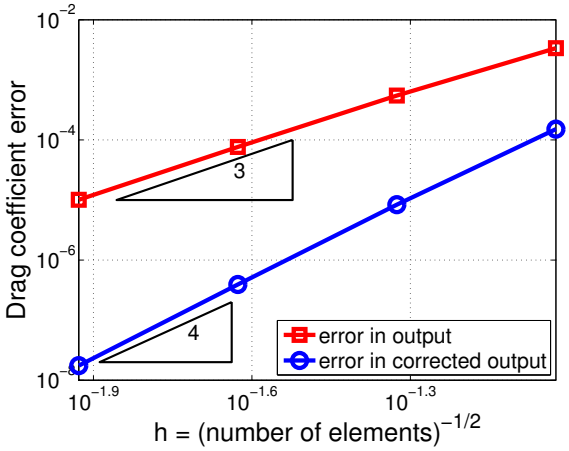
[3] When using curved elements it is important to bisect curved edges as close to along the arc length as possible, and the same applies for element interiors; for example, bisecting a reference element when using a reference-to-global mapping with a lot of nonlinear stretching would not necessarily reduce the size of each element by the same amount.



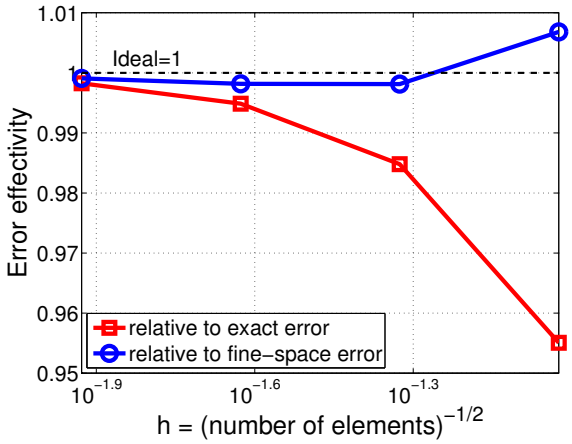
12.1: Mach number contours; range is 0.27 to 0.43



12.2: Coarsest mesh for uniform refinement study: quartic curved triangles



12.3: Convergence of baseline and corrected output



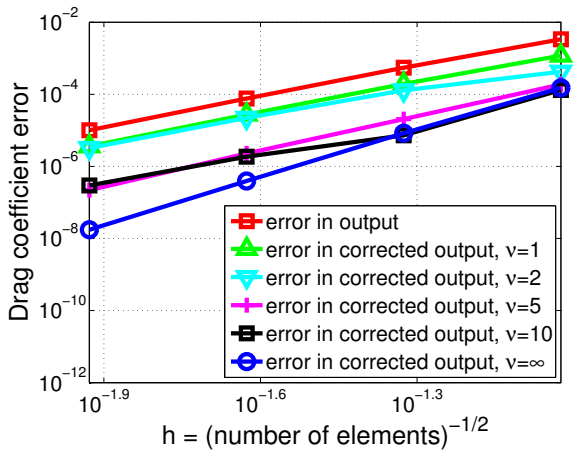
12.4: Convergence of error effectivity

Figure 12: Error estimates and effectivity for inviscid flow in a channel with a Gaussian bump, at $M_\infty = 0.3$, using $p = 1$ approximation. The output of interest is the drag force coefficient on the bottom wall. The adjoint is solved exactly on the fine space.

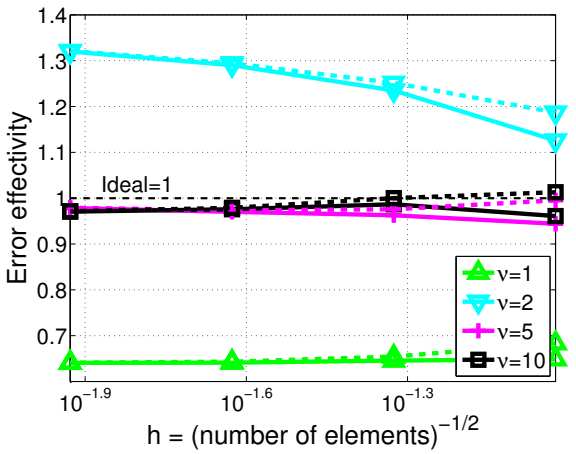
Figure 12.3 shows the convergence of the errors in the coarse-space output, $J_H - J$, and in the corrected output, $J_H - \delta J - J$. Based on the formula in (45), we expect (asymptotically) straight lines on a log-log plot, and this is what we see. The slopes tell us the convergence rates. First, we observe a rate of $k = 3$ for the output error $J_H - J$, and for our $p = 1$ approximation, this is a super-convergent result ... approximation theory would lead us to expect a rate of $p + 1$ but we are actually seeing a rate of $2p + 1$ (we can verify this with data from higher p). Second, we observe a rate of 4, $2p + 2$, for the corrected output. So we see that the output correction buys us an extra order of convergence for the output, which is a reasonable result as we are using order enrichment for the fine space.

Figure 12.4 plots the error effectivity, (43), for the solutions on the 4 mesh refinements. We actually plot two effectivities: one as defined in (43) (this is relative to the exact output J), and a similar one but relative to the fine-space output J_h. The latter choice gives us an idea of the errors we make in estimating $J_H - J_h$, while the former tells us about $J_H - J$. We see that as expected (by design), we estimate the error better relative to the fine space than relative to the true output, but that both effectivities approach 1 as $H \to 0$.

We mentioned that in practice we sometimes solve the fine-space adjoint problem approximately, to avoid the cost of a full solve on the fine space. It turns out that this approximation degrades the accuracy of the error estimates, resulting in worse convergence rates for the corrected output compared to when using an exact fine space adjoint. Figure 13 shows this degradation when using ν iterations of an element-block Jacobi solver to smooth the fine-space adjoint after injection from the coarse space. We see that when



13.1: Convergence of baseline and corrected outputs



13.2: Convergence of error effectivities

Figure 13: Effect of inexact fine-space adjoint solve, using ν element-block smoothing iterations, on the error estimates for inviscid flow in a channel with a Gaussian bump, at $M_\infty = 0.3$, using $p = 1$ approximation. The output of interest is the drag force coefficient on the bottom wall.

ν is small (less work on the fine space), the error estimates are not great, although the corrected output is still better than the original coarse one. As ν increases, the error estimates approach the exact adjoint solve result ($\nu \to \infty$), and the effectivities close in on 1. The "adequate" number of smoothing iterations will depend on the case and on

the smoother, in addition to the desired accuracy levels, so these results should not be interpreted as a recipe for how to choose ν. Instead, one take-away message is that to get the best error estimates possible, we need the best possible fine-space adjoint; however, another one is that even approximate fine-space adjoints can lead to improved outputs. Furthermore, in an adaptive setting, the indicators obtained from these error estimates will turn out to be very good at driving mesh refinement.

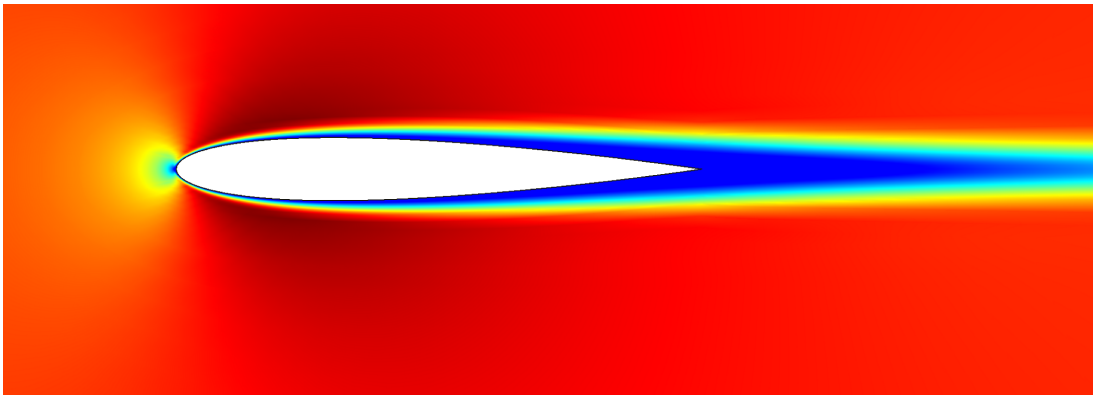
3.5.2 Drag Error for Viscous Flow over a NACA 0012 Airfoil

In this example, we consider viscous subsonic ($M_\infty = 0.5$) flow over a NACA 0012 airfoil (closed-trailing-edge [31]) at zero angle of attack. The compressible Navier-Stokes equations, at $Re = 5000$, $Pr = 0.71$, and constant viscosity, govern the flow, and the output J is the drag (horizontal) force on the airfoil, at which an adiabatic no-slip boundary condition is imposed. The solution on a fine mesh is shown in Figure 14.1.

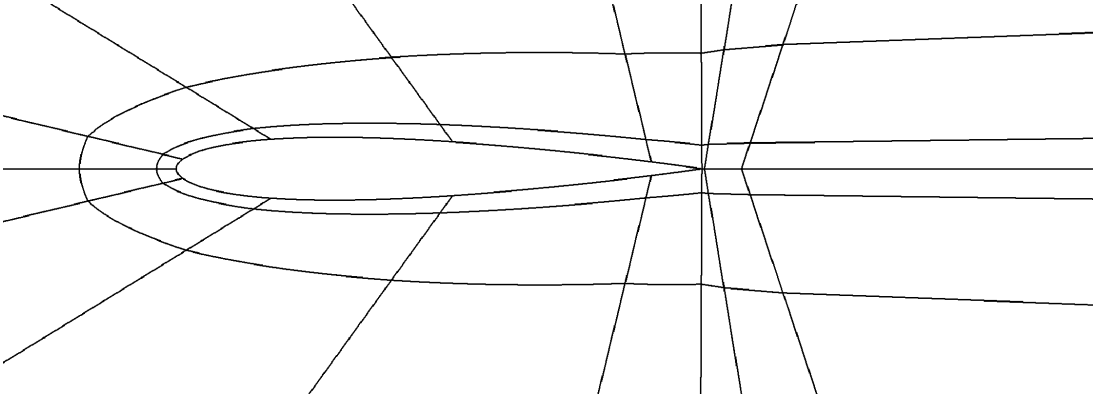
We follow the same steps as in the previous example for measuring the convergence rate of the output and the corrected output (i.e. the error estimate)[4]. Figure 14.3 shows the convergence of the errors in the coarse-space output, $J_H - J$, and in the corrected output, $J_H - \delta J - J$. The lines are not straight initially, indicating that the solution is not yet in the asymptotic regime. However, by the finer meshes we observe relatively straight lines on the log-log plot. We observe a rate of $k \approx 2.7$ for the output error $J_H - J$, and for our $p = 1$ approximation, this is still super-convergent relative to the expected rate of $p + 1$. For the corrected output, we observe a higher rate of 3.8, and so we see that the output correction buys us an extra order of convergence, which is again reasonable as we are using order enrichment for the fine space.

The effectivity story in Figure 12.4 is similar to that of the previous example. The error estimate does a better job at predicting the error between the coarse space and the fine space than at predicting the error relative to the exact output, but both of the effectivities converge to 1 as $H \to 0$.

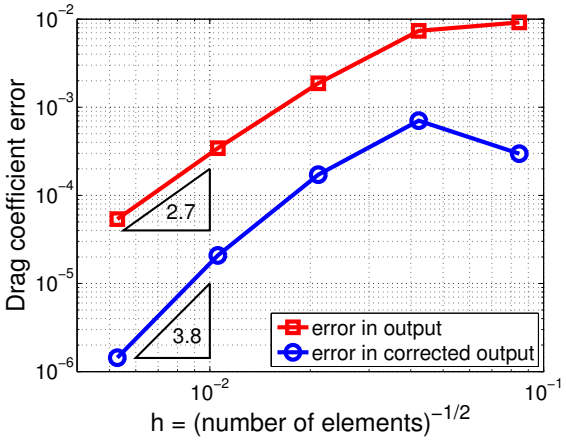
[4]We also follow the remedy for alleviating the effects of p-dependence of the BR2 residual on the error estimates, by evaluating the fine-space residual with order p integration rules and stabilization approximation. [113]



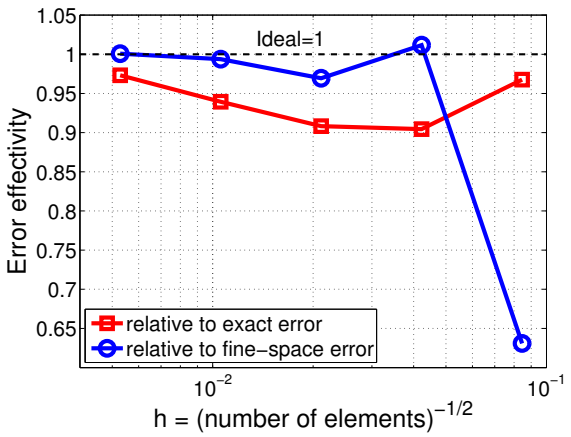
14.1: Mach number contours; range is 0.0 to 0.6



14.2: Coarsest mesh for uniform refinement study: quartic curved quadrilaterals



14.3: Convergence of outputs



14.4: Convergence of error effectivity

Figure 14: Error estimates and effectivity for viscous $Re = 5000$, $M_\infty = 0.5$ flow over a NACA 0012 airfoil at $\alpha = 0$, using $p = 1$ approximation. The output of interest is the drag force coefficient on the airfoil, and the adjoint is solved exactly on the fine space.

4 Mesh Adaptation

The output error estimate derived in the previous section provides error bars on quantities of interest from numerical simulations. However, the benefit of these estimates extends beyond the error bars. Because the output error estimate takes the form of a weighted residual, and because local mesh refinement decreases residuals, the error estimate provides a means of targeting for refinement areas of the computational domain that give rise to the output error. This is the key idea of output-based mesh adaptation.

4.1 Error Localization

The adjoint-weighted residual error estimate in (40) can be localized to the elements by keeping track of the contributions from each fine-space element, indexed by e below,

$$\begin{aligned} J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) &= -\sum_e \mathbf{\Psi}_{he}^T \mathbf{R}_{he}(\mathbf{U}_h^H) \\ \Rightarrow \quad \epsilon_e &\equiv \left| \mathbf{\Psi}_{he}^T \mathbf{R}_{he}(\mathbf{U}_h^H) \right|, \end{aligned}$$

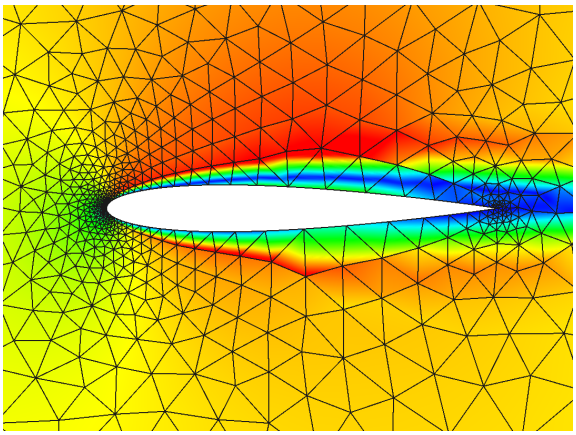
where the subscript e indicates restriction to element e, and the adaptive indicator ϵ_e is obtained by taking the absolute value of the elemental contributions. When order enrichment is used for the fine space in error estimation, this ϵ_e is the adaptive indicator for each element in the current mesh and can be used directly to drive a mesh adaptation strategy. When element refinement is used for the fine space, then the indicators for the fine sub-elements of a coarse element need to be summed to obtain the coarse-element adaptive indicator. The steps involved in obtaining the adaptive indicator, ϵ_e are summarized graphically in Figure 15.

4.2 Adaptation Mechanics

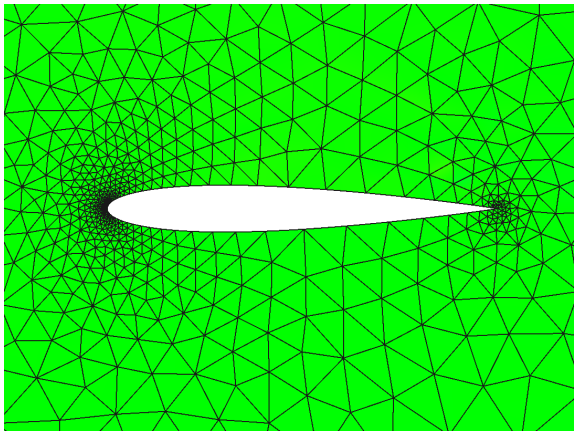
Numerous strategies exist for translating the error indicator into a modified computational mesh. In CFD for aerospace engineering, the most popular adaptation strategy is h-adaptation, in which only the triangulation forming the mesh is modified. This modification usually consists of targeted refinement and coarsening, although pure node repositioning, sometimes called r-refinement, has also been investigated [19, 72]. For high-order methods, additional strategies include p-adaptation, in which the approximation order is changed on a fixed triangulation [104, 68], and hp-adaptation in which both the order and the triangulation are varied [48, 14, 89, 2, 59, 26, 56, 74, 58, 62]. For CFD applications, in which solutions often possess localized, singular features, h-adaptation is key to an efficient adaptation strategy. With the growing availability of high-order methods, however, hp-adaptation can now be used to further improve efficiency.

4.2.1 Local Refinement

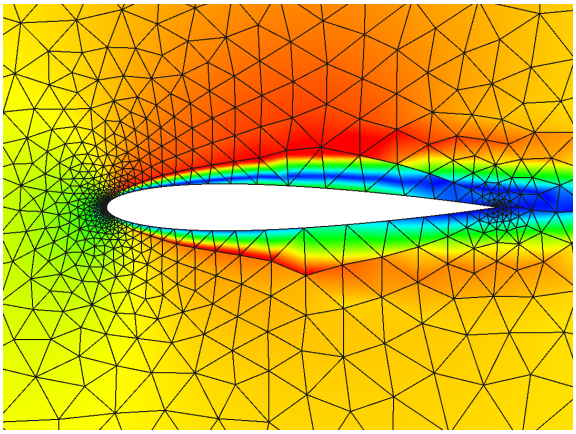
Many approaches to adapting a mesh rely upon the application of local operators through which the mesh is modified incrementally. A simple example of a local operator is element sub-division in a setting that supports non-conforming, or hanging, nodes [13, 63, 90, 26, 102]. For triangular and tetrahedral meshes, local mesh modification operators consist of



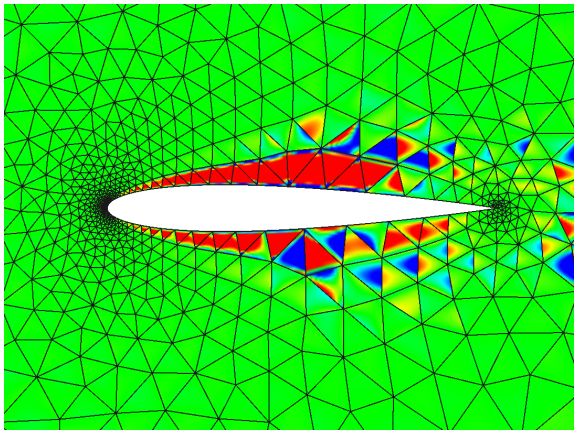
15.1: $p = 1$ state, $\mathbf{U}_H$



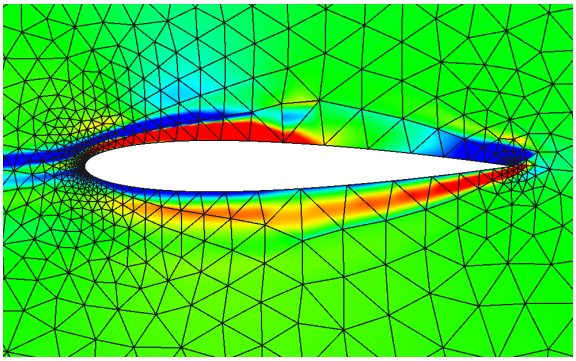
15.2: $p = 1$ residual, $\mathbf{R}_H(\mathbf{U}_H)$ (zero as expected)



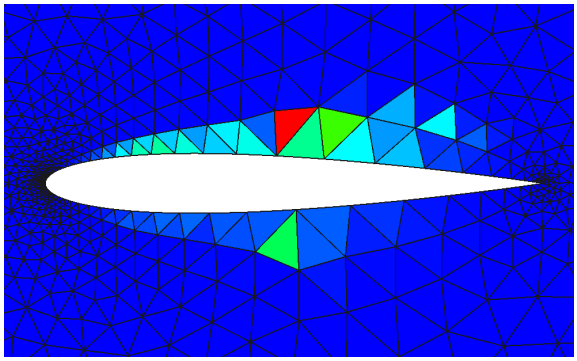
15.3: $p = 1$ state injected to $p = 2$, $\mathbf{U}_h^H$



15.4: $p = 2$ residual, $\mathbf{R}_h(\mathbf{U}_h^H)$



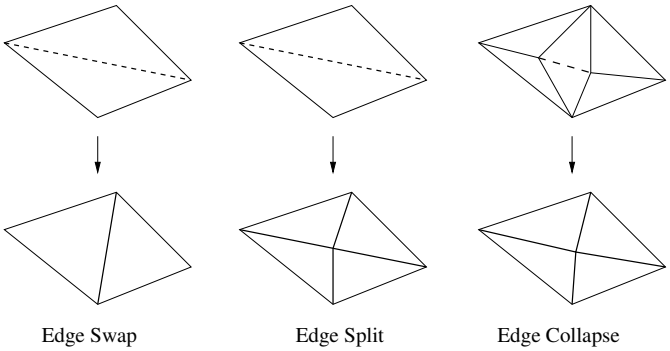
15.5: $p = 2$ lift adjoint, $\mathbf{\Psi}_h$



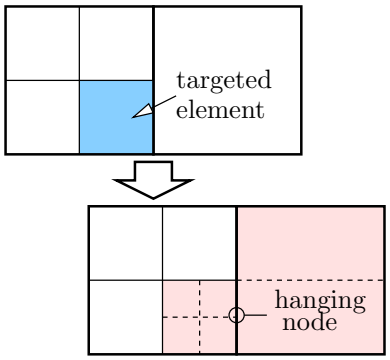
15.6: Error indicator, $\epsilon_e = |\mathbf{\Psi}_{he}^T \mathbf{R}_{he}(\mathbf{U}_h^H)|$

Figure 15: Quantities involved in the calculation of the error estimate and adaptive indicator for $Re = 5000$, $\alpha = 2°$, $M = 0.1$ flow over a NACA 0012 airfoil.

node insertion, face/edge swapping, edge collapsing, and node movement, as illustrated in Figure 16.1. These operators have been studied extensively by various authors [41, 18, 20,



16.1: Unstructured local mesh operators



16.2: Hanging-node refinement

Figure 16: Local mesh modification adaptation operators in two dimensions.

109, 49, 110, 4, 82, 83] in different contexts. The primary advantage of local operators is their robustness: the entire mesh is not regenerated all at once, but rather each operator affects only a prescribed number of nodes, edges, or elements.

Another local operation, especially relevant for discontinuous Galerkin discretizations, is hanging-node mesh refinement, illustrated for quadrilateral elements in Figure 16.2. The non-conforming nature of a hanging node mesh does not significantly affect the DG discretization, which does not enforce solution continuity between elements. Hanging node refinement could be isotropic or directional [21]. It is particularly useful for initially-structured, stretched meshes for viscous flows, in which only a few refinements of key regions may dramatically improve the accuracy of an output.

4.2.2 Global Re-Meshing

Another approach to adapting a mesh is global re-meshing, in which a new mesh is generated for the entire computational domain. The original, or background, mesh is used to store desired mesh characteristics during regeneration. For applications to adaptation, the desired mesh characteristics are often described using a Riemannian metric, the idea being that in an optimal mesh, all edge lengths will have unit measure under the metric [20, 49]. In a Cartesian coordinate system of dimension d, an infinitesimal segment $\delta\mathbf{x}$ has length $\delta\Gamma$ under a Riemannian metric $\mathbf{M}$,

$$\delta\Gamma^2 = \delta\mathbf{x}^T \ \mathbf{M} \ \delta\mathbf{x} = \delta x_i \ M_{ij} \ \delta x_j, \tag{47}$$

where δx_i are the components of $\delta\mathbf{x} \in \mathbb{R}^d$, M_{ij} are the components of the symmetric, positive definite metric, $\mathbf{M} \in \mathbb{R}^{d\times d}$, and summation is implied on the repeated indices $i, j \in [1, \ldots, d]$.

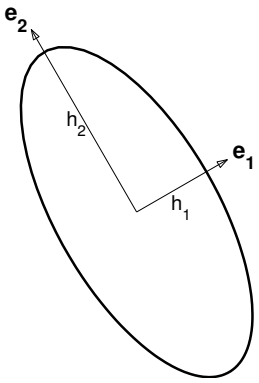
The metric $\mathbf{M}$ contains information on the desired mesh edge lengths in physical space. As $\mathbf{M}$ is symmetric and positive definite, the unit measure requirement,

$$\mathbf{x}^T \ \mathbf{M} \ \mathbf{x} = 1,$$

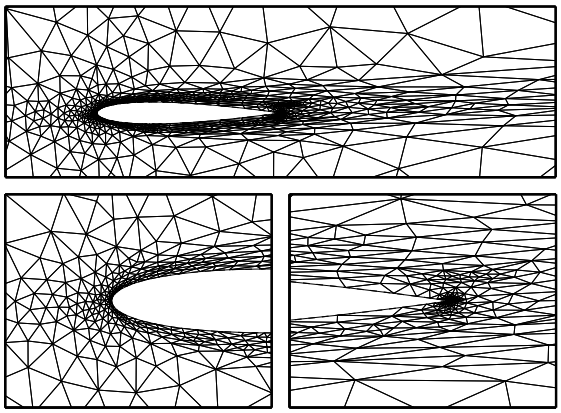
describes an ellipsoid in physical space that maps to a sphere under the action of the metric. The eigenvectors of $\mathbf{M}$ form the orthogonal axes of the ellipsoid – i.e. the principal directions. The corresponding eigenvalues, λ_i, are related to the lengths of the axes, h_i, via

$$\lambda_i = \frac{1}{h_i^2} \quad \Rightarrow \quad \frac{h_i}{h_j} = \left(\frac{\lambda_j}{\lambda_i}\right)^{1/2}$$

Physically, the h_i are the principal stretching magnitudes. A diagram of a possible ellipse resulting from the unit-measure requirement in two dimensions is given in Figure 17.1. Thus, the ratio of eigenvalues of $\mathbf{M}$ can be used to define a desired level of anisotropy.



17.1: Riemannian metric ellipse



17.2: Sample mesh obtained by global re-meshing

Figure 17: Depiction of adaptation using global metric-based re-meshing. On the left, an ellipse representing requested mesh sizes implied by equal measure under a Riemannian metric $\mathbf{M}$, together with the principal directions, $\mathbf{e}_i$, and the associated principal stretching magnitudes, h_i. On the right, sample mesh for viscous flow generated using global re-meshing.

A successful approach for generating simplex meshes based on a Riemannian metric is mapped Delaunay triangulation, in which a Delaunay mesh generation algorithm [99] is applied in the mapped space, allowing for the creation of stretched and variable-size triangles or tetrahedra [71]. This method is implemented in the Bi-dimensional Anisotropic Mesh Generator (BAMG) [15, 55], which has been used in various finite volume [96, 107] and discontinuous Galerkin [34, 6, 79] applications requiring anisotropic meshes. An example of an output-adapted mesh obtained using BAMG is shown in Figure 17.2.

4.2.3 Targeting Strategies

In global re-meshing, all elements can be refined or coarsened based on their error indicators. However, when using local operators, such as hanging-node or order refinement, one must decide which elements to target. The determination of which elements to refine or coarsen has important implications, as too little refinement at each adaptive iteration may result in an unnecessary number of iterations, while too much refinement may result in an expensive solve on an overly-refined mesh. A useful tool for analyzing adaptation targeting strategies is an error distribution histogram [1], in which elements are binned

according to the error indicator. A typical assumption is that in an ideal mesh, the error is equidistributed among the elements [4], and this situation yields a "delta" histogram, in which all elements lie in the same bin. In contrast, the initial coarse mesh will generally have some distribution of error indicators, and the goal of an adaptation targeting strategy is to drive the histogram towards the ideal delta distribution. This characterization of adaptation targeting strategies also holds for runs in which a maximum element count is specified instead of an error tolerance. The ideal mesh in this case is one for which the error is equidistributed among a number of elements within the element budget [112, 113].

Most adaptation targeting strategies are based on a decreasing refinement threshold [76], in which elements with the highest error are targeted for refinement first so that the mesh size grows gradually. For example, a fixed-fraction approach prescribes a fraction of elements with the highest error indicator to be refined at each adaptation iteration, such that the decreasing threshold is a function of the shape of the error histogram. Then, the elements targeted for adaptation are typically refined in a locally uniform manner, e.g. by splitting all edges in half. This simple approach has been applied to output-based adaptation in several studies[11, 54, 7, 101, 62, 75, 23]. The fixed-fraction parameter is often chosen heuristically in a trade-off between an excessive number of iterations and a risk of over-refinement. Nevertheless, the method works quite well for practical problems.

4.2.4 Incorporating Anisotropy

An important ingredient in h-adaptation for aerodynamics is the ability to generate stretched elements in areas such as boundary layers, wakes, and shocks, where the solution exhibits anisotropy, which refers to variations of disparate magnitudes in different directions. Anisotropy can be created to a limited, albeit often sufficient, extent with hanging-node refinement [100, 103]; yet global re-meshing with unstructured triangular and tetrahedral grids offers the most flexibility in tailoring the required stretching.

For spatially second-order methods, the dominant method for detecting anisotropy involves estimating the Hessian matrix of second derivatives of a scalar (e.g. Mach number) computed from the solution [**?** 71, 20, 49]. A mesh metric is obtained from the Hessian by requiring that the approximation error estimate of the scalar quantity u be the same in any chosen spatial direction. The Hessian matrix stores precisely this information, so that this requirement leads to a metric that is directly proportional to the Hessian. The proportionality constant can be tied to the error indicator to incorporate the output error estimate [107].

The definition of a metric tensor becomes difficult for high-order methods because the standard Hessian matrix approach assumes linear approximation of the scalar quantity. One possible extension is based on constructing a metric around the direction of maximum $p+1$st derivative [31, 30, 81]. Another extension involves the calculation of the order $p+1$ derivative tensor that is analogous to the Hessian for $p=1$, or the use of surrogate heuristics based on inter-element jumps for quadrilateral or hexahedral meshes [66].

The metric tensor may also be used to guide an adaptation procedure based on local operators, for example in an effort to make all edges approximately the same length when measured using the metric tensor [20, 28, 110, 83].

The methods mentioned so far rely on a priori analysis and heuristics to predict the desired mesh anisotropy. For example, the approximation error assumptions are made

without regard to the output of interest by using a single scalar, such as the Mach number, to control the anisotropy for a system of equations. However, the assumption that the directional approximation error must be equidistributed for one or more scalar variables at each point in the domain may not be valid, especially when only a scalar output is desired. This observation has motivated research into adaptation algorithms that more directly target the error indicator.

Hessian-based approximation error estimates have been combined with output-based a posteriori error analysis to arrive at an output-based error indicator that explicitly includes the anisotropy of each element [40, 39**?**]. Other works have considered directional output error estimates for quadrilateral and hexahedral meshes [92], and direct node position optimization using an auxiliary adjoint problem [98].

For general unstructured meshes, output error estimation has been incorporated into local mesh operators of element swapping, node movement, element collapse, and element splitting [84]. A direct optimization approach for output error reduction has also been applied to quadrilateral and hexahedral elements, using anisotropic discrete refinement options [60, 52, 21]. Finally, a recent work extends the ideas of direct mesh optimization to general unstructured meshes with global re-meshing, through optimization of a parametrized metric field via local refinement sampling [113].

4.2.5 Adapting in Order

In order/p-adaptation [104], the approximation space is refined or coarsened by changing the order of approximation. With the discontinuous Galerkin method, changing the order is simple and can be done locally on each element [68, 33]. Advantages of p-adaptation are that the computational mesh remains fixed and that an exponential error convergence with respect to degrees of freedom (DOF) is possible for sufficiently smooth solutions. Disadvantages include difficulty in handling singularities and areas of anisotropy and the need for a reasonable starting mesh.

hp-adaptation strives to combine the best of both strategies, employing p-refinement in areas where the solution is smooth and h-refinement near singularities or areas of anisotropy. The motivation for this strategy is that, in smooth regions, p-refinement is more effective at reducing the error per unit cost, compared to h-refinement [**?** 62]. Implemented properly, hp-adaptation can isolate singularities and yield exponential error convergence with respect to DOF. In practice, however, the difficulty of hp-adaptation lies in making the decision between h- and p-refinement, which typically requires either a solution regularity estimate or a heuristic algorithm [62, 17]. An approach that employs output error information to make the hp decision and to choose the appropriate element anisotropy for quadrilateral and hexahedral elements has also been presented [23].

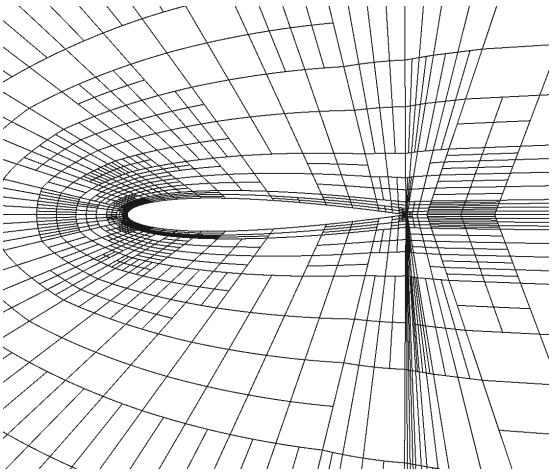
4.3 Examples

The following sub-sections demonstrate applications of output-based refinement to various aerodynamic flows. We consider inviscid, viscous, and turbulent flows in two and three dimensions. Our figure of merit will generally be degrees of freedom, although we also present some results for computational time.

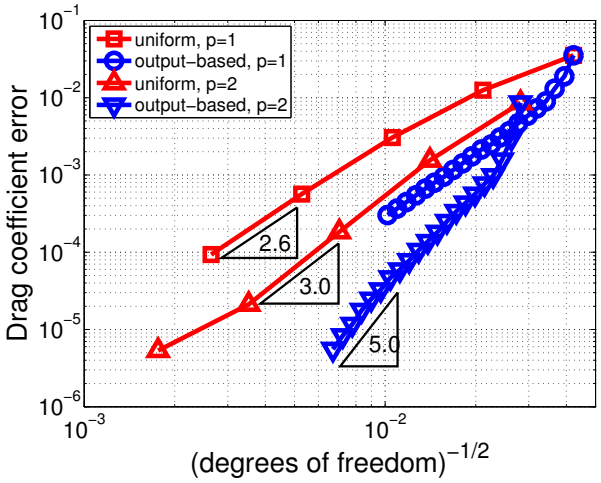
4.3.1 Inviscid Flow over an Airfoil

In this example we adapt a mesh to predict drag in inviscid flow over a NACA 0012 airfoil at $\alpha = 2°$. The initial mesh is the same as that shown in Figure 14.2. We compare uniform refinement to output-driven, isotropic, hanging-node, fixed-fraction refinement with $f^{\text{adapt}} = 0.05$.

Figure 18 shows the convergence of the drag coefficient error with degrees of freedom for $p = 1$ and $p = 2$ approximation. We see that uniform refinement converges at a rate of 2.6 for $p = 1$ and 3.0 for $p = 2$. This can be deemed super-convergent for $p = 1$ but sub-optimal for $p = 2$. The reason for this sub-optimal high-order convergence is due to a singularity of the flow at the trailing edge, which then dominates the error convergence. On the other hand, we see that adaptive refinement produces meshes that make better



18.1: Final drag-adapted mesh for $p = 2$



18.2: Output convergence

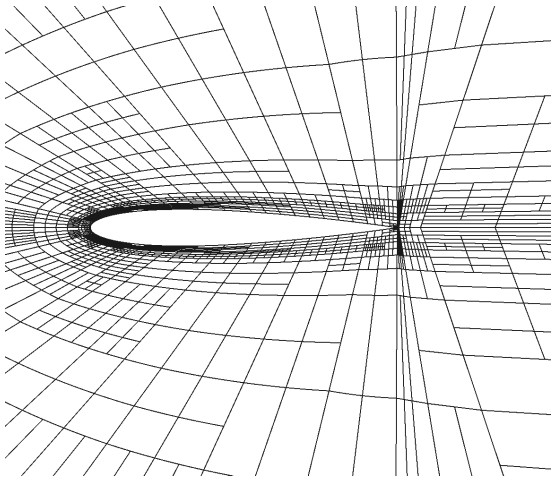
Figure 18: Adaptation results for an inviscid $M_\infty = 0.5$ flow over a NACA 0012 airfoil at $\alpha = 2°$. The output of interest is the drag force coefficient on the airfoil, and, at each adaptive iteration, the adjoint is solved approximately with $\nu = 10$ element-block Jacobi smoothing iterations on the fine space. Using an exact adjoint solve does not perceptibly change the adaptive result.

use of degrees of freedom: the equivalent "convergence rate" for $p = 2$ adaptation is approximately 5, i.e. the expected super-convergent rate of $2p+1$ for an inviscid simulation. We note that although we usually speak of convergence rates only for uniform refinement studies, this notion generalizes to adaptive mesh refinement [112]. For example, for a case in which the entire domain is important (e.g. a very smooth solution), adaptive refinement would (eventually, in turn) target the entire domain too so that its error versus degrees of freedom would converge at the same rate as for uniform refinement. More interestingly, when there are (isolated) singularities present, a situation in which uniform refinement would see its rate limited, adaptive refinement could "quarantine" these singularities with nominal resources and allocate the rest to resolving the remaining smooth regions at the optimal rate. This is indeed what we observe in Figure 18, where, while uniform refinement sees its rate limited by non-smooth solution features for high order, adaptive refinement attains a higher, super-convergent rate.

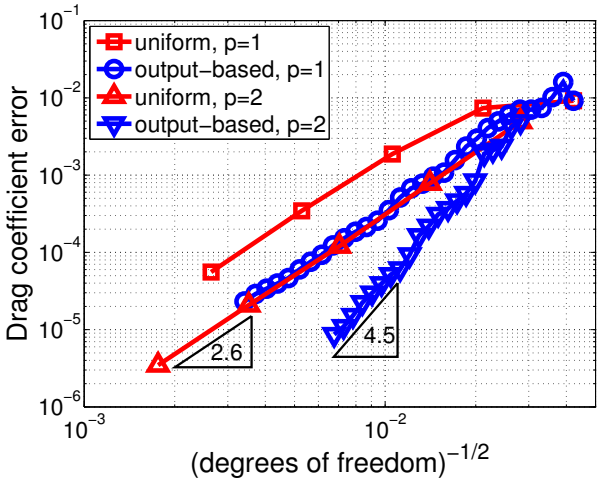
4.3.2 Viscous Flow over an Airfoil

We consider again the case of a NACA 0012 airfoil in viscous flow, introduced in Section 3.5.2. We use $p = 2$ solution approximation and fixed-fraction, $f^{\text{adapt}} = 0.05$, and isotropic hanging-node adaptation.

Figure 19 shows the adaptive results when using a drag-adjoint weighted residual indicator, compared to uniform refinement. As in the previous example, we see that



19.1: Final drag-adapted mesh for $p = 2$



19.2: Output convergence

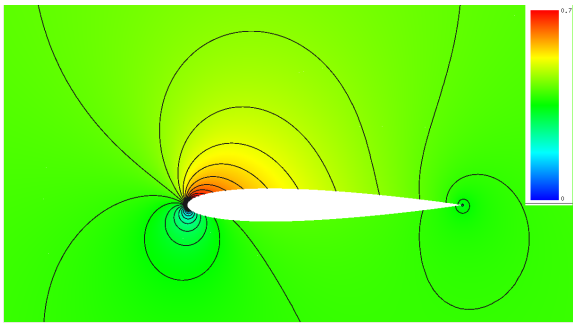
Figure 19: Adaptation results for viscous $Re = 5000$, $M_\infty = 0.5$ flow over a NACA 0012 airfoil at $\alpha = 0$. The output of interest is the drag force coefficient on the airfoil, and, at each adaptive iteration, the adjoint is solved approximately with $\nu = 10$ element-block Jacobi smoothing iterations on the fine space. Using an exact adjoint solve does not perceptibly change the adaptive result.

adaptive refinement produces meshes that make better use of degrees of freedom: the equivalent convergence rate for $p = 2$ is approximately 4.5, slightly above the expected rate of $2p$ for a viscous simulation. Uniform mesh refinement at $p = 2$ only attains a rate of 2.6. Thus, as in the previous example, adaptive refinement can "uncover" the optimal convergence rates for high-order discretizations, resulting in a more efficient use of degrees of freedom.

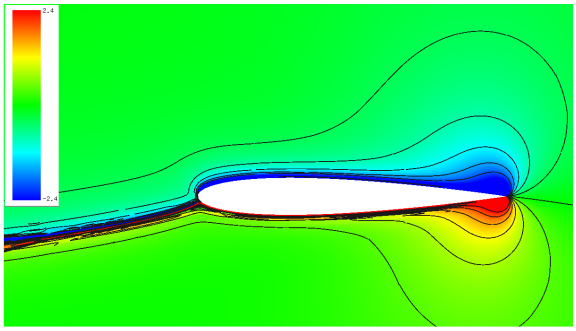
4.3.3 Which Output?

In this example, we consider a NACA 0012 airfoil with a closed trailing edge and a far field approximately 40 chord-lengths away. The initial mesh of cubic quadrilateral elements is illustrated in Figure 22. While the initial mesh appears structured, this structure disappears with the first adaptation iteration and the mesh storage is always fully unstructured. In the following results, quadratic solution approximation, $p = 2$, was used in the discretization, and isotropic h-adaptation was driven by a fixed-fraction strategy with $f^{\text{adapt}} = 0.1$, meaning that at each step of the adaptation, those elements lying in the top 10% of the error criterion were chosen for refinement.

Mach number contours for the airfoil in inviscid flow at $M_\infty = 0.4$, $\alpha = 5°$ are shown in Figure 20. Three different engineering outputs are considered: drag coefficient, lift co-



20.1: Mach number contours



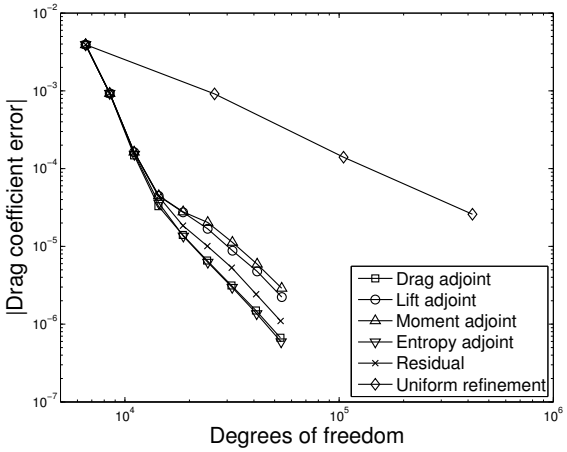
20.2: x-momentum moment adjoint

Figure 20: NACA 0012, $M_\infty = 0.4$, $\alpha = 5°$: Contours of the Mach number and the x-momentum component of the moment adjoint.

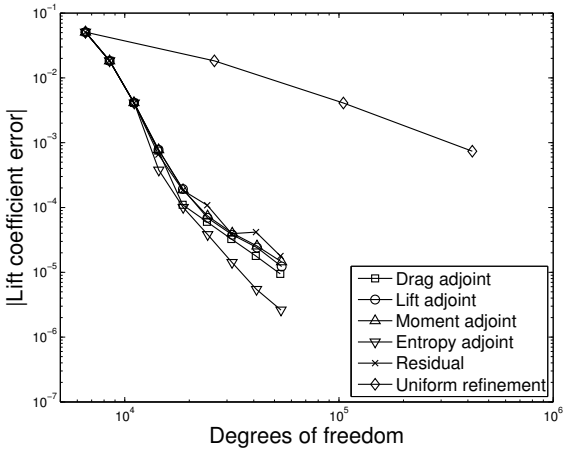
efficient, and leading-edge moment coefficient. All of these outputs were computed using integrals of the inviscid momentum flux, that is, the pressure, on the airfoil surface. Adjoint solutions associated with these outputs were used to drive three different adaptation runs. One adaptation run was also performed using an "entropy-adjoint" indicator [38], in which the entropy variables are interpreted as a "free adjoint" for an output that expresses an entropy balance statement. For comparison, an unweighted residual indicator, equivalent to summing the absolute values of the discrete fine-space residuals, was also tested.

Figure 21 shows the results of adaptation runs driven by the different indicators. Uniform mesh refinement results are given for comparison. The plots show the error in the engineering outputs versus degrees of freedom. Each "truth" output was calculated from a $p = 3$ solution on a mesh obtained by uniformly refining the finest output-adapted mesh. For all three outputs of interest, the adjoint-based adaptive strategies, including the entropy adjoint, perform similarly and are orders of magnitude better than uniform mesh refinement. The unweighted residual indicator performs well for the drag output, and similarly to the output adjoints for the lift and moment outputs. Interestingly, the refinement based on the entropy adjoint actually gives better predictions for lift and moment than the refinements that specifically target those outputs. These results are certainly surprising, but not actually paradoxical, because the procedure does have empirical elements. For example, the lift and moment adaptive indicators target the stagnation streamline in front of the airfoil, perhaps excessively so. As shown in Figure 20 for the moment output, the adjoint varies rapidly across the stagnation streamline. This behavior was suggested in the analysis of Giles and Pierce who found that a square root singularity with respect to distance from the stagnation streamline exists for sources that perturb the stagnation pressure [44]. Intuitively, a force output on the airfoil should respond differently to perturbations that affect the flow over the upper surface of the airfoil versus to those that affect the flow over the lower surface of the airfoil. The singularity is strongest for the lift and moment outputs, and for these cases the performance of the output adjoint adaptation deteriorates the most. The noise created by polynomial approximation of the adjoint on discrete finite elements in this area may be responsible for the excessive refinement.

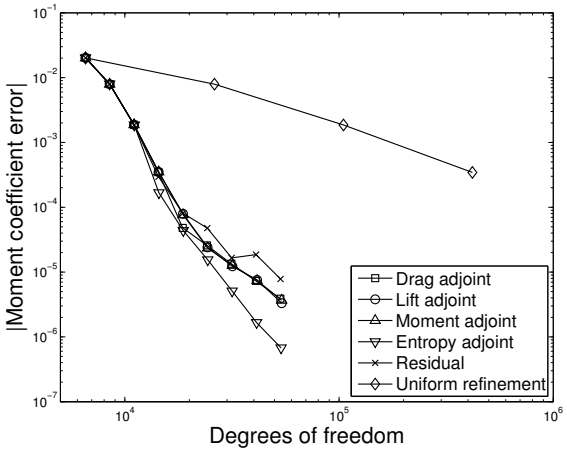
The meshes after eight adaptation iterations of each strategy are shown in Figure 22.



21.1: Drag output



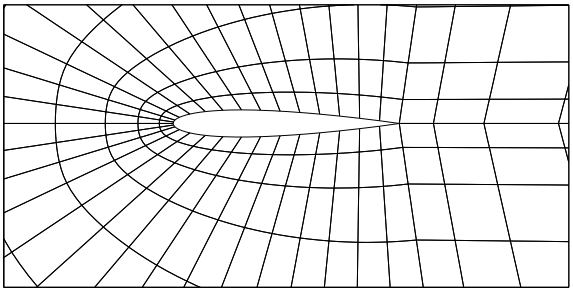
21.2: Lift output



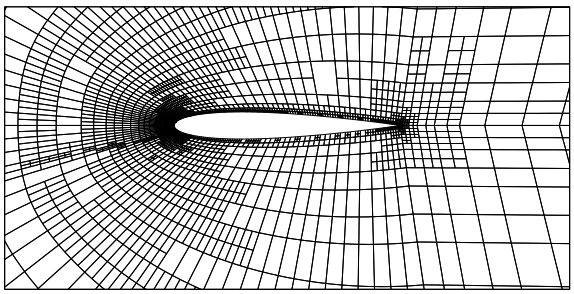
21.3: Moment output

Figure 21: NACA 0012, $M_\infty = 0.4$, $\alpha = 5°$: Comparison of output convergence histories for various adaptation strategies.

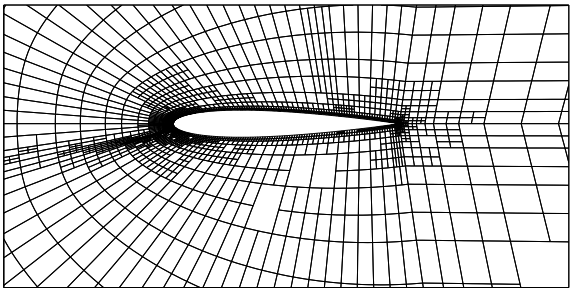
The leading edge, trailing edge, and upper surface of the airfoil are consistently targeted for refinement by the adjoint indicators. The unweighted residual adaptation targets the vicinity of the leading edge and the trailing edge, but not the upper surface of the airfoil, leading to errors in the lift and moment outputs. Refinement of the stagnation streamline is evident in the adjoint-based runs, especially for the lift and moment adaptations. Fi-



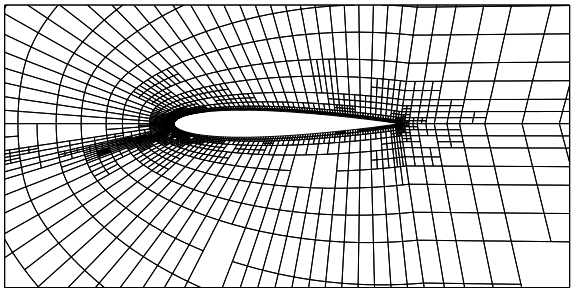
22.1: Initial mesh



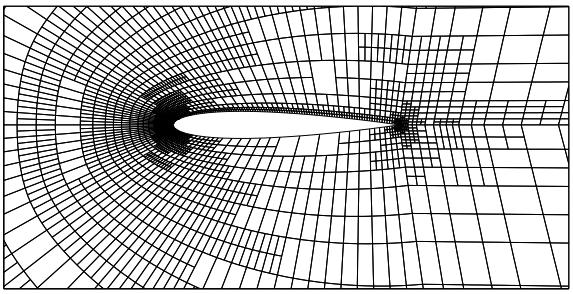
22.2: Drag-adapted



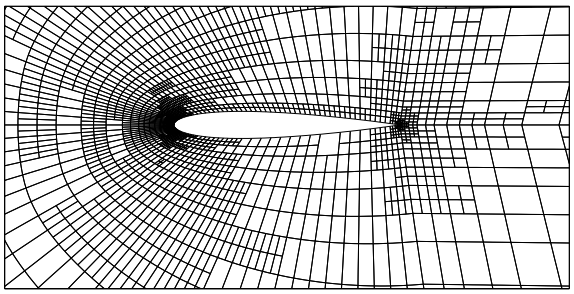
22.3: Lift-adapted



22.4: Moment-adapted



22.5: Entropy-adjoint-adapted



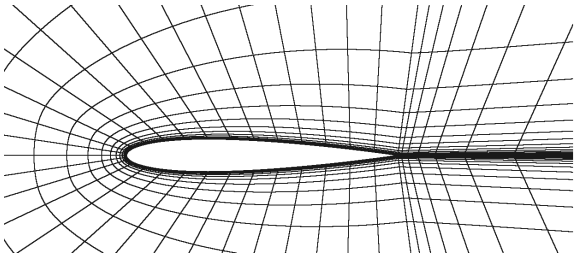
22.6: Residual-adapted

Figure 22: NACA 0012, $M_\infty = 0.4$, $\alpha = 5°$: Meshes after eight adaptation iterations for the tested adaptation strategies.

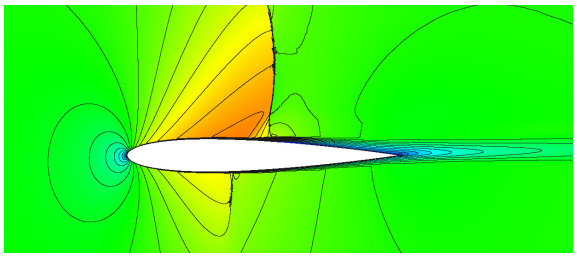
nally, we remark that although residual-based refinement performs well in this example, this is not always the case for more complicated flows (e.g. see Section 5). In addition, residual-based adaptation does not provide an error estimate that could be used as a stopping criterion for the adaptation. The same may be argued for adaptation with the entropy adjoint, as that output is typically not of direct engineering interest. However, it turns out that the entropy adjoint output can be related to drag error [29] and hence could be used as an adaptive stopping criterion when targeting drag.

4.3.4 Transonic Turbulent Flow over an Airfoil

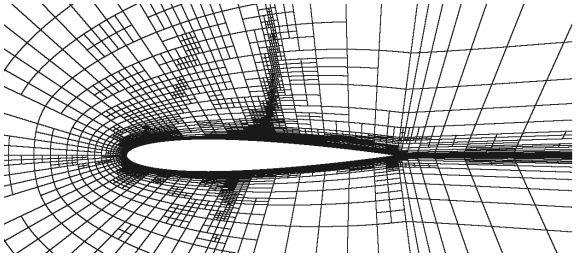
To demonstrate the importance of anisotropy, we present an example of output-based adaptation for turbulent, transonic flow over an NACA 0012 airfoil. The free-stream conditions are $M = 0.8, \alpha = 1.25°, Re = 100,000$. In this case, the flow experiences a relatively strong normal shock on the upper surface and a weak shock on the lower surface. Element-wise constant artificial viscosity is used to capture the shocks [87]. $p = 2$ is used for solution approximation on all elements, and the initial mesh consists of 1740 cubic quadrilateral elements, as shown in Figure 23.



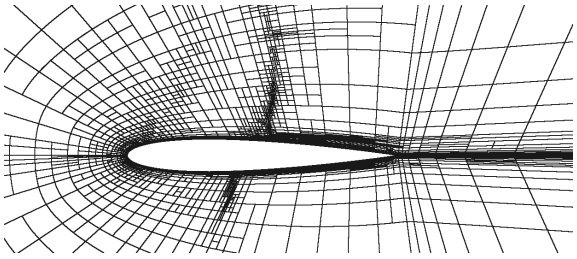
23.1: Initial mesh (1740 elements)



23.2: Mach number contours



23.3: 6^{th} adapted mesh, isotropic (8,736 elements)

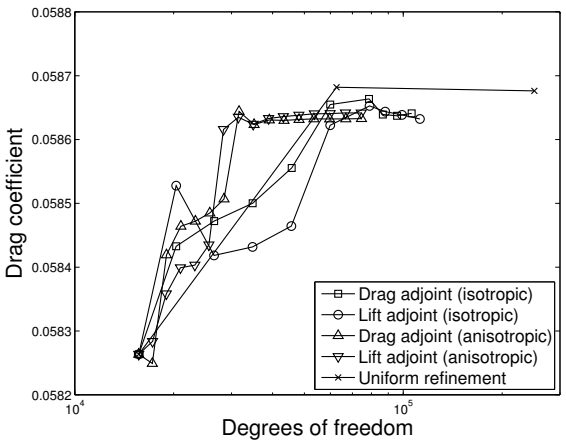


23.4: 10^{th} adapted mesh, anisotropic (4,816 elements)

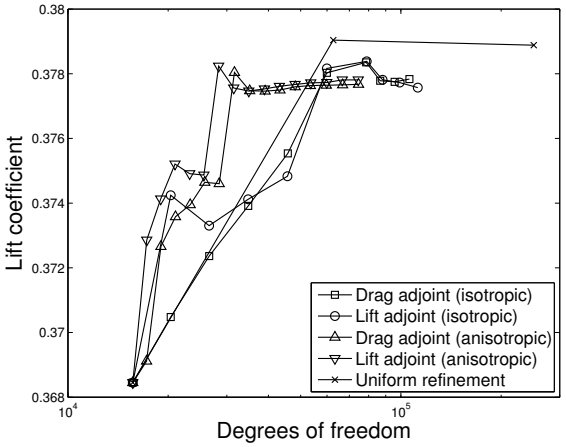
Figure 23: Turbulent NACA 0012, $M = 0.8, \alpha = 1.25°, Re = 100,000$: Initial mesh, solution contours, and adapted meshes. Note, under isotropic adaptation, some elements are refined anisotropically, and these arise from additional refinements required to keep a maximum refinement ratio between adjacent elements below 2. of 2-to-1 ratio between is due to

Convergence of the drag and lift coefficients is shown in Figure 24. The plots are similar in that the lift output converges as rapidly as the drag output for all of the adaptation schemes. Also, the drag adaptation performs well for the lift output and vice versa. The anisotropic adaptations, performed using a discrete-choice output-based approach [21], converge much more rapidly compared to the isotropic adaptations: the outputs do not change much after 40,000 degrees of freedom with the anisotropic adaptation, while changes are still observed after nearly 100,000 degrees of freedom when using isotropic adaptation.

Two of the drag-adapted meshes are shown in Figure 23: one from isotropic adaptation after six iterations and one from anisotropic adaptation after ten iterations. Differences are evident in the boundary layer and wake, where anisotropic adaptation is more efficient. The shock also appears to be more tightly resolved in the anisotropically-adapted mesh.



24.1: Drag output



24.2: Lift output

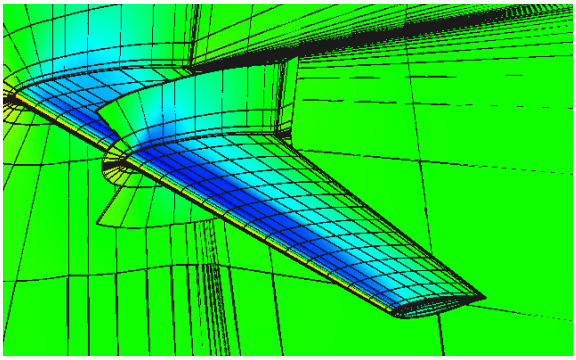
Figure 24: Turbulent NACA 0012, $M = 0.8, \alpha = 1.25°, Re = 100,000$: Drag and lift convergence with degrees of freedom using isotropic and anisotropic refinement. Note, outputs from uniform refinement overshoot the exact values.

4.3.5 Transonic Turbulent Flow over a Wing

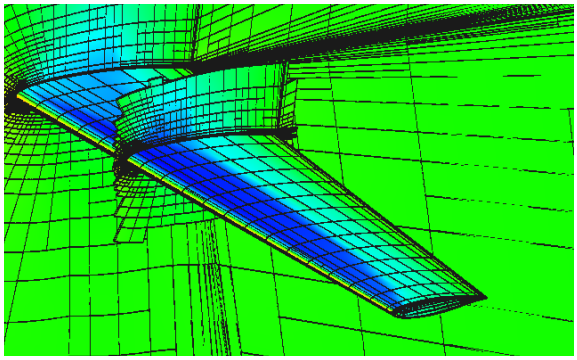
In this example, we demonstrate drag-based adaptation for a steady, three-dimensional, turbulent, transonic flow. We consider the baseline wing geometry (DPW-W1) from the third AIAA Drag Prediction Workshop [43]. The initial curved mesh, shown in Figure 25.1, was obtained through agglomeration of cells from a finer structured linear C-grid generated specifically for this purpose. In the agglomeration, each curved hexahedral element was obtained by merging twenty seven linear elements using a distance-based Lagrange interpolation of the nodal coordinates, resulting in cubic ($q = 3$) geometry interpolation. Also, the spacing of the linear mesh is such that the agglomerated mesh presents $y^+ \approx 1$ for the first element off the wall as recommended in the workshop guidelines [42] and the outer boundary is located at 100 mean-aerodynamic-chord-lengths away from the wing.

We use the Spalart-Allmaras turbulence model without trip terms, and element-wise constant artificial viscosity for shock capturing [87]. The baseline flow solution is obtained with linear ($p = 1$) approximation order, and we study anisotropic, output-based hp-adaptation using discrete choices that include order enrichment [23] – in this strategy, when deciding which discrete refinement option to choose, a figure of merit is used that takes into account the benefit (error addressed) and the cost (the increased Jacobian matrix size). The fine-space adjoint used for error estimation is obtained approximately by using $\nu^{\text{smooth}} = 5$ iterations of an element-block Jacobi smoother. All of the adaptive schemes start from the same initial solution. For the adjoint-based adaptation methods, the CPU time taken for the initial adjoint solve is also included in the initial starting time.

We compare two mesh improvement strategies starting from the initial $p = 1$ solution shown in Figure 25.1. One of the strategies is uniform h-refinement, in which all hexahedra are divided into 8 elements. The other is drag-based hp-adaptation in which $f^{\text{adapt}} = 10\%$



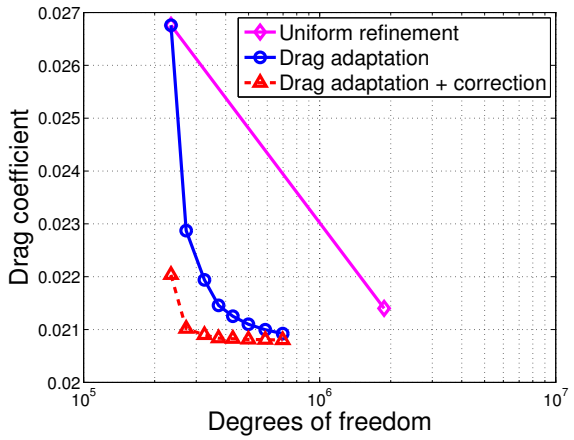
25.1: Initial pressure contours (29310 cubic elements, $p = 1$).



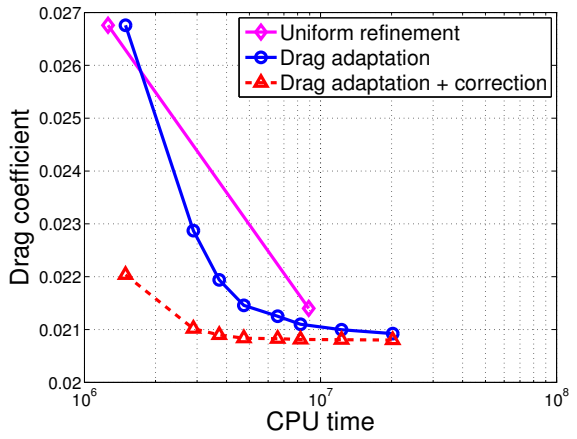
25.2: Pressure contours on the 7^{th} drag-adapted mesh (85377 cubic elements).

Figure 25: DPW Wing 1, $M_\infty = 0.76, \alpha = 0.5°, Re = 5 \times 10^6$: Initial and drag-adapted meshes with pressure contours.

of the elements is selected for refinement at each adaptation step. Additionally, we fix the overall budget of CPU wall-time for each of the three runs and the last converged solutions obtained within that budget are shown in Figure 25.



26.1: Convergence with degrees of freedom



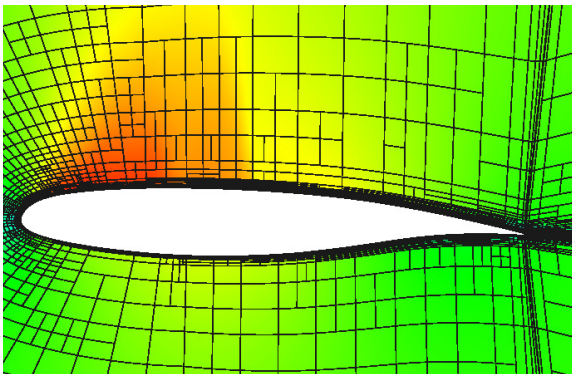
26.2: Convergence with CPU time

Figure 26: DPW Wing 1, $M_\infty = 0.76, \alpha = 0.5°, Re = 5 \times 10^6$: drag coefficient convergence for output-based adaptation compared to uniform refinement, using both degrees of freedom and CPU time.

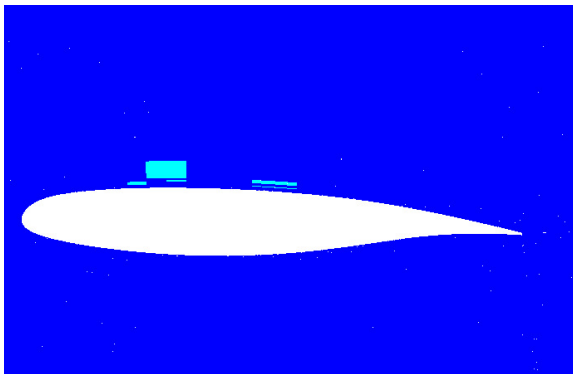
Figure 26 shows the drag coefficient convergence for the mesh refinement strategies. Note that the dashed lines indicate the output corrected with the error estimate. The difference between these corrected values for the last two adaptation steps of the output-based strategy is within 0.15 counts of drag. Note that the performance in terms of degrees of freedom and CPU time of the output-based strategy is better compared to uniform refinement, especially when using the corrected output results.

Figures 27 and 28 show two cuts at representative span-wise positions for the output-based strategy. Note the presence of anisotropic cells along the shock and on the boundary

layer, and the minimal use of p-refinement in this case – this is due in part to the under-resolved nature of the mesh for this flow, and to the relatively large cost ascribed to order enrichment in the definition of the merit function.

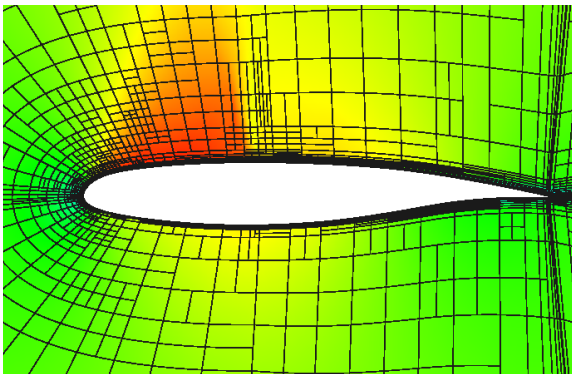


27.1: 7^{th} adapted mesh with Mach contours.

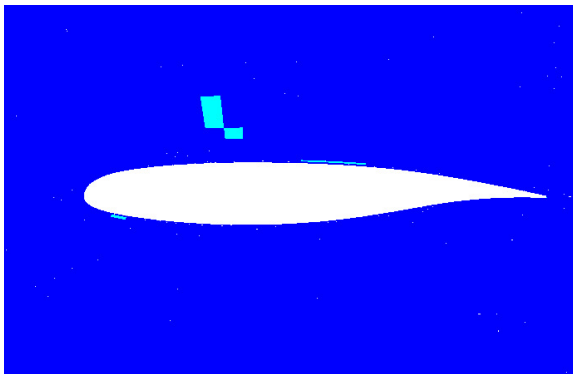


27.2: 7^{th} adapted mesh p-order distribution; the range is $p = 1 \rightarrow 5$.

Figure 27: DPW Wing 1, $M_\infty = 0.76, \alpha = 0.5°, Re = 5 \times 10^6$: cut at $y = 220$mm of the drag-adapted mesh. Note, the reference span is $1524mm$.



28.1: 7^{th} adapted mesh with Mach contours.

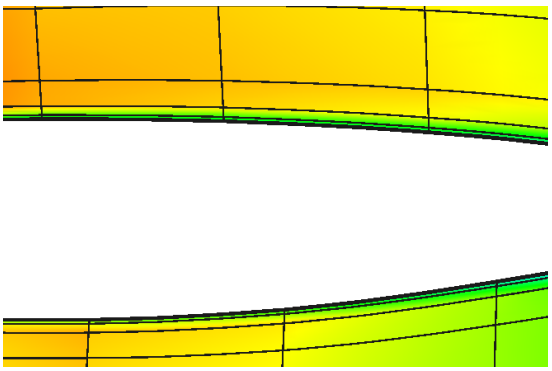


28.2: 7^{th} adapted mesh p-order distribution; the range is $p = 1 \rightarrow 5$.

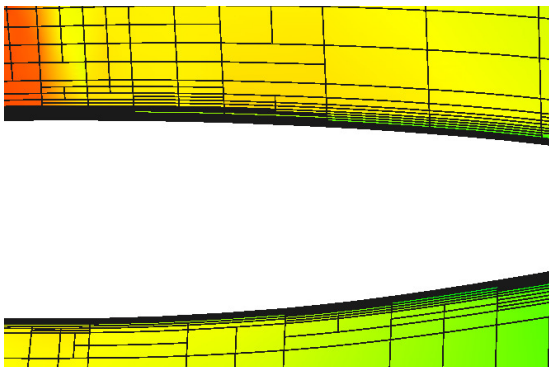
Figure 28: DPW Wing 1, $M_\infty = 0.76, \alpha = 0.5°, Re = 5 \times 10^6$: cut at $y = 620$mm of the drag-adapted meshes.

An optimization-based mesh adaptation algorithm may offer insight on "best-practice" gridding guidelines. We notice that several regions of the flow are frequently targeted for refinement. One of these regions is near the leading edge where the flow accelerates through the sonic condition. This change in character of the flow causes strong variations in the adjoint solution, which are responsible for large error indicators. Another region is the edge of the boundary layer, where the turbulent working variable, $\tilde{\nu}$, transitions to zero rapidly. The other two regions are the location of shock-boundary-layer interaction and the trailing edge. These regions exhibit strong gradients in $\tilde{\nu}$ that contribute to the drag output. Figure 29 shows the interaction between the shock and the boundary layer.

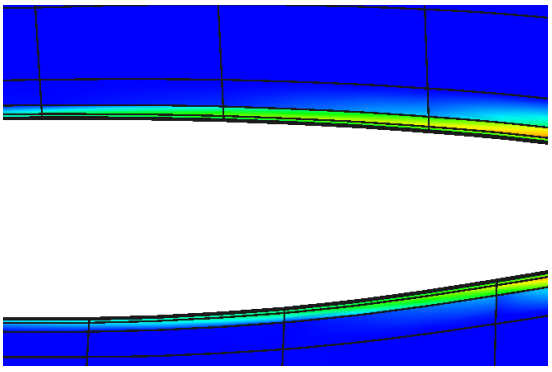
Note the concentration of cells in the boundary layer and the sharp variation of $\tilde{\nu}$. Further downstream, in the trailing edge region (Figure 30), the beginning of the turbulent wake is also adapted.



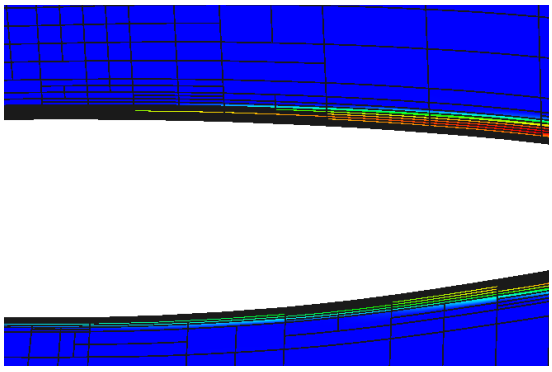
29.1: Mach contours for initial mesh.



29.2: Mach contours for the 7^{th} adapted mesh.

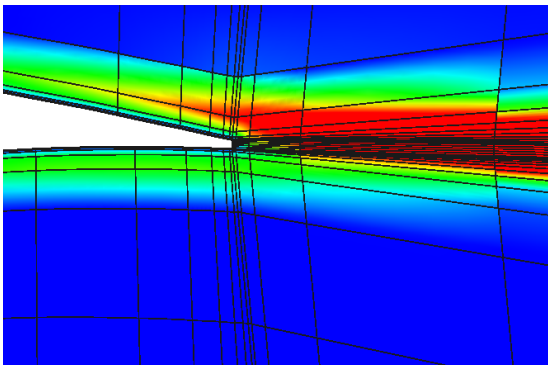


29.3: $\tilde{\nu}$ contours for initial mesh.

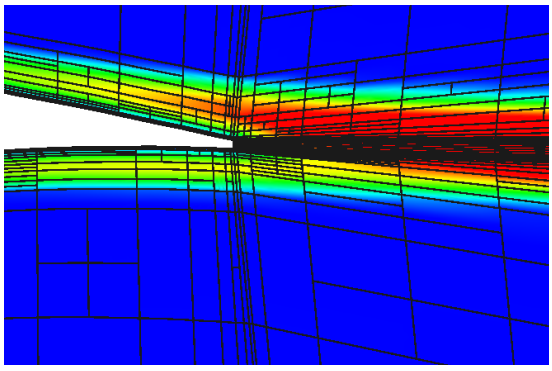


29.4: $\tilde{\nu}$ contours for the 7^{th} adapted mesh.

Figure 29: DPW Wing 1, $M_\infty = 0.76, \alpha = 0.5°, Re = 5 \times 10^6$: interaction between shock and boundary-layer at $y = 620$mm.



30.1: $\tilde{\nu}$ contours for initial mesh.



30.2: $\tilde{\nu}$ contours for the 7^{th} adapted mesh.

Figure 30: DPW Wing 1, $M_\infty = 0.76, \alpha = 0.5°, Re = 5 \times 10^6$: trailing edge at $y = 620$mm.

5 Unsteady Systems

Results in the previous section showed that adapting a mesh using an indicator obtained from an adjoint-weighted residual can improve robustness (through error estimates) and efficiency (through targeted refinement) of a steady-state CFD simulation. The motivation for such adaptation becomes even stronger in unsteady simulations, where we have control of both spatial and temporal resolution. Identifying the regions in both space and time that require adaptation is more challenging than identifying spatial regions alone. In addition, with an extra dimension, the percentage of the total space-time domain that is important for predicting an output is likely to be reduced compared to a steady problem, which improves the prospects of efficiency gains.

The idea of output-error estimation through an adjoint-weighted fine-space residual extends to unsteady problems without significant theoretical changes. However, unsteady problems do pose challenges for solver efficiency and adaptation mechanics. In this section we address some of these challenges and present adaptive results for reasonably-complex unsteady simulations.

5.1 Primal and Adjoint Discretizations

We discretize time using a semi-discrete approach, i.e. independently of the spatial discretization. Although our adaptive results employ a finite-element temporal discretization, we also present multi-step methods to simplify the presentation of the adjoint extension.

5.1.1 Multi-Step Methods

Primal Form: The discrete residual equation $\mathbf{R}(\mathbf{U}) = \mathbf{0}$, see (14), represents a steady system. A simple method of incorporating unsteady variations is to use a semi-discrete formulation, in which a PDE that is already discretized in space becomes

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = \mathbf{0}, \tag{48}$$

where $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix,

$$\mathbf{M}_{ij} = \mathbf{I}_s \int_\Omega \phi_i \phi_j \, d\Omega. \tag{49}$$

In the above expression, $\mathbf{I}_s \in \mathbb{R}^{s\times s}$ is the state identity matrix, and $1 \leq i, j \leq N$ index the global degrees of freedom. Note that in DG, ϕ_i will have support over only one element, which means that the mass matrix is element-wise block diagonal.

In a multi-step discretization, the time derivative in (48) is discretized using finite-differences. Denote by a superscript n or m the time nodes: e.g. in a uniform temporal subdivision, the time at each time node would be given by $t_m = m\Delta t$, where $\Delta t = T/N_t$ is the time step size for integration with N_t time intervals over a time $0 \leq t \leq T$. For example, a backward Euler method uses the approximation

$$\left.\frac{d\mathbf{U}}{dt}\right|_{t_m} \approx \frac{\mathbf{U}^m - \mathbf{U}^{m-1}}{\Delta t}, \tag{50}$$

so that the fully-discrete form of (48), evaluated at t^m, becomes

$$\underbrace{\mathbf{M}\frac{\mathbf{U}^m - \mathbf{U}^{m-1}}{\Delta t} + \mathbf{R}(\mathbf{U}^m)}_{\text{unsteady residual: } \mathbf{R}^m} = \mathbf{0}. \tag{51}$$

The initial condition sets the state at $t = t_0$. We have defined the unsteady residual, $\mathbf{R}^m$, as the left-hand side of the discrete equation. In general the unsteady residual will depend on states at several time nodes, and so we write $\mathbf{R}^m(\mathbf{U}^n)$. These N_t residual vectors have to be driven to zero (usually sequentially) to obtain the full unsteady solution.

Adjoint Form: Let's look at the adjoint for an unsteady multi-step discretization. The unsteady version of the sensitivity chain in (17) is

$$\boldsymbol{\mu} \rightarrow \mathbf{R}^m(\mathbf{U}^n, \boldsymbol{\mu}) = \mathbf{0} \rightarrow \mathbf{U}^n \rightarrow J(\mathbf{U}^n), \tag{52}$$

where, as in the primal form, $0 \leq n, m \leq N_t$ are the time indices in the unsteady discretization. The equations derived for the steady adjoint in Section 2.2 apply directly, but the vectors are now much larger (length $\mathbb{R}^{NN_t}$). The adjoint equation is

$$\sum_{m=1}^{N_t} \left(\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}^n}\right)^T \boldsymbol{\Psi}^m + \left(\frac{\partial J}{\partial \mathbf{U}^n}\right)^T = 0. \tag{53}$$

Due to the transpose on the unsteady Jacobian matrix, the adjoint system is most easily solved by marching backwards in time. For example, given a backward Euler temporal discretization, the unsteady Jacobian $\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}^n}$ and its transpose will look like ($\star = N \times N$ block),

Primal unsteady Jacobian **Adjoint unsteady Jacobian**

$$\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}^n} = \begin{bmatrix} \star & & & & & & \\ \star & \star & & & & & \\ & \star & \star & & & & \\ & & \star & \star & & & \\ & & & \star & \star & & \\ & & & & \star & \star & \\ & & & & & \star & \star \end{bmatrix} \qquad \left(\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}^n}\right)^T = \begin{bmatrix} \star & \star & & & & & \\ & \star & \star & & & & \\ & & \star & \star & & & \\ & & & \star & \star & & \\ & & & & \star & \star & \\ & & & & & \star & \star \\ & & & & & & \star \end{bmatrix}$$

We see that the primal Jacobian is zero above the main diagonal, making forward substitution in time the method of choice for solving the primal problem. On the other hand, the adjoint Jacobian is zero below the main diagonal, so that backward substitution is the natural solution strategy. Hence, an unsteady adjoint solution requires marching *backward* in time from the final state to the first state.

For nonlinear problems, the state history $\mathbf{U}^n$ needs to be stored for computing the linearizations in the adjoint solve. These states are typically written to disk, and solution checkpointing can alleviate storage requirements if they are burdensome [47]. Once the unsteady adjoint solution is available, the sensitivities are calculated as

$$\frac{dJ}{d\boldsymbol{\mu}} = \sum_{m=1}^{N_t} (\boldsymbol{\Psi}^m)^T \left(\frac{\partial \mathbf{R}^m}{\partial \boldsymbol{\mu}}\right). \tag{54}$$

5.1.2 Discontinuous Galerkin in Time

Primal Form: Instead of finite differences we could also consider finite elements for the temporal discretization. This is in fact a reasonable approach for output error estimation, which is most rigorous in a variational formulation. In particular, for our adaptive work, we use a discontinuous Galerkin method in time. This consists of time slabs on which the temporal solution variation is approximated with polynomials of order r. The term "time slab" is used only to emphasize that no local-in-space time stepping is performed, and that, at a given time, all elements advance at the same time step, Δt. This Δt is just the width of the current time slab, which can vary once adaptation is performed.

Figure 31 illustrates the concept of time slabs for a two-dimensional simulation. We

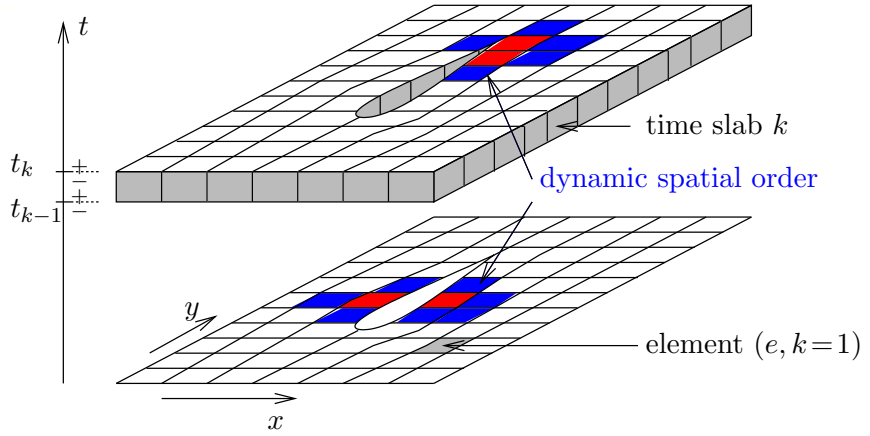


Figure 31: Illustration of time slabs and dynamic-order (depicted by color) refinement for a DG-in-time discretization.

enumerate elements in the spatial mesh by $1 \leq e \leq N_e$, and these are assumed fixed (not moving or refined) over the course of the simulation. We enumerate the time slabs by $1 \leq k \leq N_k$, where N_k is the total number of time slabs. Each space-time element is then identified by two indices, (e, k). p_e^k denotes the order of approximation in element e, time slab k, and this can vary across elements and time slabs.

On each space-time element (e, k), we approximate the state spatially using a standard DG-in-space approximation, as introduced in Section 2. The result of the spatial approximation is that on each element we have $N_{p_e^k}$ spatial unknowns, $\mathbf{U}_{he}^k$. In DG-in-time, we approximate the temporal variation of these unknowns using polynomials in time,

$$\mathbf{U}_{he}^k(t) \;=\; \sum_{n=1}^{r+1} \mathbf{U}_{he}^{kn} \varphi_h^n(t), \tag{55}$$

where $\varphi_h^n(t)$ are order r temporal basis functions. The subscript h on the above terms indicates the space-time mesh, i.e. the approximation space. When discussing error estimation, we will be dealing with coarse and fine spaces, which we will denote by H and h subscripts, respectively.

For compactness of notation, we lump all spatial degrees of freedom associated with time node n on slab k into one vector,

$$\mathbf{U}_h^{kn} = \left\{ \mathbf{U}_{he}^{kn} \right\}_{\forall e} \in \mathbb{R}^{N_{hk}}. \tag{56}$$

As a shorthand, we will denote by $\mathbf{U}_h^k$ the sets of unknowns over a whole time slab, k. Finally, the set of states over the entire space-time domain will be referred to simply as $\mathbf{U}_h$.

A nonlinear system of equations on each time slab is obtained by substituting the approximation from (55) into (48), multiplying by test functions in the same space as the approximation functions, and integrating by parts to incorporate discontinuities at time slab and spatial element interfaces. We group the resulting set of discrete equations by time nodes on each time slab into $(r+1)$ unsteady residual vectors, enumerated by m,

$$\mathbf{R}_h^{km} \equiv a^{mn}\mathbf{M}_h^{k,k}\mathbf{U}_h^{kn} - \varphi_h^m(t_{k-1})\mathbf{M}_h^{k,k-1}\mathbf{U}_h^{k-1,r+1} + \int_{t_{k-1}}^{t_k} \varphi_h^m(t)\mathbf{R}_h\left(\mathbf{U}_h^k(t)\right) dt = \mathbf{0}, \quad (57)$$

where $\mathbf{M}_h^{k,l} \in \mathbb{R}^{N_{hk}\times N_{hl}}$ is the spatial mass matrix formed by spatial basis functions on neighboring time slabs k and l, which could be different in cases of dynamic spatial order refinement. $\mathbf{U}_h^{k-1,r+1}$ is the end-of-slab state from the previous time slab, as shown in Figure 32. For the first time slab, $k=1$, this is $\mathbf{U}_h^{0,r+1} \equiv \mathbf{U}_h^0$, which is the initial condition.

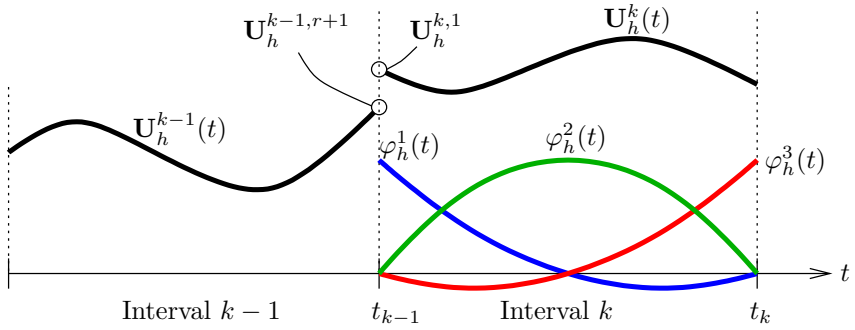


Figure 32: DG in time for $r=2$: depiction of temporal basis functions on one time slab and definitions of key quantities in (57). The vertical axis represents a "state" – from the spatial discretization, we actually have N_h states, and we can imagine having N_h such plots.

The values a^{mn} in (57) constitute a temporal stiffness matrix and are given by

$$a^{mn} = -\int_{t_{k-1}}^{t_k} \varphi_h^n \frac{d\varphi_h^m}{dt}\, dt + \varphi_h^n(t_k)\varphi_h^m(t_k). \quad (58)$$

Using a Lagrange temporal basis on equally-spaced nodes, we have,

$$\text{for } r=1,\ a^{mn} = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{bmatrix}; \quad \text{for } r=2,\ a^{mn} = \begin{bmatrix} 1/2 & 2/3 & -1/6 \\ -2/3 & 0 & 2/3 \\ 1/6 & -2/3 & 1/2 \end{bmatrix}. \quad (59)$$

The time integral in (57) is evaluated with $(r+1)$ points of Gauss quadrature for order $2r+1$ accuracy.

Adjoint Form: The discrete adjoint equation, (53), remains valid for a DG-in-time discretization. The number of time nodes is now $N_t = N_k(r+1)$, where N_k is the number of time slabs, since each time slab has $r+1$ temporal unknowns. Denoting the adjoint at time slab k, time node m, by $\mathbf{\Psi}_H^{km}$, a more descriptive version of (53) is

$$\underbrace{\left(\frac{\partial \mathbf{R}_h^{km}}{\partial \mathbf{U}_h^{ln}}\right)^T \mathbf{\Psi}_h^{km} + \left(\frac{\partial J_h}{\partial \mathbf{U}_h^{ln}}\right)^T}_{\mathbf{R}_{\psi h}^{ln}(\mathbf{\Psi}_h^{km})} = 0, \tag{60}$$

where k, l index time slabs and n, m index time nodes. We have defined the adjoint residual, $\mathbf{R}_{\psi h}^{ln}(\mathbf{\Psi}_h^{km})$, as the entire left-hand side of the adjoint equation. Linearizing the residual expressions in (57), the $r+1$ adjoint residual vectors on time slab k are

$$\mathbf{R}_{\psi h}^{ln} = a^{mn}\mathbf{M}_h^{l,l}\mathbf{\Psi}_h^{lm} - \varphi_h^n(t_l)\mathbf{M}_h^{l,l+1}\mathbf{\Psi}_h^{l+1,1} + \int_{t_{l-1}}^{t_l} \varphi_h^n \left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|_{\mathbf{U}_h(t)}^T \mathbf{\Psi}_h^l(t)\, dt + \left(\frac{\partial J_h}{\partial \mathbf{U}_h^{ln}}\right)^T \tag{61}$$

In this equation, $\mathbf{\Psi}_h^l(t) = \sum_m \mathbf{\Psi}_h^{l,m}\varphi_h^m(t)$, and $\mathbf{\Psi}_h^{l+1,1}$ is the adjoint vector associated with the start of the next time slab. When calculating on the last time slab, $\mathbf{\Psi}_h^{l+1,1} = 0$.

Both the primal and adjoint equations are solved using a Newton iteration based on an approximate factorization. This solver requires solutions of systems that are the same size as a steady-state solution. Details are given in [36].

5.2 Deformable Domains

5.2.1 An Arbitrary Lagrangian Eulerian Treatment

In an Arbitrary Lagrangian Eulerian (ALE) method, the mesh can move at a velocity different from that of the flow. This is useful for analyzing problems in which the computational domain undergoes deformation (e.g. flutter or flapping flight in aerodynamics), as in these situations the mesh generally needs to "move with" the geometry so that much of the computational domain may be affected.

The key idea of an ALE formulation is to map the original PDE on the deforming physical domain to a modified PDE on a static reference domain, as solving on a static domain is something we already know how to do. This transformation is illustrated graphically in Figure 33, and Table 2 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 [86]. The system of conservation laws on the physical domain is, repeating (1),

$$\partial_t \mathbf{u} + \partial_i \mathbf{H}_i(\mathbf{u}, \nabla\mathbf{u}) = \mathbf{0}, \qquad \mathbf{H}_i = \mathbf{F}_i(\mathbf{u}) - \mathbf{G}_i(\mathbf{u}, \nabla\mathbf{u}), \tag{62}$$

where both inviscid and viscous fluxes are included. Integrating over a time-varying volume $v(t)$ yields,

$$\int_{v(t)} \partial_t \mathbf{u} dv + \int_{\partial v(t)} \vec{\mathbf{H}} \cdot \vec{n}\, da = 0, \qquad \vec{n} \text{ is outward-pointing on } \partial v(t), \tag{63}$$

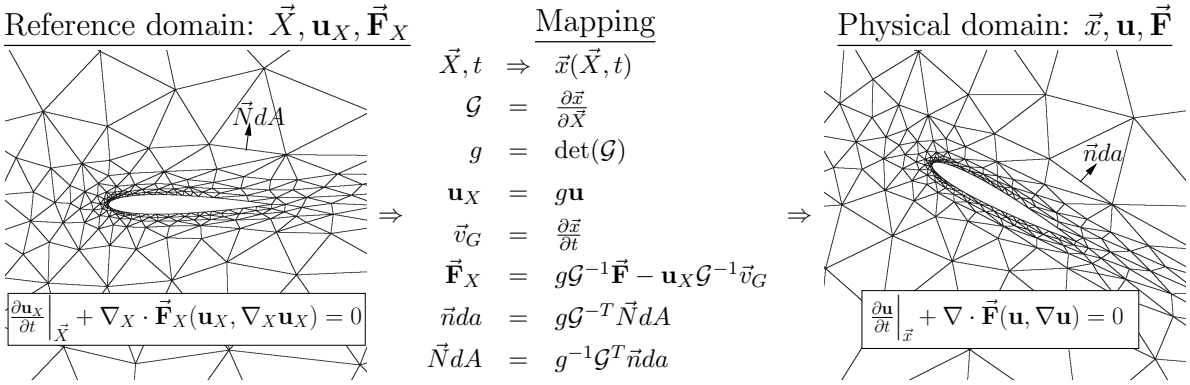


Figure 33: Summary of the mapping between reference and physical domains. The equations are solved on the reference domain, which remains fixed for all time. When denoting reference-domain quantities, we use a subscript X.

Table 2: 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
$\vec{\mathbf{F}}_X$	=	flux vector on reference domain	$\vec{\mathbf{F}}$	=	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			

where $\vec{\mathbf{H}}$ is a spatial vector with components $\mathbf{H}_i$. We now transform to the reference domain, V. The boundary integral of the flux is

$$\int_{\partial v(t)} \vec{\mathbf{H}} \cdot \vec{n}\, da = \int_{\partial V} \vec{\mathbf{H}} \cdot (g\mathcal{G}^{-T}\vec{N})\, dA = \int_{\partial V} (g\mathcal{G}^{-1}\vec{\mathbf{H}}) \cdot \vec{N}\, dA. \tag{64}$$

The first integral in (62) transforms using Leibniz's rule,

$$\begin{aligned} \int_{v(t)} \frac{\partial \mathbf{u}}{\partial t} &= \frac{d}{dt}\int_{v(t)} \mathbf{u}\, dv - \int_{\partial v(t)} (\mathbf{u}\vec{v}_G) \cdot \vec{n}\, da \\ &= \frac{d}{dt}\int_V \mathbf{u} g\, dV - \int_{\partial V} (\mathbf{u}\vec{v}_G) \cdot (g\mathcal{G}^{-T}\vec{N})\, dA \\ &= \int_V \frac{\partial (g\mathbf{u})}{\partial t}\, dV - \int_{\partial V} (g\mathbf{u}\mathcal{G}^{-1}\vec{v}_G) \cdot \vec{N}\, dA. \end{aligned} \tag{65}$$

Substituting (64) and (65) into (63) and applying the divergence theorem gives the PDE on the reference domain,

$$\begin{aligned} \left.\frac{\partial \mathbf{u}_X}{\partial t}\right|_{\vec{X}} + \nabla_X \cdot \vec{\mathbf{H}}_X(\mathbf{u}_X, \nabla_X \mathbf{u}_X) &= 0, \\ \text{where} \qquad \mathbf{u}_X &= g\mathbf{u}, \\ \vec{\mathbf{H}}_X &= g\mathcal{G}^{-1}\vec{\mathbf{H}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G. \end{aligned} \tag{66}$$

∇_X denotes the gradient with respect to the reference coordinates. We break up the transformed flux, $\vec{\mathbf{H}}_X$, into inviscid and viscous fluxes by lumping the grid-velocity term into the inviscid flux,

$$\vec{\mathbf{H}}_X = \vec{\mathbf{F}}_X - \vec{\mathbf{G}}_X, \qquad \vec{\mathbf{F}}_X = g\mathcal{G}^{-1}\vec{\mathbf{F}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G, \qquad \vec{\mathbf{G}}_X = g\mathcal{G}^{-1}\vec{\mathbf{G}}. \tag{67}$$

The gradient of the state transforms via the chain and product rules. Using implied summation,

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial x_j} = \frac{\partial (g^{-1}\mathbf{u}_X)}{\partial X_d}\frac{\partial X_d}{\partial x_j} &= \left(g^{-1}\frac{\partial \mathbf{u}_X}{\partial X_d} - g^{-2}\frac{\partial g}{\partial X_d}\mathbf{u}_X\right)\mathcal{G}^{-1}_{dj} \\ &= g^{-1}\left(\frac{\partial \mathbf{u}_X}{\partial X_d} - g^{-1}\frac{\partial g}{\partial X_d}\mathbf{u}_X\right)\mathcal{G}^{-1}_{dj}, \end{aligned} \tag{68}$$

where d and j index the reference and physical coordinates, respectively. We also have,

$$\mathcal{G} = \mathcal{G}_{jd} = \frac{\partial x_j}{\partial X_d}, \qquad \delta_{ji} = \frac{\partial x_j}{\partial x_i} = \frac{\partial x_j}{\partial X_d}\frac{\partial X_d}{\partial x_i} = \mathcal{G}_{jd}\mathcal{G}^{-1}_{di} \quad \Rightarrow \quad \mathcal{G}^{-1} = \mathcal{G}^{-1}_{di} = \frac{\partial X_d}{\partial x_i}.$$

In a DG setting, discretization of the new reference-domain equation requires modifications to the numerical flux function on inter-element faces, to the boundary conditions, to the face normal vectors, and to the quadrature integration weights. These modifications are based on the reference-to-global mapping and its derivatives.

The weighted residual statement on the reference domain is obtained from the PDE by multiplying by test functions (defined in the reference domain) and integrating over

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. A natural approach that minimizes intrusion into the code is to express the reference-space fluxes and boundary conditions in terms of the physical fluxes and boundary conditions.

For example, 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,

$$\vec{\mathbf{F}}_X = g\mathcal{G}^{-1}\vec{\mathbf{F}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G = g\mathcal{G}^{-1}\left(\vec{\mathbf{F}} - \mathbf{u}\vec{v}_G\right). \tag{69}$$

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

$$\vec{\mathbf{G}}_X = g\mathcal{G}^{-1}\vec{\mathbf{G}}. \tag{70}$$

Since the physical viscous flux is calculated using a diffusion matrix and the physical state gradient, $\mathbf{G}_i = \mathbf{K}_{ij}\partial_j\mathbf{u}$, then, using (68) for the physical gradient, the reference-domain viscous flux is

$$\begin{aligned}\mathbf{G}_{X,d} &= g\mathcal{G}^{-1}_{di}\mathbf{K}_{ij}\partial_{x_j}\mathbf{u} \\ &= g\mathcal{G}^{-1}_{di}\mathbf{K}_{ij}g^{-1}\left(\partial_{X_c}\mathbf{u}_X - \mathbf{u}_X g^{-1}\partial_{X_c}g\right)\mathcal{G}^{-1}_{cj} \\ &= \underbrace{\mathcal{G}^{-1}_{di}\mathbf{K}_{ij}\mathcal{G}^{-1}_{cj}}_{\mathbf{K}_{X,dc}}\left(\partial_{X_c}\mathbf{u}_X - \mathbf{u}_X g^{-1}\partial_{X_c}g\right),\end{aligned} \tag{71}$$

where c, d index the reference domain coordinates. $\mathbf{K}_{X,dc}$ represents the diffusion matrix on the reference domain. It can be re-written in a more symmetrical form as

$$\mathbf{K}_{X,dc} = \mathcal{G}^{-1}_{di}\mathbf{K}_{ij}\mathcal{G}^{-1}_{cj} = \mathcal{G}^{-1}_{di}\mathbf{K}_{ij}\mathcal{G}^{-T}_{jc}. \tag{72}$$

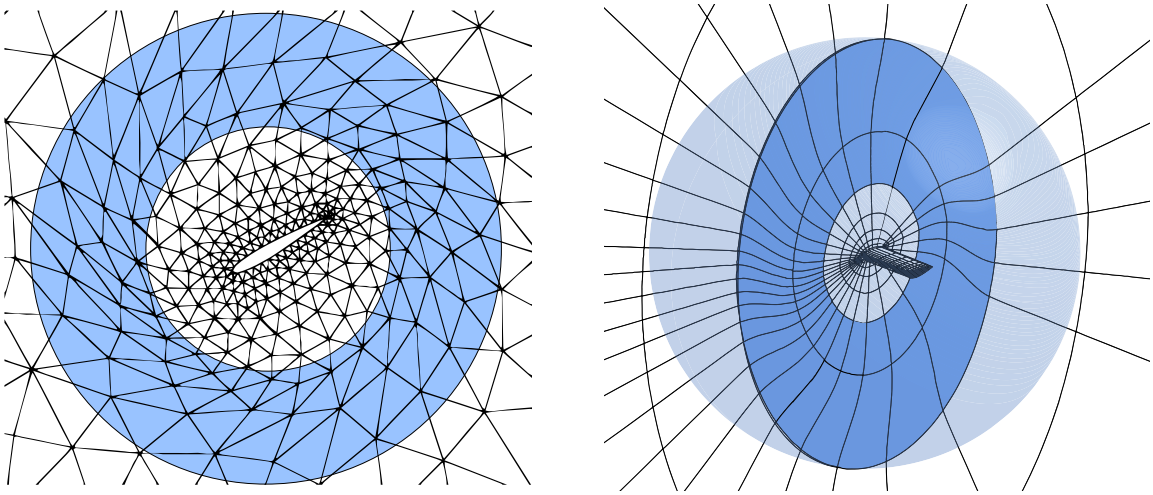
Boundary conditions also require modifications when simulating problems on deformable domains. In particular, the physical boundary flux must be aware of motion on the boundary, $\vec{v}_G$. For example, on a moving wall, the flow tangency boundary condition states that the normal component of the fluid 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 the viscous flux dotted with the normal, since

$$\vec{\mathbf{G}}\cdot\vec{n}da = \left(g^{-1}\mathcal{G}_{id}\vec{\mathbf{G}}_X\right)\cdot\left(g\mathcal{G}^{-1}\vec{N}\right)dA = \vec{\mathbf{G}}_X\cdot\vec{N}dA. \tag{73}$$

5.2.2 Blended Analytical Mesh Motions

The ALE method described above requires an analytically defined mapping between reference and physical domains. Therefore, the user must prescribe a motion (e.g. sinusoidal pitch/plunge) in a certain region of the domain. If only a portion of the domain needs to move, the mapping can be smoothly blended into the static domain outside the moving region. As Persson *et al* present in [86], a polynomial blending function is a simple way



34.1: Airfoil 34.2: Wing

Figure 34: Airfoil and wing undergoing analytical motions. The blue regions are those in which the prescribed inner motion is blended into the static outer mesh. The boundaries of these blending regions are circular in 2D and spherical in 3D.

to transition between deforming and static regions. A typical scenario is to have an inner disk in 2D (or sphere in 3D) undergo a prescribed rigid-body motion, and to then blend this motion into the static mesh via the polynomial blending function. Figure 34 shows an example of this blending for an airfoil and a wing.

5.2.3 The Geometric Conservation Law

The ability to preserve a free stream is a desirable property of numerical schemes. However, for schemes employing a finite-dimensional basis (say a set of polynomials in space-time), a constant state $\bar{\mathbf{u}}$ in the physical domain will generally not be a solution to the discrete form of (66) in the reference domain. This means that for an arbitrary motion of the mesh, an initially free-stream state will not be preserved.

This lack of free-stream preservation can be explained by noting that, for general mappings, the Jacobian g will be non-polynomial in both space and time. Hence, the reference state $\mathbf{u}_X = g\bar{\mathbf{u}}$ will likewise be non-polynomial, which means it cannot be represented exactly with the reference-domain bases. This inexact representation will introduce both spatial and temporal errors into an initially free-stream state, which manifest as conservation errors that accumulate in time.

To eliminate these conservation errors, a separate Geometric Conservation Law (GCL) is enforced alongside the governing equations. The idea of the GCL is twofold: (i) to address the representation issues mentioned above, replace the analytical g with a new variable, $\bar{g}$, which is a polynomial approximation to g in space-time; and (ii) to ensure that this $\bar{g}$ actually allows for free-stream preservation, compute it from the following equation [86]:

$$\frac{\partial \bar{g}}{\partial t} - \nabla_X \cdot (g\mathcal{G}^{-1}\vec{v}_G) = 0. \tag{74}$$

This equation ensures that the change in element area (i.e. the change in $\bar{g}$) is directly linked to what the grid velocities on element boundaries claim it should be. Hence, there is no disagreement between grid velocities and Jacobians on what the geometry is, and in that sense we have "geometric conservation."

The strategy then is to use this $\bar{g}$ to define a new reference-domain state $\mathbf{u}_{\bar{X}} = \bar{g}g^{-1}\mathbf{u}_X = \bar{g}\mathbf{u}$, which is used instead of the original state $\mathbf{u}_X = g\mathbf{u}$. If $\bar{g}$ is discretized using the same spatial basis as the state and is marched in time using the same unsteady solver, a free-stream state $\bar{\mathbf{u}}$ will be preserved. In the end, what we have done is replaced the original analytical g with a "best fit" space-time polynomial $\bar{g}$, which then makes the free-stream state $\mathbf{u}_{\bar{X}} = \bar{g}\bar{\mathbf{u}}$ exactly representable in the discrete space.

Once $\bar{g}$ is obtained on each element, it is used instead of g to convert the stored reference state to the physical state. The final form of the reference-domain equation is then

$$\left.\frac{\partial \mathbf{u}_{\bar{X}}}{\partial t}\right|_X + \nabla_X \cdot \vec{\mathbf{H}}_{\bar{X}}(\mathbf{u}_{\bar{X}}, \nabla_X \mathbf{u}_{\bar{X}}, \bar{g}) \;=\; 0, \tag{75}$$

where $\vec{\mathbf{H}}_{\bar{X}}$ is just $\vec{\mathbf{H}}_X$ but with $\mathbf{u}_{\bar{X}}/\bar{g}$ replacing $\mathbf{u}_X/g$ in the calculation of the physical state.

Figure 35 shows the effect of the GCL on a free-stream preservation test. In this case, we solve the Navier-Stokes equations on a rectangular domain with an analytical sinusoidal deformation mapping defined in the domain interior. The temporal and spatial discretization are both discontinuous Galerkin, with order $r = 1$ and p, respectively. Without the GCL, the free-stream solution is not preserved, though the L_2 error converges with both h- and p-refinement of the spatial mesh. With the GCL, the free stream is maintained to residual tolerance, which was approximately ten orders of magnitude for these runs. To achieve this level of accuracy, high order quadrature rules are required, with rules of order $6p$ used in this case to demonstrate the GCL's full effect. In practical cases we use more modest rules, namely $2p + 5$ for the Navier-Stokes equations, since discretization errors tend to dominate and make high quadrature rules unnecessary.

Finally, note that (74) has introduced an additional numerical quantity, $\bar{g}$, which is subject to discretization errors just like the state quantities, and this is relevant for error estimation.

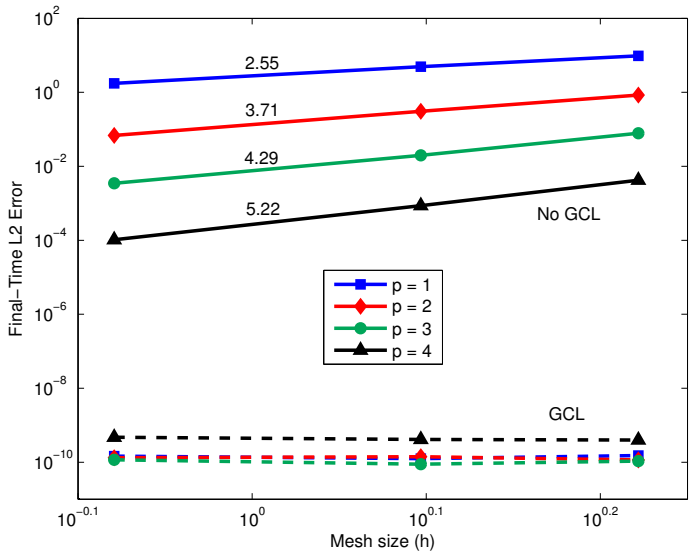


Figure 35: Free-stream errors with and without the GCL. A large number of time steps is used so that the spatial error dominates the temporal error in each case.

5.3 Error Estimation and Adaptation

5.3.1 The Adjoint-Weighted Residual and Error Localization

The adjoint-weighted residual output error estimate in (40) extends directly to unsteady problems,

$$\delta J \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) = -\sum_{k=1}^{N_k} \sum_{m=1}^{r+1} \left(\mathbf{\Psi}_h^{km}\right)^T \mathbf{R}_h^{km}(\mathbf{U}_h^H). \tag{76}$$

The effect of the GCL can be incorporated into the above formula by extending the state, residual, and adjoint vectors to include the GCL variable and equation. The contribution of the GCL to the error estimate is important because, when the GCL is used, the GCL residuals tell us where the mesh or time step should be refined to more accurately represent the true motion [64].

To localize the error contributions to individual space-time elements in the mesh, we note that the output error estimate in (76) can be written as a sum over all space-time elements,

$$\delta J \approx \sum_{k=1}^{N_k} \sum_{e=1}^{N_e} \varepsilon_e^k, \tag{77}$$

where the error contribution of a given space-time element (e, k) is, assuming a temporal order of r,

$$\varepsilon_e^k = \sum_{m=1}^{r+1} \left(-\mathbf{\Psi}_{he}^{km}\right)^T \mathbf{R}_{he}^{km} \left(\mathbf{U}_h^H\right), \tag{78}$$

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

$$\text{error indicator for element } e \text{ of time slab } k = \epsilon_e^k = \left|\varepsilon_e^k\right|. \tag{79}$$

We are sometimes also interested in a "conservative" error estimate given by the sum of the individual error indicators,

$$\text{sum of error indicators} = \epsilon = \sum_{k=1}^{N_k} \sum_{e=1}^{N_e} \epsilon_e^k. \tag{80}$$

5.3.2 Incorporating Space-Time Anisotropy

The indicator in (79) is sufficient for isotropic space-time refinement in which elements are targeted for refinement in both space and time. However, isotropic refinement can produce inefficient meshes. For example, if the discretization is under-resolved in the time domain, spatial elements would needlessly be refined on account of the temporal error. Therefore, important for an efficient adaptation strategy is a measure of the space-time anisotropy of the error; in other words, is the error on a given space-time element due primarily to the *spatial* or *temporal* discretization?

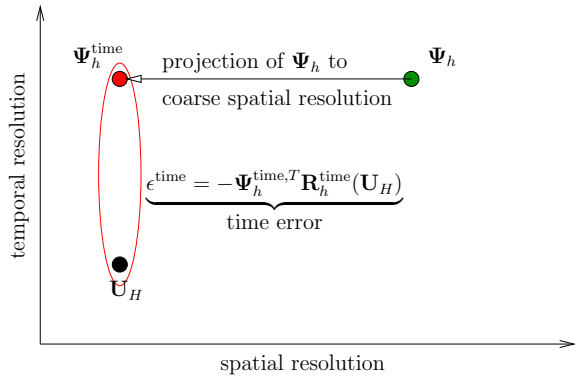
As a heuristic measure of space-time anisotropy, one could consider inter-element jumps in the solution. For each space-time element, the average jump in the state computed across the spatial interfaces and across the time slab interfaces could serve as an estimate of which direction (space or time) is better resolved. Although cheap to evaluate, such a heuristic requires normalization and some arbitrary decisions, especially for systems of equations.

We can eliminate some of the heuristics by basing the anisotropy measure on the output error estimate [32, 33]. Specifically, we calculate the error anisotropy using separate projections of the fine-space adjoint onto semi-coarsened spatial and temporal spaces, as illustrated in Figure 36. The spatial and temporal error estimates for space-time element (e, k) are obtained by using these projected adjoints in (77), resulting in separate $\varepsilon_e^{k,\text{space}}$ and $\varepsilon_e^{k,\text{time}}$ estimates. We then use the ratio of these values to estimate the fractions (β) of spatial and temporal error on element (e, k) as

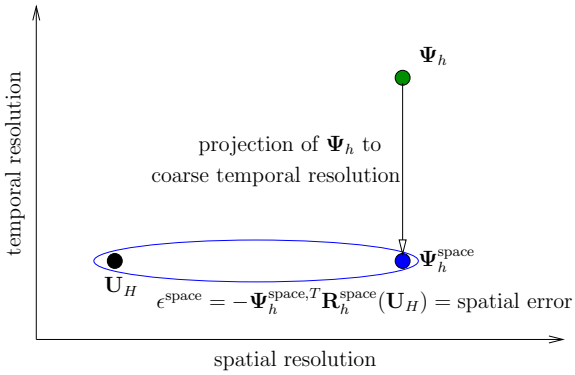
$$\beta_e^{k,\text{space}} = \frac{\left|\varepsilon_e^{k,\text{space}}\right|}{\left|\varepsilon_e^{k,\text{space}}\right| + \left|\varepsilon_e^{k,\text{time}}\right|}, \qquad \beta_e^{k,\text{time}} = 1 - \beta_e^{k,\text{space}}. \tag{81}$$

5.3.3 Space-Time Mesh Adaptation

With the above estimates of spatial and temporal error, we have the fundamental information needed for adaptation. We adapt by resolving the unsteady problem several times on successively refined space-time meshes, as illustrated in Figure 37. After each primal solve, the adjoint equations are marched backward in time, new error indicators are obtained, and a new adapted mesh is generated. This process is repeated until the output error drops below a specified tolerance. A key choice in adaptation is the decision of *what* to adapt. First, to address temporal errors, we could refine the time-step on each



36.1: Estimation of $\varepsilon^{\text{time}}$



36.2: Estimation of $\varepsilon^{\text{space}}$

Figure 36: Measure of space-time anisotropy via projections of the fine-space adjoint into semi-coarsened spaces. In this work, the coarse spatial and temporal resolutions are obtained by order decrement, and the projection is least-squares.

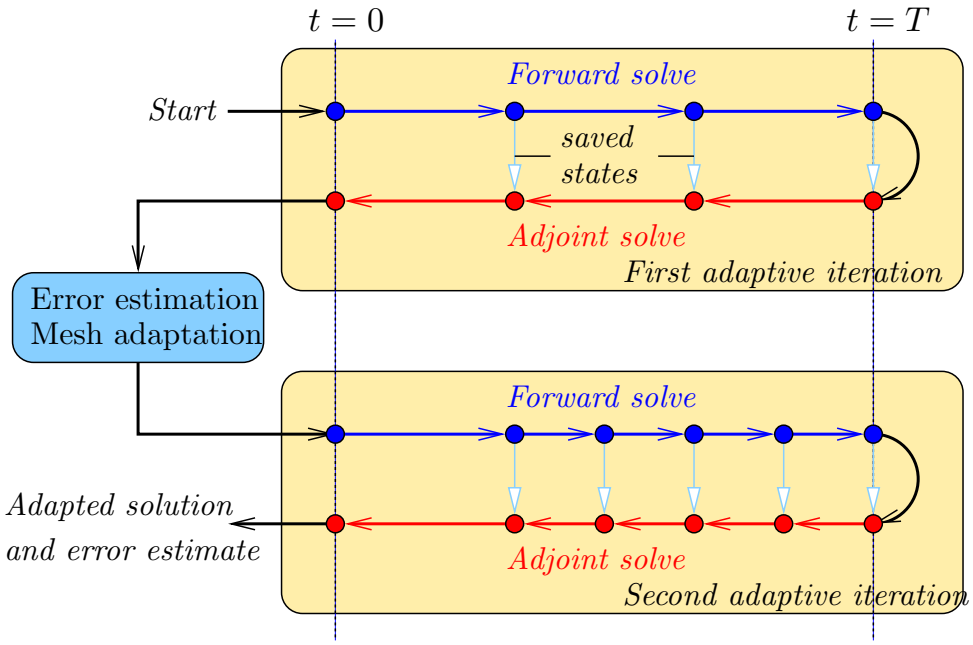
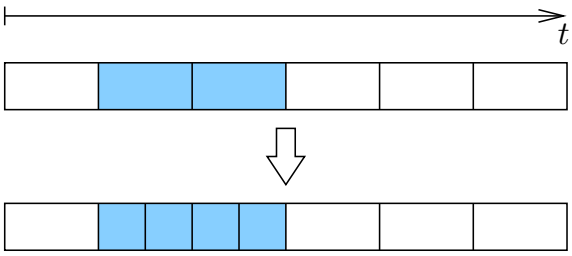


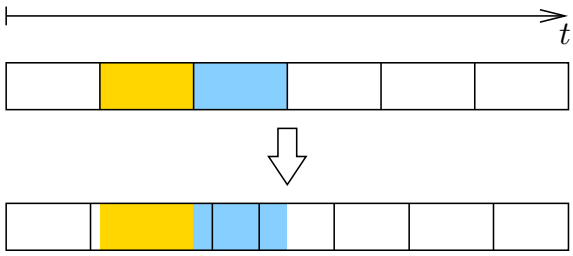
Figure 37: Adaptive solution process for unsteady problems. Adjoint vectors are written during each adaptive iteration, which consists of a forward and adjoint solve.

space-time element separately, and this would require a type of hanging-node treatment in time. However, to enable efficient solution of the DG-in-time discrete system by our approximate Newton solver, we limit the temporal refinement to entire time slabs, as described below. Second, to address spatial errors, we can adapt a mesh and leave it adapted for the course of the simulation (static spatial refinement) or we can devise an adaptive schedule in which the refinement changes during the course of the simulation (dynamic refinement). We consider both options, as described below.

Temporal Refinement: To address temporal errors, we adapt the DG-in-time discretization by adjusting the size of each time slab. When adaptation consists of only refinement (no coarsening), then this adjustment consists of simple bisection of those time slabs targeted for refinement, as shown in Figure 38.1. However, when coarsening also occurs, the boundaries of a coarsened slab will typically encroach on those of the neighboring slabs, and the entire temporal grid must be shuffled to make room for the coarsened slab, as illustrated in Figure 38.2. To perform this shuffling, time slabs are redistributed using one-dimensional metric-based meshing, the details of which are given in Appendix B.



38.1: Time slab bisection



38.2: Time-slab shuffling

Figure 38: Temporal mesh refinement using bisection and shuffling. Time slabs shaded in blue are flagged for refinement, and those in gold are flagged for coarsening.

Static Spatial Refinement: One simplification that minimizes storage and complexity of the data structures is to have refinement of the spatial mesh remain fixed throughout the unsteady simulation. That is, the number/location of elements and approximation order on each element are both constant in time, as illustrated in Figure 39. Although this can limit efficiency of the adaptation, especially for problems that exhibit spatially-localized sources of error that move in time, there are many problems for which such a static refinement strategy works well.

In static spatial refinement, the "objects" to be adapted are (i) spatial elements and (ii) time slabs. Adaptive indicators identifying the spatial error on each spatial element

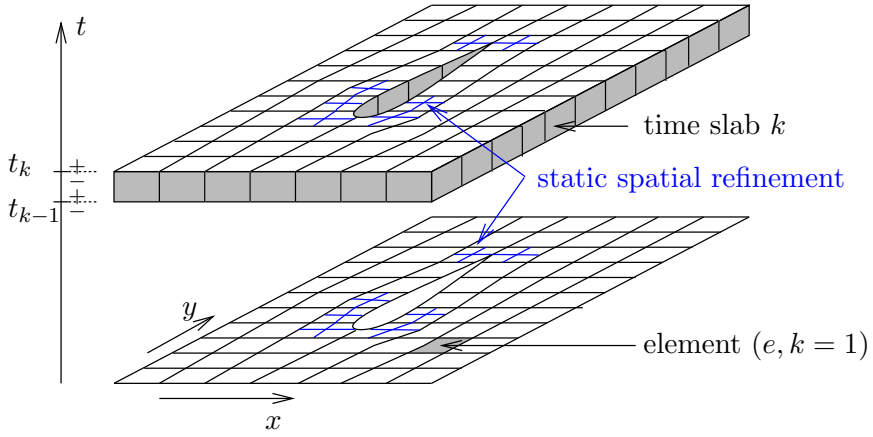


Figure 39: Illustration of static spatial refinement for a DG-in-time discretization, using hanging-node spatial mesh refinement. Static order refinement follows a similar approach.

and the temporal error on each slab are given by

$$\text{aggregate spatial indicator on element } e = \quad \epsilon_e^{\text{space}} \quad = \sum_{k=1}^{N_k} \epsilon_e^k \beta_e^{k,\text{space}}, \tag{82}$$

$$\text{aggregate temporal indicator on time slab } k = \quad \epsilon^{k,\text{time}} \quad = \sum_{e=1}^{N_e} \epsilon_e^k \beta_e^{k,\text{time}}, \tag{83}$$

With these indicators, we could then conceivably lump all time slabs and space-time elements into the same "bin," rank them according to their error indicators, and determine which to adapt based on their relative positions in that ranking. However, this approach would neglect the *cost* of each refinement option. For example, if a time slab and a spatial element had the same error but there were many more time slabs than spatial elements, one would not want to weigh refining the time slab equally compared to refining the element. Rather, refining the time slab would be cheaper because it would add fewer degrees of freedom compared to refining a spatial element. Thus, rather than adapting directly on errors, we adapt on a slightly different figure of merit – the amount of error on a given element or slab (Eqns. 82 and 83) *divided by* the additional degrees of freedom associated with adapting that element or slab. This figure of merit then ensures that we eliminate the most output error for the least additional cost.

To first approximation, the number of degrees of freedom introduced in a time slab division is estimated as the number of spatial degrees of freedom in the current mesh, while the number of degrees of freedom introduced in a spatial refinement is estimated as N_k times the number of new spatial degrees of freedom obtained from a refinement of that element. The respective adaptive indicators $\epsilon^{k,\text{time}}$ and $\epsilon_e^{\text{space}}$ are divided by these quantities and then sorted highest to lowest. The element or time slab with the highest error indicator per proposed additional number of degrees of freedom is chosen for refinement first, and the process continues until a growth budget is reached or surpassed. This is effectively a fixed-growth adaptation strategy. We note that in some cases the refinement could target only spatial elements or only time slabs, depending on the relative resolution in time and space.

Dynamic Spatial Refinement: In dynamic spatial refinement, the spatial resolution of an element can change during the course of the simulation. This change could arise from hanging node refinement, mesh motion, or order refinement. We focus on order refinement, illustrated in Figure 31, as it is one of the simplest options for a DG discretization in space and time.

Recall that p_e^k is the order of spatial approximation on spatial element e at time slab k. So the "objects" to be adapted are both (i) individual space-time elements in order and (ii) time slabs. Adaptive indicators identifying the spatial error on each space-time element and the temporal error on each slab are given by

$$\text{spatial indicator on space-time element } e,k = \quad \epsilon_{ek}^{\text{space}} \quad = \epsilon_{ek}\beta_{e,k}^{\text{space}}, \tag{84}$$

$$\text{aggregate temporal indicator on time slab } k = \quad \epsilon^{k,\text{time}} \quad = \sum_{e=1}^{N_e} \epsilon_{e,k}\beta_{e,k}^{\text{time}}, \tag{85}$$

These definitions are similar to the static refinement case, except that now the spatial indicator exists for every space-time element, instead of just as a sum over time slabs for each element. As in static spatial refinement, we make a decision of what to adapt based on a figure of merit that incorporates both the localized error estimate and the cost (in terms of additional degrees of freedom) of the refinement operation. We now also allow for order decrease, i.e. coarsening of the spatial mesh, in addition to refinement. More details on this algorithm are given in Appendix B.

5.3.4 Implementation Notes

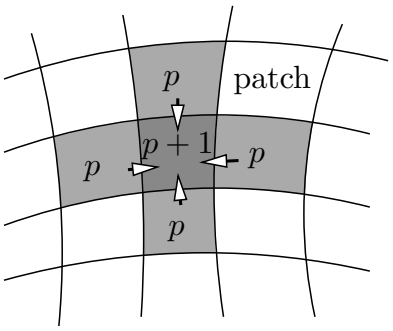
The adjoint equations require several derivative terms, including residual Jacobians and output linearizations. In the current work, we perform all differentiation analytically, with the exception of some terms required when using the geometric conservation law, which we evaluate using finite differences for ease of implementation.

When solving the adjoint equations, we use the entire time history of the primal state and GCL variable, which we store to disk during the primal solve. While for the present work this storage has not been prohibitive, for larger problems solution checkpointing [47] or local-in-time adjoint solvers [111] may be considered.

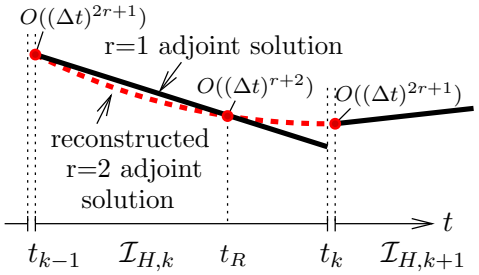
The space-time error indicators are computed on the fly during the backwards time-marching solution of the unsteady adjoint problem. Following each unsteady adaptive iteration, the error indicators are sorted and space-time elements are identified for refinement and coarsening. A schedule of orders p_e^k and time slab sizes is then written to disk for the next adaptive iteration. This schedule takes the form of individual files that are read in by the forward solver during the next forward solve.

Error estimation requires that the adjoint be solved in an enriched space, and for our work we use order refinement in both space and time: $p \to p+1$, and $r \to r+1$. For large (e.g. 3D) simulations, the cost of an adjoint solution on this space may become prohibitive due to memory limitations associated with the linear solve (we store the full residual Jacobian matrix). To address this issue, in these cases we employ a reconstruction-based strategy in lieu of computing the fine-space adjoint directly. That is, we compute the adjoint in the same space as the primal problem, and then reconstruct it in both space and time to obtain an approximation to the fine-space adjoint. This reconstruction is

performed locally, with patches of neighboring elements used to generate a least-squares spatial reconstruction, and pairs of neighboring time slabs used to obtain a high order temporal interpolant [32], as illustrated in Figure 40.



40.1: Spatial reconstruction



40.2: Temporal reconstruction

Figure 40: Illustration of spatial and temporal adjoint reconstructions. On the left, a patch of nearest-neighbor elements used for spatial high order reconstruction via least-squares interpolation. On the right, reconstruction of an $r = 1$ adjoint to $r = 2$ using the left node from the adjacent future time slab and superconvergent nodes on the current time slab. t_R indicates the root of the left Radau polynomial for $r = 1$.

5.4 Examples

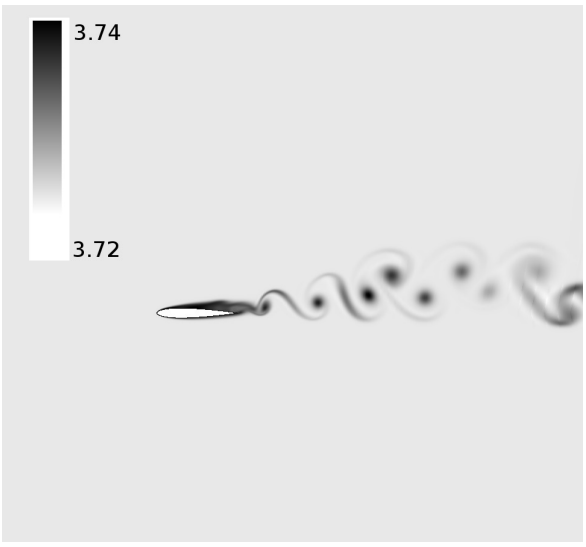
5.4.1 Static h-Refinement for an Impulsively-Started Airfoil

In this example we demonstrate the performance of static hanging-node spatial mesh refinement together with time slab bisection for an unsteady computation on a static domain. The adaptive mechanics consist of only refinement (no coarsening) in space and time, driven by a prescribed fixed-growth of degrees of freedom, $f^{\text{growth}} = 2$, per adaptive iteration. Adaptation using the output-based error indicator is compared to several other strategies: adaptation using an indicator obtained from an unweighted residual; adaptation driven by a measure of inter-element jumps; and uniform space and time refinement. Details on these adaptive strategies are given in [32].

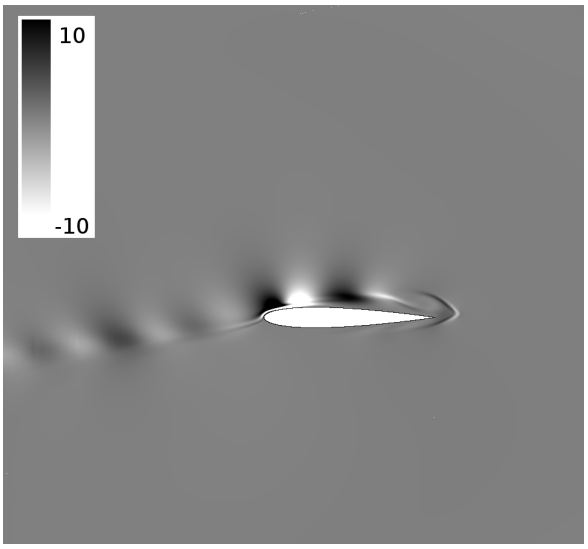
We consider an impulsively-started NACA 0012 airfoil in viscous flow, where for $t \geq 0$, the freestream conditions are $M_\infty = 0.25$, $\alpha = 8°$, $Re = 5000$. To prevent a non-physical step change in the velocity of the fluid at the airfoil surface, the initial condition at $t = 0$ consists of the freestream with the velocity blended smoothly to zero in a circular disk around the airfoil. Specifically, the velocity in the blended region, $r_1 \leq r \leq r_2$ is $\mathbf{v} = \mathbf{V}_\infty(1 - \cos(\pi(r - r_1)/(r_2 - r_1)))/2$ where r_1 and r_2 are radial distances from the airfoil mid-chord (set to one and three chord lengths respectively) and $\mathbf{V}_\infty$ is the freestream velocity. No steady solve is performed prior to the unsteady simulation.

Figure 41a shows the entropy contours at $t = 10$ units, the final time in the simulation. By this time an alternating pattern of shed vortices has developed and is clearly visible. The output of interest is the lift coefficient integral from $t = 9$ to $t = 10$, as illustrated in Figure 42a. A snapshot adjoint solution for the y-momentum equation at $t = 6$ is

illustrated in Figure 41b. A "reverse wake" is evident in the adjoint solution, signifying an oscillatory sensitivity of the output to y-momentum residual perturbations upstream.



41.1: Entropy contours at $t = 10$



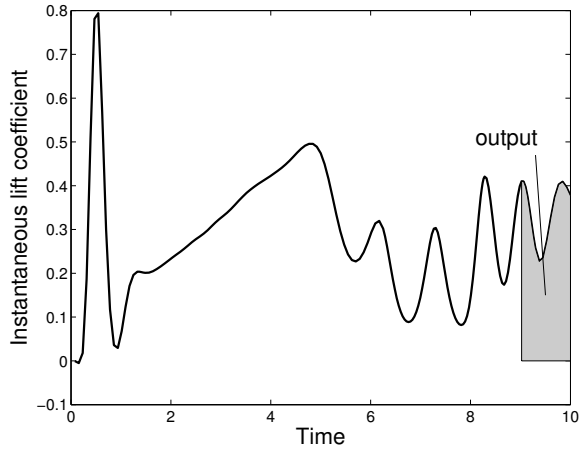
41.2: y-momentum adjoint at $t = 6$

Figure 41: Impulsively-started airfoil: primal state at the final time and the adjoint state at $t = 6$.

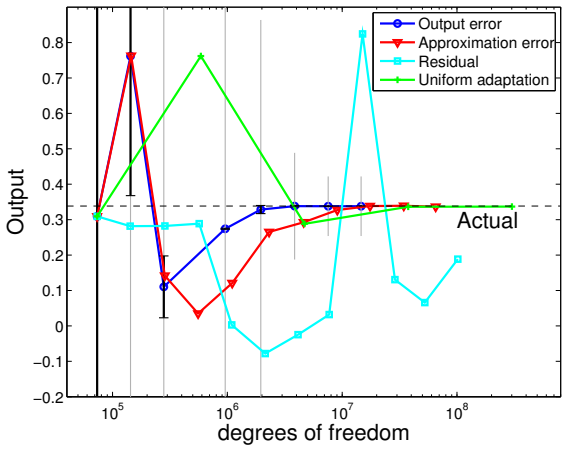
For the adaptive runs, an initial spatial mesh of 510 elements is used, and the initial temporal mesh contains 16 time slabs. The output convergence for the various indicators is shown in Figure 42b. The residual indicator does not perform well at all again: the output varies significantly from iteration to iteration. The other indicators converge, with the fastest being output-based adaptation, followed by approximation error and then uniform refinement. The advantage of the output-based refinement with degrees of freedom is a factor of 3-4 savings over the approximation error indicator. The error estimates under-predict the error in the middle stages of output-based refinement, while the conservative whiskers at $\pm\epsilon$ are more robust.

Figure 43a shows the time histories of the lift coefficient for adapted space-time meshes of similar size. The source of the error in the residual-adapted case is clear: it does not predict oscillatory vortex shedding, but rather an increasing lift coefficient. The other three adaptive indicators track the actual time history well. Figure 43b shows the L_2 time history error convergence for all of the methods, versus the cube root of the degrees of freedom. Under uniform refinement, the observed convergence rate is a suboptimal 1.5, which could be caused by high-order derivative discontinuities in the blended initial condition or the resolution not yet being in the asymptotic regime. Of primary interest, however, is the performance of output-based adaptation, which remains the fastest out of the methods tested. Even though the output is only measured on the final 10% of the simulation time, accurate resolution prior to this metric time is important as it affects the state at the start of the output measurement.

The corresponding adapted meshes are shown in Figure 44. Output-based adaptation targets the airfoil leading and trailing edges, the boundary-layer region above the front of the airfoil, and slightly the stagnation line in front and the wake behind the airfoil. The approximation-error indicator also targets the leading and trailing edges and puts

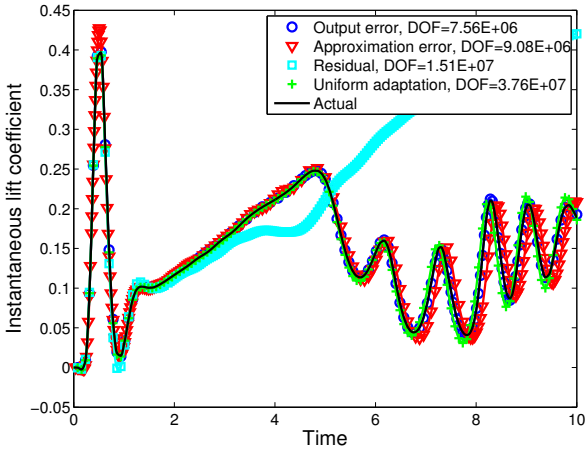


42.1: Output definition

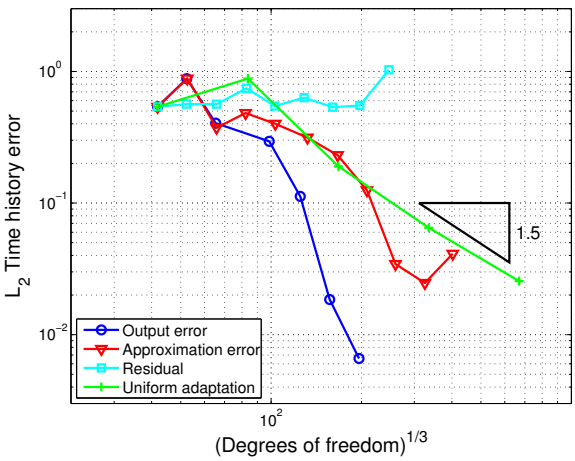


42.2: Output convergence

Figure 42: Impulsively-started airfoil: time integral output definition and its convergence under the adaptive indicators. Error bars at $\pm\delta J$ and whiskers at $\pm\epsilon$ are shown for the output-based results. The "actual" output is computed on a uniformly-refined final output-adapted space-time mesh.



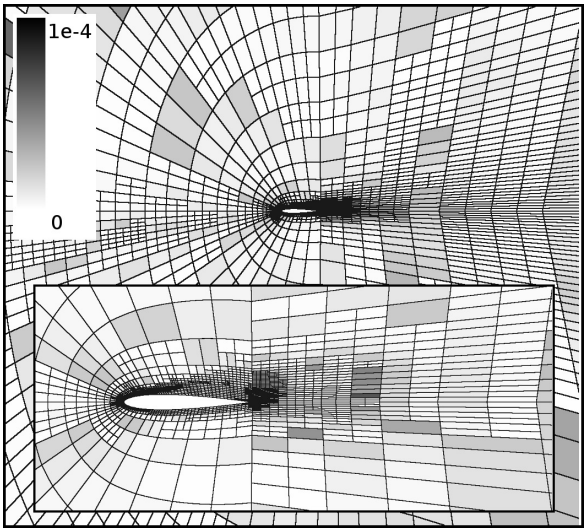
43.1: Sample output histories



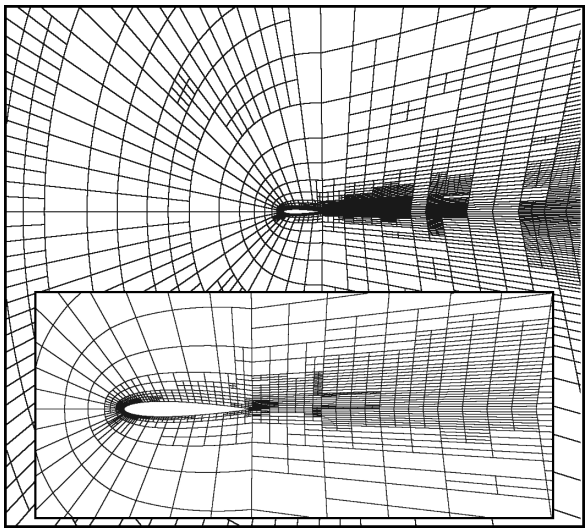
43.2: L_2 error convergence

Figure 43: Impulsively-started airfoil: sample lift coefficient time histories and convergence of the L_2 time history error for various adaptive indicators. The "actual" time history is computed on a uniformly-refined final output-adapted space-time mesh.

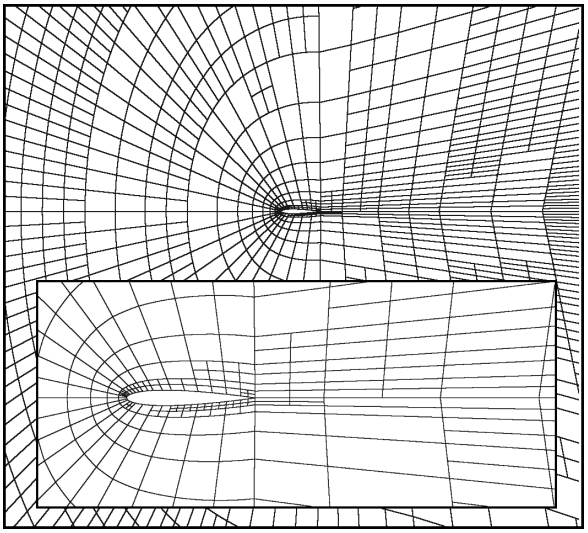
more emphasis on the wake, where the shed vortices propagate. The residual indicator is distracted by effects of the initial condition: the velocity blending near the airfoil sends out acoustic waves that the residual indicator attempts to track as they propagate away from the airfoil.



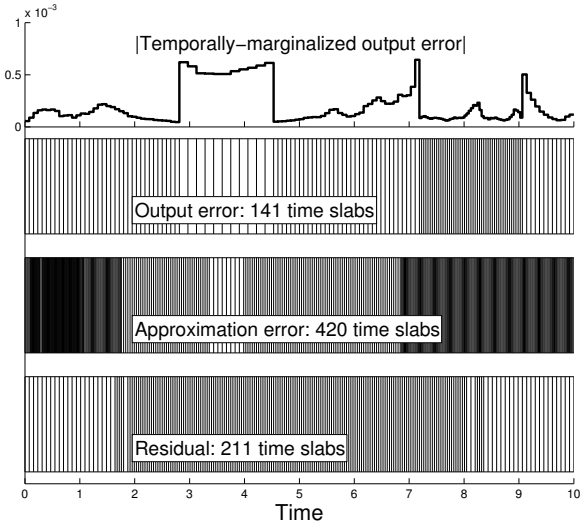
44.1: Adapted on output error (5956 elements)



44.2: Adapted on approx. error (4585 elements)



44.3: Adapted on residual (7929 elements)



44.4: Temporal meshes

Figure 44: Impulsively-started airfoil: adapted spatial and temporal meshes for the seventh adaptive iteration. Localized output error estimates ϵ_e and ϵ^k are shown for the output-error adapted meshes.

The temporal meshes are shown in Figure 44.4. The output-based indicator creates a fairly uniform temporal refinement, with slightly higher resolution prior to the metric time. The approximation error focuses on the initial time, as it tracks the evolution of the blended velocity field, and the latter 1/3 time when the shed vortices develop. The residual indicator again creates a mostly-uniform temporal mesh as it tracks the acoustic waves.

5.4.2 Dynamic-Order Adaptation for Pitching and Plunging Airfoils

In this example we demonstrate the performance of dynamic-order spatial mesh refinement/coarsening together with time slab refinement/coarsening for an unsteady computation on a deformable domain. The adaptive mechanics consist of refinement driven by a prescribed fixed-growth factor for degrees of freedom of $f^{\text{growth}} = 1.35$ per adaptive iteration, together with a coarsening fraction of $f^{\text{coarsen}} = .05$. Adaptation using the output-based error indicator is compared to adaptation using an indicator obtained from an unweighted residual, and to uniform h and p refinement of the spatial mesh. The temporal adaptation is always time slab refinement/coarsening.

This case involves two airfoils pitching and plunging in series at low Reynolds number [64]. The airfoils start from an impulsive free-stream condition and undergo three periods of motion. The plunge amplitude is 0.25 chords, the pitch amplitude is 30°, and the period of both motions is $T = 2.5$. The Strouhal, Mach, and Reynolds numbers are 2/3, 0.3, and 1200, respectively. The airfoils are offset 4.5 chords horizontally and 1 chord vertically, and are situated in a 60 x 60 chord-length mesh with $3,534$ triangular elements.

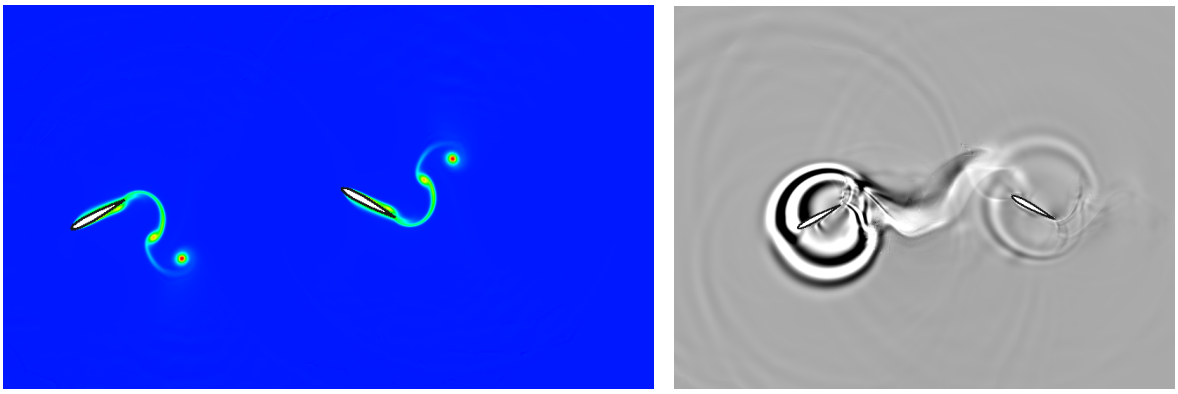
Entropy contours at various phases of the motion are shown in Figure 45. A reverse Kármán vortex street develops behind each airfoil, and the second airfoil interacts with the wake from the first airfoil near the end of the simulation. Our output of interest is the lift on the second airfoil integrated from time $t = 7.25$ to $t = 7.5$ (the final time).

Adaptations were performed starting from an initial $p = 1$, 90 time step solution, with a 35% growth factor and 5% coarsening factor used for the output- and residual-based methods. The spatial order p was constrained to lie between 0 and 5, and a DG1 scheme was used in time.

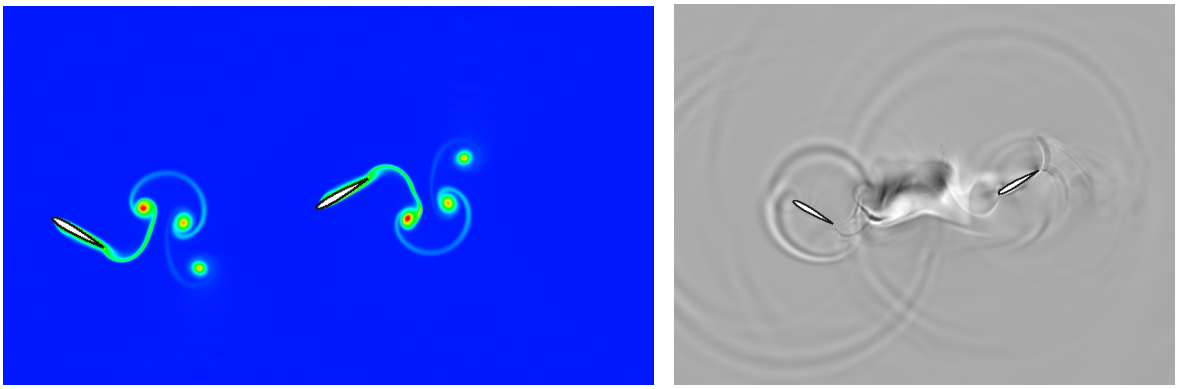
The output convergence for each adaptive method as a function of total space-time degrees of freedom is shown in Figure 46. We see that the output-based adaptation converges much faster than uniform refinement, requiring roughly two orders of magnitude fewer degrees of freedom. These gains relative to uniform refinement are impressive, though not entirely unexpected. Equally interesting is the difference between the output-based and residual adaptation. The residual indicator targets regions of the domain where the governing equations are not well-satisfied, and hence usually performs well for static problems. However, in this case, its performance is erratic and no better than uniform refinement. This erratic behavior is a consequence of the acoustic waves that emanate from the airfoils as they pitch back and forth. The residual indicator becomes distracted by these waves and exhausts degrees of freedom trying to resolve them as they propagate throughout the domain. The output-based method, on the other hand, dismisses the majority of these waves as irrelevant and simply ignores them.

The spatial and temporal meshes from the final output-based adaptation are shown in Figures 47 and 48, respectively. We see that the near-airfoil and vortex shedding regions are targeted for adaptation, as well as the group of large elements surrounding the mesh motion regions. While somewhat difficult to observe in the still-frames, the initial vortex shed from the first airfoil is heavily targeted throughout the simulation, since this vortex later collides with the second airfoil near the final time.

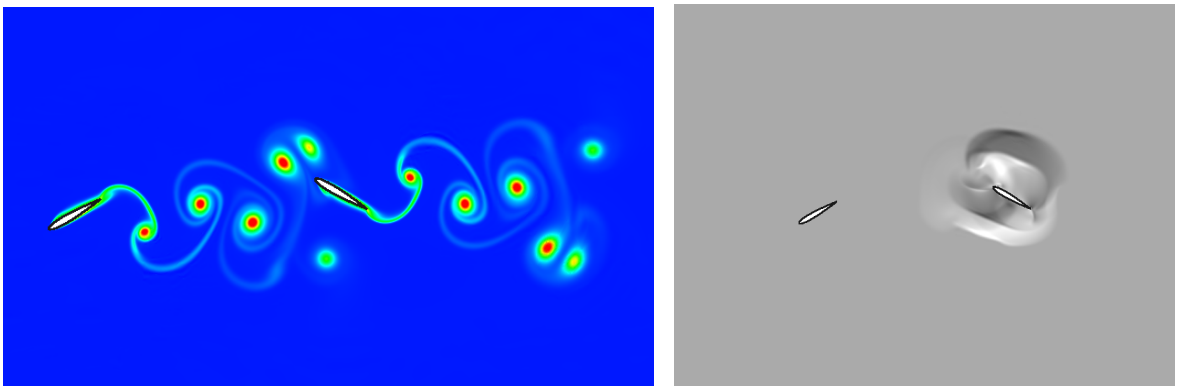
To highlight one of the factors driving the adaptation, contours of the GCL adjoint are shown alongside the entropy contours in Figure 45. The time $t = 0.70T$ is the instant before the initial vortex is shed, and the large sensitivity of the output to this event can be



45.1: $t = 0.70T$



45.2: $t = 1.25T$



45.3: $t = 2.75T$

Figure 45: Two-airfoil case: Entropy (left) and GCL adjoint (right) contours at various stages of the motion on a fine mesh. The GCL adjoint contours have been re-scaled to more clearly show the features (black is -2, white is 1). Both acoustic and convective modes of error propagation can be seen in the first two contours, while at the final time, the adjoint field collapses on the second airfoil.

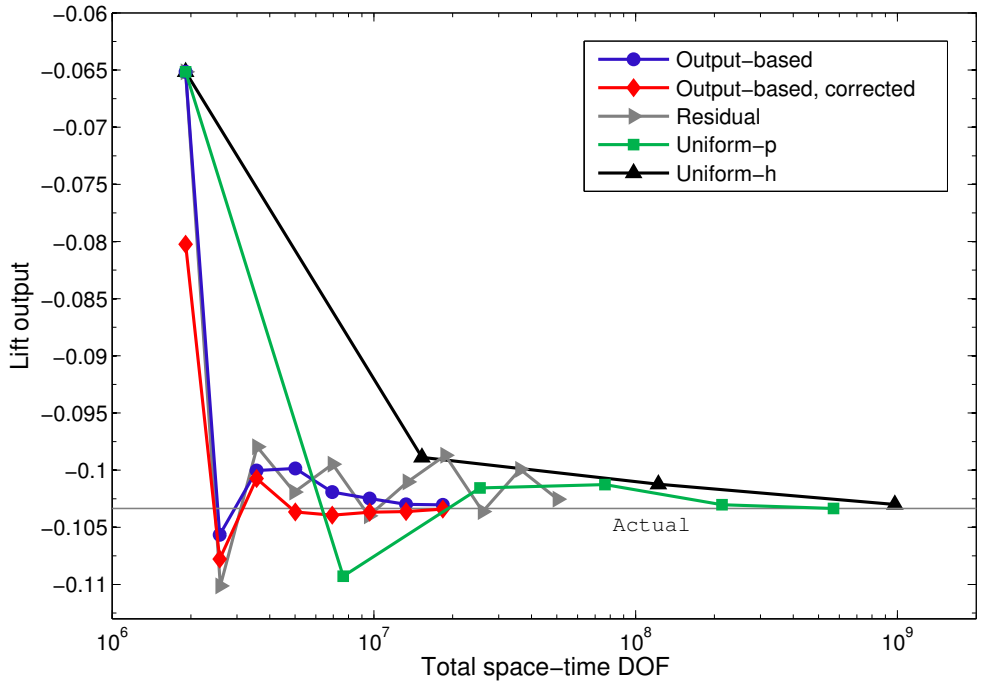
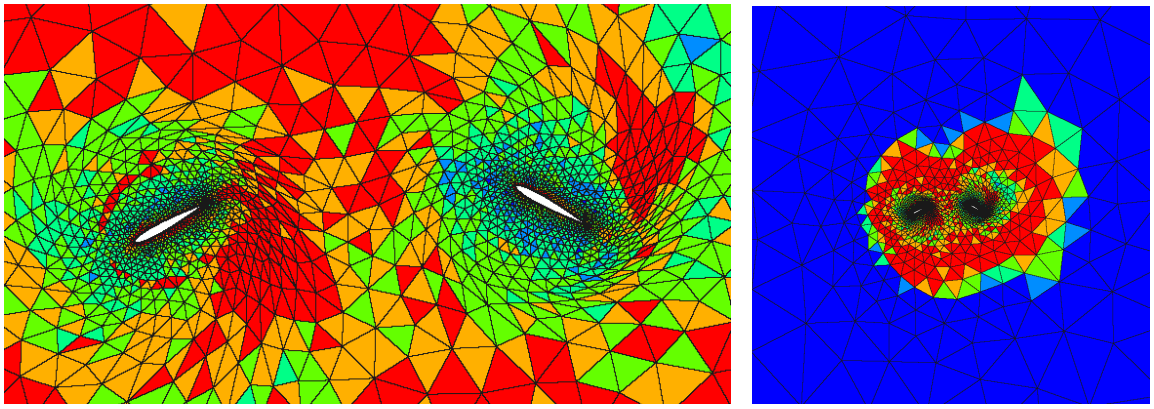


Figure 46: Two-airfoil case: Output convergence for various adaptive methods. The output-based method performs the best. Note that for all plots, the "actual" value is taken from the final uniform p-refinement.

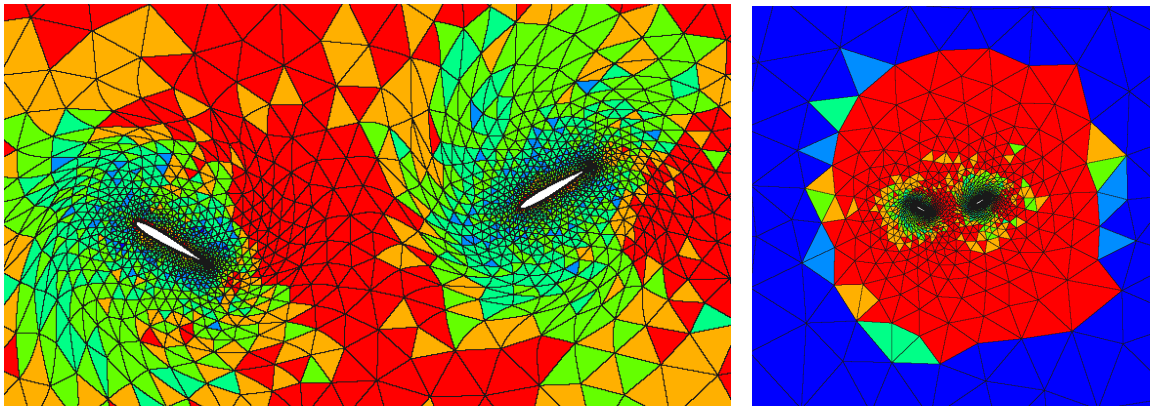
seen in the adjoint contours. As the simulation proceeds, the output sensitivity gradually shifts from the first airfoil to the second, before collapsing upon the second airfoil at the final time.

Some other aspects of the GCL adjoint are worth pointing out. In the first two contours, the near-circular rings represent inward-moving (adjoint) acoustic waves, which converge upon a particular region as the simulation proceeds. The existence of a ring implies that an important event in space-time is about to occur, and any errors made within the circumference of the ring have the ability to influence this event. In this simulation, the important events tend to be instances of vortex shedding, and the rings converge on the trailing edge regions. Lastly, between the two airfoils, a path can be seen tethering them together. This path appears because any errors within it ultimately reach the second airfoil via convection, and can therefore directly affect the output.

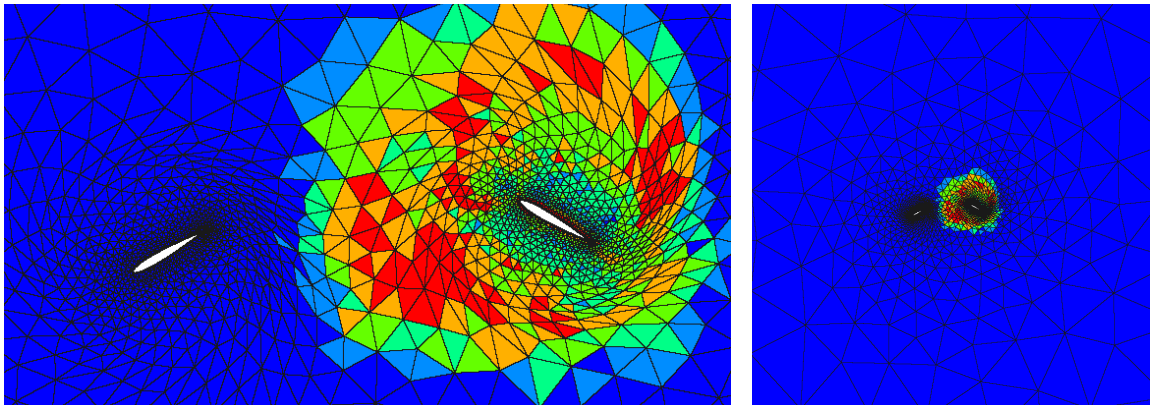
Above, we solved the fine-space adjoint to machine precision to ensure accurate error estimates and efficient allocation of mesh degrees of freedom. In practice, CPU time is another important factor, and solving the adjoint to machine precision is generally not the most efficient strategy. In Figure 49, we show a wall time comparison for the various adaptive strategies, with the adjoint smoothed to a residual tolerance of 1×10^{-3}. This tolerance is not necessarily optimal, and we note that the code itself has not been optimized for CPU time. However, our aim is simply to give an idea of the relative timings. While the performance gains for the output-based method are not as substantial in this context, it converges roughly $5 - 10$ times faster than the uniform refinement strategies. It also significantly outperforms the residual-based adaptation, which fails to converge in any reasonable amount of time.



47.1: $t = 0.70T$



47.2: $t = 1.25T$



47.3: $t = 2.75T$

Figure 47: Two-airfoil case: Output-adapted meshes at various stages of the motion. Blue is $p = 0$, red is $p = 5$.

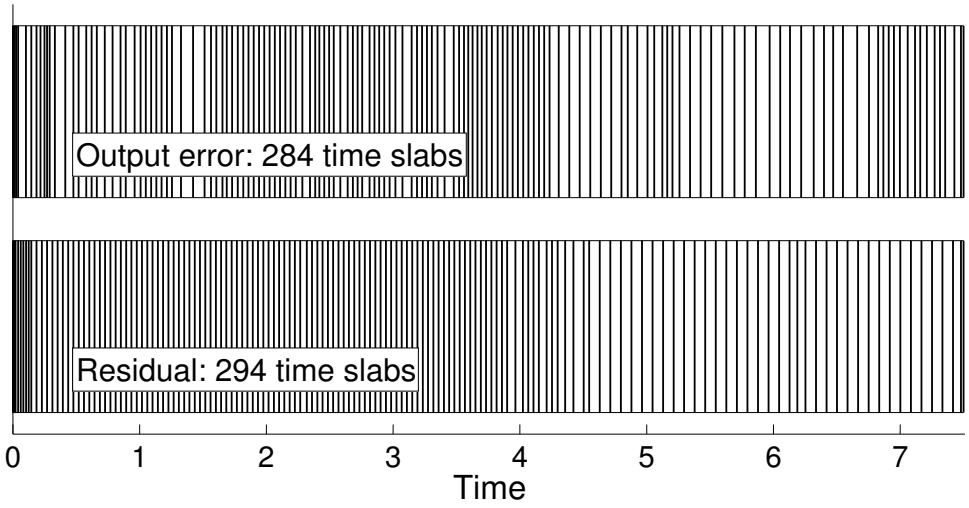


Figure 48: Two-airfoil case: Temporal grids from the seventh adaptation of both output-based and residual runs. For clarity, only every other time slab is plotted.

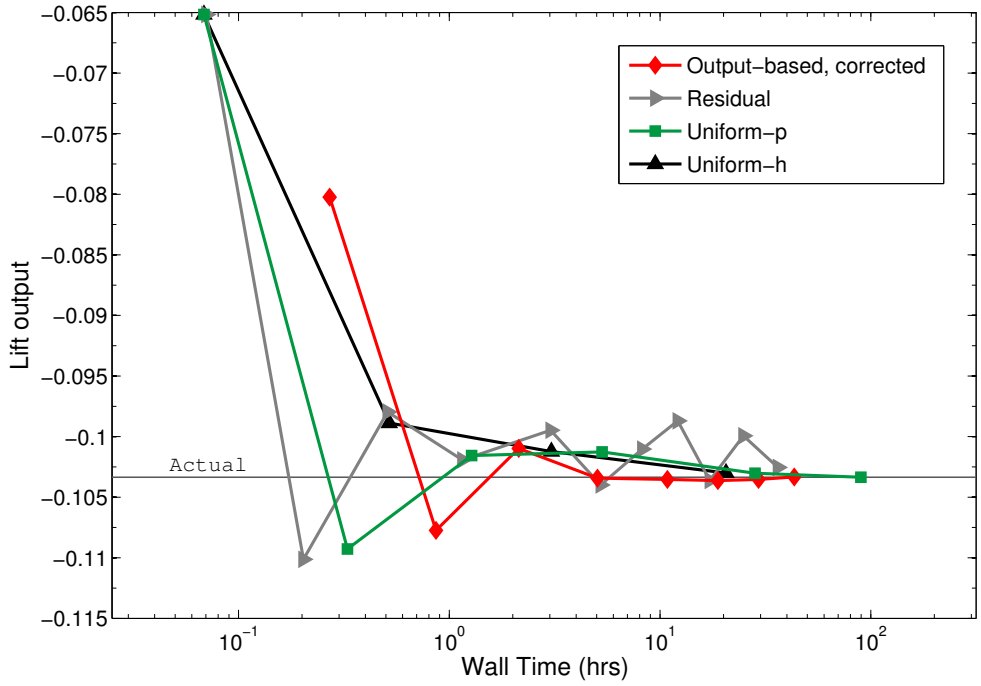


Figure 49: Two-airfoil case: Wall-time comparison for the adaptive methods. The output-based method converges the fastest.

5.4.3 Dynamic-Order Adaptation for a Three-Dimensional Wing

In this example we demonstrate the performance of dynamic-order spatial mesh refinement/coarsening together with time slab refinement/coarsening for an unsteady computation on a deformable domain in three dimensions. The adaptive mechanics consist of refinement driven by a prescribed fixed-growth factor for degrees of freedom of $f^{\text{growth}} = 1.3$ per adaptive iteration, together with a coarsening fraction of $f^{\text{coarsen}} = .05$. Adaptation using the output-based error indicator is compared to adaptation using an indicator obtained from an unweighted residual, as well as to uniform h and p refinement of the spatial mesh – the temporal adaptation is always time slab refinement/coarsening.

In three dimensions, the underlying ideas for mesh motion and adaptation remain the same, but the increased algorithmic complexity and new dimensional scaling do not allow for a simple extrapolation of the 2D results. In particular, the extra dimension makes the scaling for the adjoint problem less favorable. For a $(p=1, r=1)$ primal solve in 2D, the dimension of the fine-space adjoint is approximately 3 times that of the primal, while in 3D this factor increases to over 5. When combined with additional CPU costs, computing (or even smoothing) the adjoint on the fine space becomes an expensive proposition. We therefore employ spatial and temporal reconstruction of the adjoint in lieu of a solution on the fine space, as discussed in Section 5.3.4.

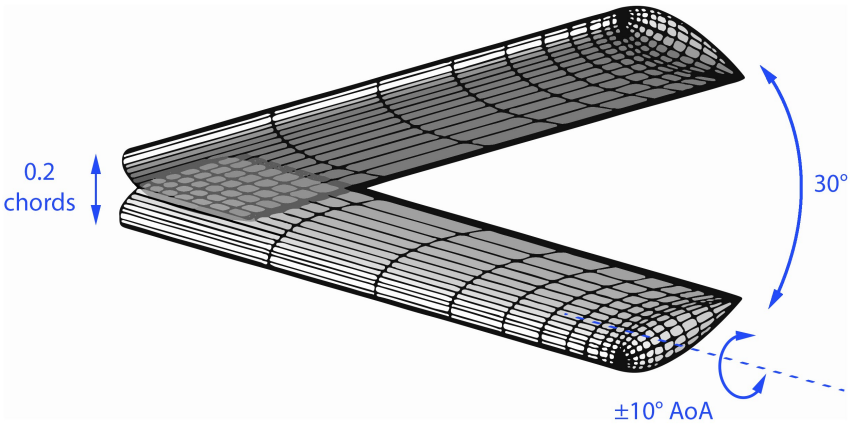


Figure 50: 3D wing: Schematic of the flapping motion. The flow regime is approximately that of a housefly.

To test this strategy, we proceed in a similar manner as in 2D, though now simulating a full wing rather than just an airfoil. The wing is shown in Figure 50, along with a schematic of the flapping motion. The wing moves in all dimensions, with a 30° stroke angle, a slight vertical plunge simulating movement of the "shoulder joint," and an angle of attack variation of ±10°. The Mach number is 0.3, while the Reynolds number of 500, Strouhal number of 0.4, and aspect ratio of 1.5 place the wing in the flight regime of a small housefly.

Three periods of motion were simulated, and the solution at various times is shown in Figure 51. Strong vortex cores develop near the leading edge and wingtip regions before detaching and shedding into the wake. For adaptation purposes, the output of interest is taken to be the lift integrated over the final 5% of the simulation time (from $t = 7.1$

to $t = 7.5$). To solve the problem, output-based, residual, and uniform p-refinement strategies were employed. Adaptations were performed starting from an initial $p = 0$, 75 time step solution, the spatial order p was constrained to lie between 0 and 5, and a DG1 (i.e. $r = 1$) scheme was used in time. The convergence for each adaptive method is

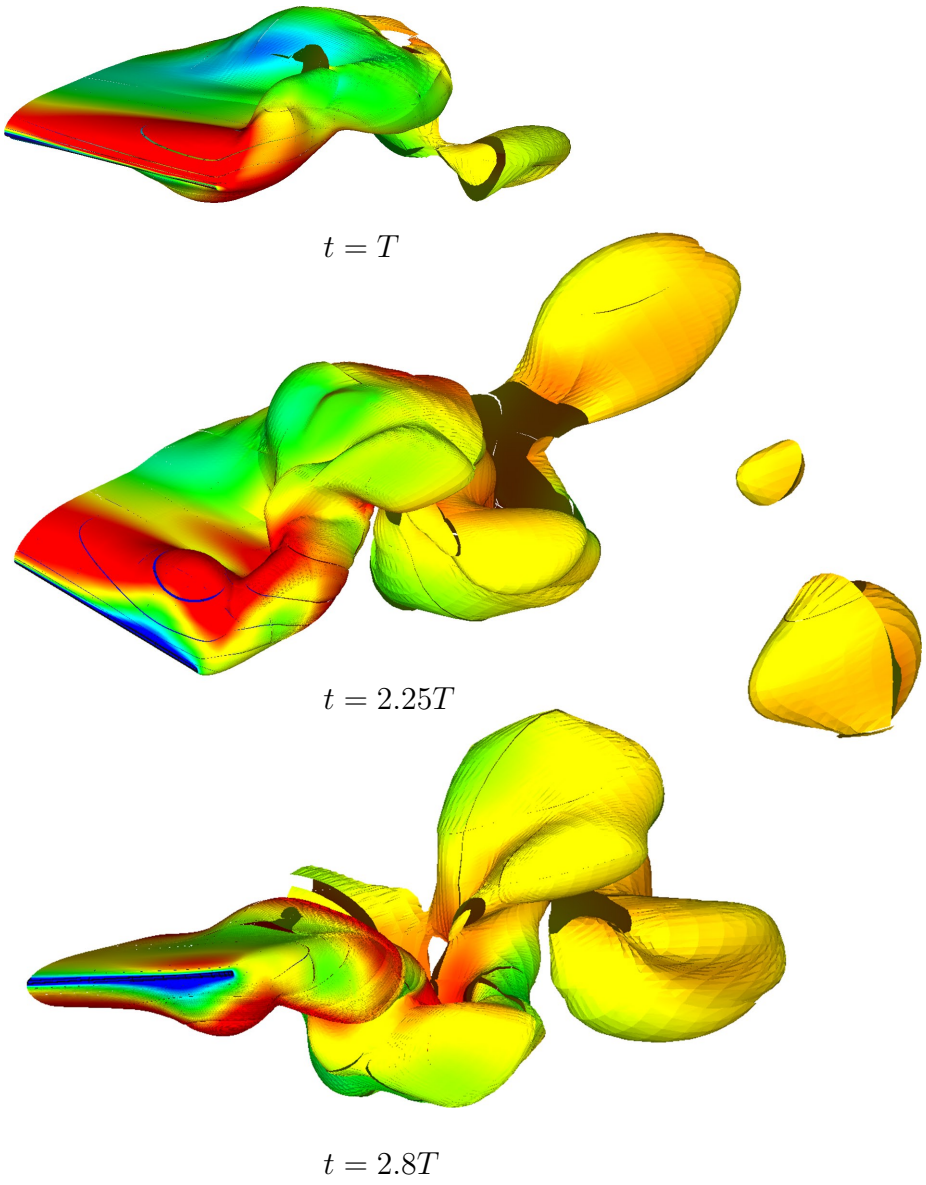
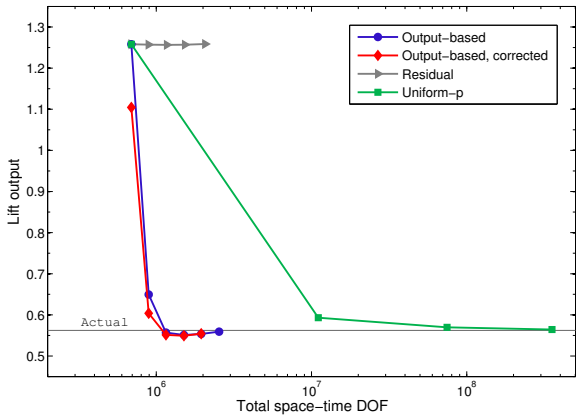
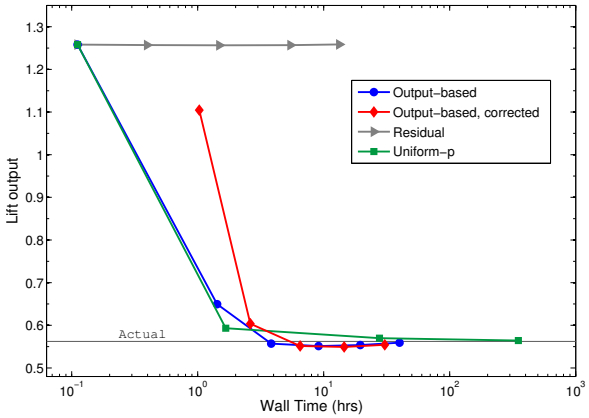


Figure 51: 3D wing: Mach contours projected onto entropy isosurfaces, shown for several stages of the flapping motion. The maximum Mach number (in red) is approximately 0.5. The images are taken from the final ($p=3$) uniform refinement.

shown in Figure 52, and the results are encouraging. We see that although reconstructed adjoints were used, the output-based adaptation still performs well, and converges with about two orders of magnitude fewer degrees of freedom than uniform refinement. As expected, the accuracy of the error estimate itself is not great (judging by the fact that the corrected output converges no faster than the uncorrected one), but is enough to guide the adaptation in the correct direction. Convergence as a function of wall time is shown in Figure 52, and we see that the method also performs well in this context.



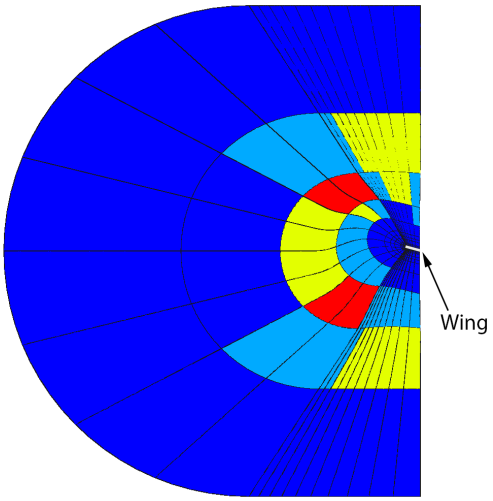
52.1: Convergence versus DOF



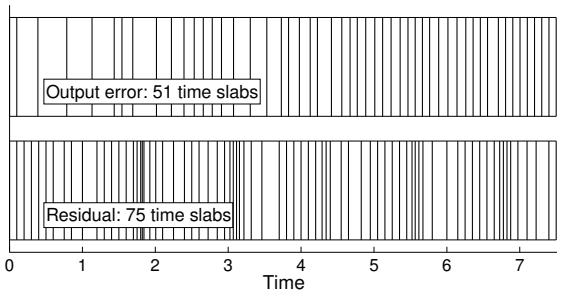
52.2: Convergence versus wall time

Figure 52: 3D wing: Output convergence as a function of degrees of freedom and wall time.

Of additional interest is the performance of the residual-based adaptation, which is so poor that the curve is easy to miss in the plots. The reason for this behavior is evident in Figure 53, which shows the orders midway through the final residual adaptation. The residual indicator flags only large elements in the mesh blending region for refinement, and leaves all elements near the wing at their original low order. The output-based adaptation, on the other hand, refines elements near the wing as needed. The adapted spatial meshes from the output-based runs are shown in Figure 54, while adapted temporal grids are shown in Figure 53. We see that, at early times, the output-based indicator leaves the mesh relatively coarse, but progressively increases the resolution in both space and time as the simulation progresses.



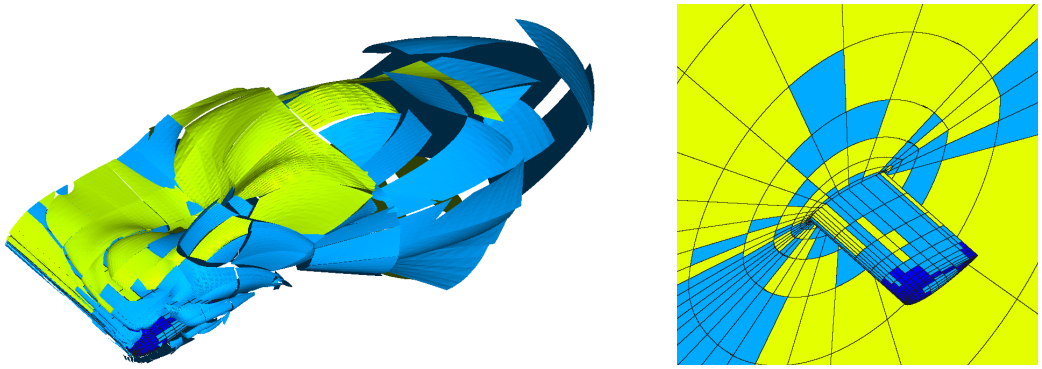
53.1: Residual-adapted orders



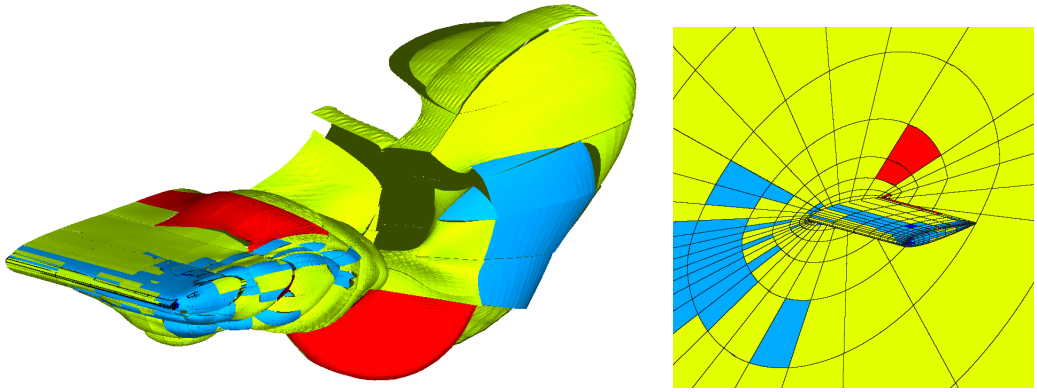
53.2: Temporal grids

Figure 53: : On the left, spatial orders from the residual adaptation midway through the simulation (blue is $p = 0$, red is $p = 3$). The adaptation targets only elements far from the wing, leading to poor output convergence. On the right, temporal grids from the final output-based and residual adaptations. The residual adaptation refines periodically with the motion, while the output-based adaptation increases the resolution near the final times.

54.1: $t = T$



54.2: $t = 2.25T$



54.3: $t = 2.8T$

Figure 54: 3D wing: Spatial orders from the final output-based adaptation, shown at several stages of the flapping motion. Dark blue is $p = 0$, red is $p = 3$. The images on the left show the interpolation orders projected onto entropy isosurfaces.

6 Hybrid Discontinuous Galerkin Discretizations

6.1 Introduction

A DG discretization uses extra degrees of freedom (DOFs) compared to continuous finite element methods for a given mesh and approximation order. While the discontinuous approximation space in DG provides stability for convection-dominated problems, it does not generally improve approximation capabilities of the underlying continuous functions. The cost of "duplicating" these degrees of freedom at element interfaces is especially pronounced in 3D. Hybrid DG methods, have been recently introduced to address this duplication problem [78, 25, 24, 77]. As illustrated in Figure 55, hybrid methods derive from mixed formulations in which additional unknowns are placed at element interfaces.

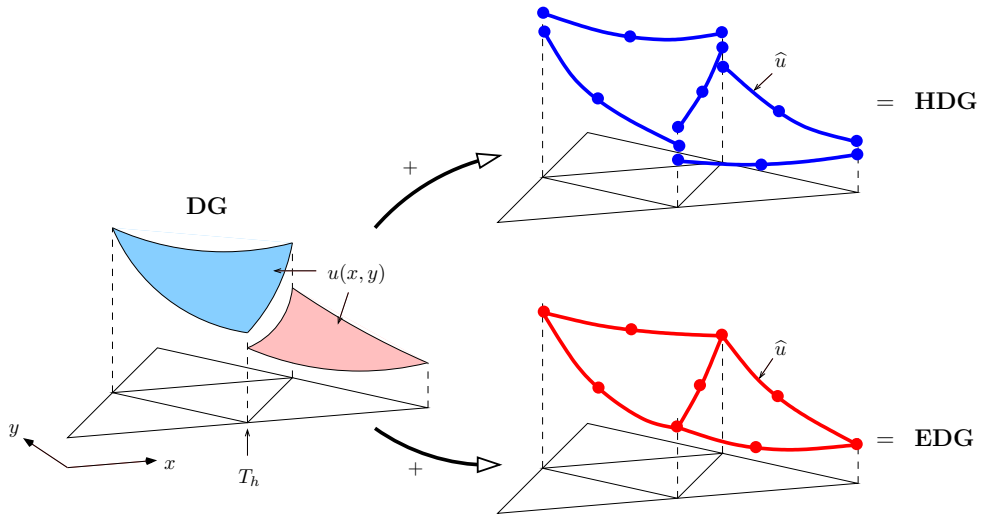


Figure 55: Illustration of the approximation space augmentation of hybrid discretizations compared to a DG discretization. In HDG (hybrid DG), a face-wise discontinuous interface state $\hat{u}$ augments the standard DG element-interior approximation. In EDG (embedded DG), the interface state $\hat{u}$ is chosen to lie in a space of functions that are continuous at mesh vertices.

In DG, the number of globally-coupled unknowns scales with order p as p^d, where d is the spatial dimension. Hybrid DG methods have the advantage that the exponent is reduced by one to p^{d-1}, because the $\hat{u}$ variables turn out to be the only globally-coupled unknowns, and these lie in a space of one lower dimension (faces instead of volumes). To quantify this advantage, we compare DG, continuous Galerkin (CG), and the two hybrid methods HDG and EDG using degrees of freedom per vertex of a mesh. This comparison, presented in Table 3 assumes typical isotropic meshes (e.g. 6 triangles or 24 tetrahedra adjacent to a vertex) and ignores boundary effects.

6.2 Weak Form

We consider a hybrid discontinuous Galerkin (HDG) discretization of the PDE in (1). Other hybrid methods, such as EDG, can be obtained by modifying the approximation space definitions. Assuming that diffusion is present, we introduce a new element-interior

Table 3: .The numbers in the below tables indicate approximately how many degrees of freedom (per equation of a system) we need per vertex of the mesh.

Triangles:

method	$p=1$	$p=2$	$p=3$	$p=4$
DG	6	12	20	30
CG	1	4	9	16
HDG	6	9	12	15
EDG	1	4	7	10

Quadrilaterals:

method	$p=1$	$p=2$	$p=3$	$p=4$
DG	4	9	16	25
CG	1	4	9	16
HDG	4	6	8	10
EDG	1	3	5	7

Tetrahedra:

method	$p=1$	$p=2$	$p=3$	$p=4$
DG	24	60	120	210
CG	1	8.2	27.4	64.6
HDG	36	72	120	180
EDG	1	8.2	27.4	58.6

Hexahedra:

method	$p=1$	$p=2$	$p=3$	$p=4$
DG	8	27	64	125
CG	1	8	27	64
HDG	12	27	48	75
EDG	1	7	19	37

variable, $\vec{\mathbf{q}} \in [\mathbb{R}^s]^d$, for approximating the gradient. The general system of equations becomes

$$\begin{aligned} \vec{\mathbf{q}} - \nabla \mathbf{u} &= \mathbf{0}, \\ \partial_t \mathbf{u} + \nabla \cdot \underbrace{\left[\vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \vec{\mathbf{q}})\right]}_{\vec{\mathbf{H}}(\mathbf{u}, \vec{\mathbf{q}})} + \mathbf{s}(\mathbf{u}, \vec{\mathbf{q}}) &= \mathbf{0}, \end{aligned} \tag{86}$$

where $\vec{\mathbf{F}} \in [\mathbb{R}^s]^d$ and $\vec{\mathbf{G}} \in [\mathbb{R}^s]^d$ are the inviscid and viscous fluxes respectively, $\vec{\mathbf{H}} \in [\mathbb{R}^s]^d$ is the *total* flux, and $\mathbf{s} \in \mathbb{R}^s$ is a source term. The HDG weak form is obtained by dotting the first equation with test functions $\vec{\mathbf{v}} \in [\boldsymbol{\mathcal{V}}_h]^d$ and the second with test functions $\mathbf{w} \in \boldsymbol{\mathcal{V}}_h$ and integrating over all elements of the domain. The result is

$$(\vec{\mathbf{q}}, \vec{\mathbf{v}})_{T_h} + (\mathbf{u}, \nabla \cdot \vec{\mathbf{v}})_{T_h} - \langle \widehat{\mathbf{u}}, \vec{\mathbf{v}} \cdot \vec{n} \rangle_{\partial T_h} = \mathbf{0}, \quad \forall \vec{\mathbf{v}} \in [\boldsymbol{\mathcal{V}}_h]^d \tag{87}$$

$$(\partial_t \mathbf{u}, \mathbf{w})_{T_h} - \left(\vec{\mathbf{H}}, \nabla \mathbf{w}\right)_{T_h} + \left\langle \widehat{\vec{\mathbf{H}}} \cdot \vec{n}, \mathbf{w} \right\rangle_{\partial T_h} + (\mathbf{s}, \mathbf{w})_{T_h} = \mathbf{0}, \quad \forall \mathbf{w} \in \boldsymbol{\mathcal{V}}_h \tag{88}$$

$$\left\langle \widehat{\vec{\mathbf{H}}} \cdot \vec{n}, \boldsymbol{\mu} \right\rangle_{\partial T_h \setminus \partial \Omega} = \mathbf{0}, \quad \forall \boldsymbol{\mu} \in \boldsymbol{\mathcal{M}}_h \tag{89}$$

where $\widehat{\cdot}$ denotes a quantity (not necessarily unique) defined on an interface between two elements. For example, $\widehat{\vec{\mathbf{H}}}$ is the total flux on an interface, but it is defined separately by each of the adjacent elements. The inner products appearing in the above equation are defined as

$$\text{Inner product over elements :} \qquad (\mathbf{u}, \mathbf{w})_{T_h} = \sum_{e=1}^{N_e} \int_{\Omega_e} \mathbf{u}^T \mathbf{w} \, d\Omega$$

$$\text{Inner product over all faces :} \qquad \langle \mathbf{u}, \vec{\mathbf{v}} \cdot \vec{n} \rangle_{\partial T_h} = \sum_{e=1}^{N_e} \int_{\partial \Omega_e} \mathbf{u}^+ \vec{\mathbf{v}}^+ \cdot \vec{n}^+ \, ds$$

Recall that N_e is the number of elements and that $(\cdot)^+$ refers to a quantity taken from the interior of an element. Also $\vec{n}^+$ is a normal pointing outward from Ω_e. In HDG, the state fluxes $\widehat{\mathbf{u}} \in \mathbb{R}^s$ are separate unknowns *on interior faces* σ_f. We do not place unknowns on boundary faces. To close the system, we need equations associated with the interior faces: these are flux continuity statements given in (89, where $\boldsymbol{\mu} \in \mathcal{M}_h$ are test functions defined on the interior faces. Figure 56 summarizes the definitions of the various quantities.

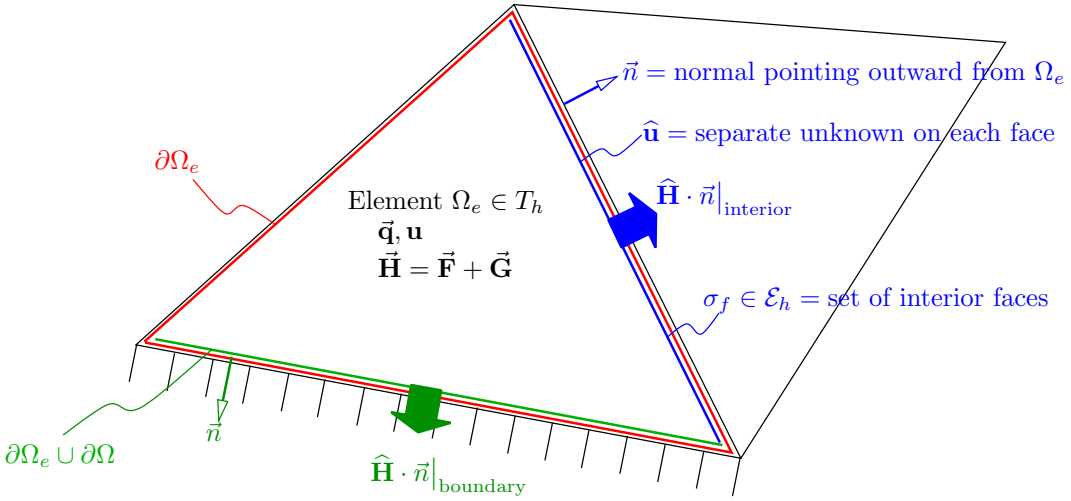


Figure 56: Definitions of various HDG quantities.

The approximation spaces used for the solution and test functions are given in Table 4. The relevant spaces are defined as,

$$\begin{aligned} \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s, \qquad \mathcal{V}_h &= \left\{ u \in L^2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^{p_e}(\Omega_e) \; \forall \, \Omega_e \in T_h \right\} \\ \boldsymbol{\mathcal{M}}_h = [\mathcal{M}_h]^s, \qquad \mathcal{M}_h &= \left\{ \widehat{u} \in L^2(\mathcal{E}_h) : \widehat{u}|_{\sigma_f} \in \mathcal{P}^{p_f}(\sigma_f) \; \forall \, \sigma_f \in \mathcal{E}_h \right\} \end{aligned}$$

We assume that each element Ω_e can potentially have its own order, p_e, and that both $\mathbf{u}$ and $\vec{\mathbf{q}}$ are approximated with this same order. Also, we assume that each face, σ_f can have its own order, p_f. Note that by approximating $\vec{\mathbf{q}}$ independently, HDG can generally predict gradients to a higher-order accuracy compared to standard DG.

Table 4: Spaces for trial and test functions in an HDG discretization.

Trial functions	Test functions	Space
$\mathbf{u}$	$\mathbf{w}$	$\boldsymbol{\mathcal{V}}_h$
$\vec{\mathbf{q}}$	$\vec{\mathbf{v}}$	$[\boldsymbol{\mathcal{V}}_h]^d$
$\widehat{\mathbf{u}}$	$\boldsymbol{\mu}$	$\boldsymbol{\mathcal{M}}_h$

6.3 Fluxes and Stabilization

Consider element Ω_e. The general form for the total flux dotted with the outward-pointing normal on an interior face of element Ω_e is

$$\widehat{\mathbf{H}} \cdot \vec{n}\big|_{\text{interior}} = \vec{\mathbf{H}}(\widehat{\mathbf{u}}, \vec{\mathbf{q}}) \cdot \vec{n} + \mathbf{S}(\widehat{\mathbf{u}})(\mathbf{u} - \widehat{\mathbf{u}}), \tag{90}$$

where $\mathbf{u}$ and $\vec{\mathbf{q}}$ are quantities taken from the interior of element Ω_e. Since we have two elements adjacent to each interior face, and since the total flux definition above uses element-interior quantities, the value of $\widehat{\mathbf{H}} \cdot \vec{n}$ may not be pointwise-continuous across the face. Note that the total flux consists of two pieces. First, we have the physical flux based on the interface state, $\widehat{\mathbf{u}}$, and the element-interior gradient, $\vec{\mathbf{q}}$. Second, we have a stabilization term that involves $\mathbf{u}$ and $\widehat{\mathbf{u}}$. The normal vector does not appear in the stabilization contribution. $\mathbf{S} \in \mathbb{R}^{s \times s}$ is a stabilization tensor that could be chosen in various ways. Two choices are given below.

6.3.1 Lax-Friedrichs Stabilization

This is a simple choice in which we take $\mathbf{S} = \tau \mathbf{I}$, where τ is a global constant and $\mathbf{I} \in \mathbb{R}^{s \times s}$ is an identity matrix. The constant τ could be set by the user, hard-coded, or computed from the following equation:

$$\tau = |c_{\max}| + \frac{\nu}{\ell_{\text{visc}}}, \tag{91}$$

where $c_{\max}$ is (an estimate of) the maximum characteristic speed, ν is the diffusivity, and ℓ_{visc} is a user-defined viscous length-scale (not dependent on mesh size). A small modification of this stabilization would be to define τ locally on each interior face, possibly using a local maximum characteristic speed.

6.3.2 Roe Stabilization

A more sophisticated stabilization takes a "Roe"-like approach to the treatment of the convective term. The stabilization matrix is

$$\mathbf{S} = \mathbf{R}\,|\mathbf{\Lambda}|\,\mathbf{L} + \tau_{\text{visc}}\mathbf{I}, \tag{92}$$

where $\tau_{\text{visc}} = \nu/\ell_{\text{visc}}$ is the same as the viscous portion of the Lax-Friedrichs stabilization. The matrices $\mathbf{R}, \mathbf{\Lambda}, \mathbf{L}$, all in $\mathbb{R}^{s \times s}$, come from an eigen-decomposition of the convective flux Jacobian matrix computed about the interface state $\widehat{\mathbf{u}}$,

$$\frac{\partial(\vec{\mathbf{F}}(\widehat{\mathbf{u}}) \cdot \vec{n})}{\partial \widehat{\mathbf{u}}} = \mathbf{R}\,\mathbf{\Lambda}\,\mathbf{L}. \tag{93}$$

6.3.3 Boundary Conditions

Boundary conditions 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,

$$\begin{aligned} \widehat{\mathbf{H}} \cdot \vec{n}\big|_{\text{boundary}} &= \widehat{\mathbf{H}}^b(\mathbf{u}, \vec{\mathbf{q}}, \vec{n}, \text{BCs}) + \mathbf{S}(\mathbf{u}^b)(\mathbf{u} - \mathbf{u}^b) \\ \mathbf{u}^b &= \mathbf{u}^b(\mathbf{u}, \text{BCs}) = \text{boundary state (a projection)} \end{aligned} \tag{94}$$

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.

When implementing the boundary condition, we first compute a boundary state $\mathbf{u}^b$ using the element-interior state $\mathbf{u}$ and the boundary condition data (BCs). The map from $\mathbf{u}$ to $\mathbf{u}^b$ has to be a projection, so that applying it more than once has no additional effect compared to applying it just once. An example of how boundary conditions come into this map is in the definition of a boundary state in the case of a no-slip wall: the map to the boundary state must zero out the velocity components. Second, we compute the boundary flux using this boundary state, the element-interior gradient $\vec{\mathbf{q}}$, the normal $\vec{n}$ and possibly the BC data again (e.g. if a heat-flux BC directly specified the energy flux through a wall).

Finally, note that stabilization is also included in the boundary flux. This stabilization penalizes jumps between the element-interior state, $\mathbf{u}$, and the boundary state, $\mathbf{u}^b$.

6.4 Discrete System, Static Condensation, and Matrix Storage

We approximate $\vec{\mathbf{q}}$, $\mathbf{u}$, and $\widehat{\mathbf{u}}$ as follows:

$$\text{inside element } \Omega_e\text{: } \vec{\mathbf{q}}(\vec{x})\big|_{\Omega_e} = \sum_{i=1}^{d}\sum_{n=1}^{N_{pe}} \mathbf{Q}_{ein}\phi_n(\vec{x})\widehat{x}_i \tag{95}$$

$$\text{inside element } \Omega_e\text{: } \mathbf{u}(\vec{x})\big|_{\Omega_e} = \sum_{n=1}^{N_{pe}} \mathbf{U}_{en}\phi_n(\vec{x}) \tag{96}$$

$$\text{on face } \sigma_f\text{: } \hat{\mathbf{u}}(\vec{s})\big|_{\sigma_f} = \sum_{n=1}^{N_{p_f}} \mathbf{\Lambda}_{fn}\mu_n(\vec{s}) \tag{97}$$

where $\vec{x} \in \mathbb{R}^d$ denotes a coordinate inside element Ω_e, $\widehat{x}_i$ is the unit normal in the direction of the x_i coordinate, $\vec{s} \in \mathbb{R}^{d-1}$ denotes a coordinate on face σ_f, and n indexes the degrees of freedom associated with a polynomial approximation of the required order. Note that we assume that $\vec{\mathbf{q}}$ and $\mathbf{u}$ are approximated with the same basis and order.

Using a Galerkin method, we convert the weak variational statements in (87-89) to discrete form by defining the following residuals:

$$\mathbf{R}^Q_{ein} = \int_{\Omega_e} \mathbf{q}_i\phi_n\,d\Omega + \int_{\Omega_e} \mathbf{u}\partial_i\phi_n\,d\Omega - \int_{\partial\Omega_e\setminus\partial\Omega} \widehat{\mathbf{u}}\phi_n n_i\,ds - \int_{\partial\Omega_e\cap\partial\Omega} \mathbf{u}^b\phi_n n_i\,ds \tag{98}$$

$$\mathbf{R}^U_{en} = \int_{\Omega_e} \partial_t\mathbf{u}\phi_n\,d\Omega - \int_{\Omega_e} \vec{\mathbf{H}}\cdot\nabla\phi_n\,d\Omega + \int_{\partial\Omega_e\setminus\partial\Omega} \widehat{\mathbf{H}}\cdot\vec{n}\big|_{\text{interior}}\phi_n\,ds \tag{99}$$

$$+ \int_{\partial\Omega_e\cap\partial\Omega} \widehat{\mathbf{H}}\cdot\vec{n}\big|_{\text{boundary}}\phi_n\,ds + \int_{\Omega_e} \mathbf{s}\,\phi_n\,d\Omega \tag{100}$$

$$\mathbf{R}^\Lambda_{fn} = \int_f \left\{\widehat{\mathbf{H}}\cdot\vec{n}\big|_L + \widehat{\mathbf{H}}\cdot\vec{n}\big|_R\right\}\mu_n\,ds \tag{101}$$

In the definition of the face residuals, $\mathbf{R}^\Lambda$, the notation L and R refers to the elements on the left and right of the face f. Note that $\vec{n}_L = -\vec{n}_R$. Using (90), the integrand appearing in $\mathbf{R}^\Lambda$ is

$$\widehat{\mathbf{H}}\cdot\vec{n}\big|_L + \widehat{\mathbf{H}}\cdot\vec{n}\big|_R = \left[\widehat{\mathbf{H}}(\widehat{\mathbf{u}},\vec{\mathbf{q}}_L) - \widehat{\mathbf{H}}(\widehat{\mathbf{u}},\vec{\mathbf{q}}_R)\right]\cdot\vec{n}_L + \mathbf{S}_L(\mathbf{u}_L - \widehat{\mathbf{u}}) + \mathbf{S}_R(\mathbf{u}_R - \widehat{\mathbf{u}}).$$

For inviscid problems, when we do not have $\vec{\mathbf{q}}$, the term inside the square brackets is zero so the flux balance becomes just an expression of a weak equality between stabilization terms. An interesting situation occurs when $p_L = p_R = p_f$ and when the stabilization tensor is a constant: setting the face residuals to zero enforces that the polynomial $(\mathbf{u}_L + \mathbf{u}_R - 2\widehat{\mathbf{u}})$ is identically zero. So the statement of weak flux continuity actually results in pointwise equality of the fluxes between the elements adjacent to face f.

Note that in the definition of the residuals, the element-interior degrees of freedom talk only to other interior degrees of freedom inside the same element and/or to degrees of freedom on faces adjacent to the element. The residual Jacobian matrix is then

	$\mathbf{Q}_1$	$\mathbf{U}_1$	$\mathbf{Q}_2$	$\mathbf{U}_2$	$\dots$	$\dots$	$\dots$	$\mathbf{\Lambda}_f$	$\dots$
$\mathbf{R}_1^Q$	$\frac{\partial \mathbf{R}_1^Q}{\partial \mathbf{Q}_1}$	$\frac{\partial \mathbf{R}_1^Q}{\partial \mathbf{U}_1}$					$\dots$	$\frac{\partial \mathbf{R}_1^Q}{\partial \mathbf{\Lambda}_f}$	$\dots$
$\mathbf{R}_1^U$	$\frac{\partial \mathbf{R}_1^U}{\partial \mathbf{Q}_1}$	$\frac{\partial \mathbf{R}_1^U}{\partial \mathbf{U}_1}$					$\dots$	$\frac{\partial \mathbf{R}_1^U}{\partial \mathbf{\Lambda}_f}$	$\dots$
$\mathbf{R}_2^Q$			$\frac{\partial \mathbf{R}_2^Q}{\partial \mathbf{Q}_2}$	$\frac{\partial \mathbf{R}_2^Q}{\partial \mathbf{U}_2}$			$\dots$	$\frac{\partial \mathbf{R}_2^Q}{\partial \mathbf{\Lambda}_f}$	$\dots$
$\mathbf{R}_2^U$			$\frac{\partial \mathbf{R}_2^U}{\partial \mathbf{Q}_2}$	$\frac{\partial \mathbf{R}_2^U}{\partial \mathbf{U}_2}$			$\dots$	$\frac{\partial \mathbf{R}_2^U}{\partial \mathbf{\Lambda}_f}$	$\dots$
$\vdots$					$\ddots$		$\dots$	$\vdots$	$\dots$
$\vdots$						$\ddots$	$\dots$	$\vdots$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdot$	$\vdots$	$\cdot$
$\mathbf{R}_f^\Lambda$	$\frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{Q}_1}$	$\frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{U}_1}$	$\frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{Q}_2}$	$\frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{U}_2}$	$\dots$	$\dots$	$\dots$	$\frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{\Lambda}_f}$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdot$	$\vdots$	$\cdot$

Because degrees of freedom associated with element Ω_e do not talk directly to degrees of freedom on other elements, we can *statically condense* all of the element-interior degrees of freedom out of a linear system involving the Jacobian matrix, to obtain a linear system for only the face degrees of freedom. Writing the above Jacobian matrix in shorthand as

$$\text{Jacobian matrix} = \left[\begin{array}{cc|c} \frac{\partial \mathbf{R}^Q}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^Q}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^Q}{\partial \mathbf{\Lambda}} \\ \frac{\partial \mathbf{R}^U}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^U}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^U}{\partial \mathbf{\Lambda}} \\ \hline \frac{\partial \mathbf{R}^\Lambda}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^\Lambda}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^\Lambda}{\partial \mathbf{\Lambda}} \end{array}\right] = \left[\begin{array}{cc} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{array}\right], \tag{102}$$

we define a matrix $\mathbf{K}$ that is the same size as $\mathbf{D} = \frac{\partial \mathbf{R}^\Lambda}{\partial \mathbf{\Lambda}}$ via

$$\mathbf{K} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}. \tag{103}$$

The matrix $\mathbf{K}$ contains information on coupling between face degrees of freedom after taking into account the "transmission" of residuals across elements through element-interior degrees of freedom. The sparsity pattern of $\mathbf{K}$ is such that off-diagonal blocks (groups of degrees of freedom associated with faces) of $\mathbf{K}$ are nonzero only if the two faces are adjacent to the same element. Note that the inversion of $\mathbf{A}$ in (103) is an element-local operation due to the fact that element-interior degrees of freedom are not coupled to each other.

We build $\mathbf{K}$ by looping over all elements Ω_e. On each element, we compute the element-interior and element-boundary terms, and their linearizations, that appear in the above residual definitions. We then consider all pairs of faces adjacent to Ω_e: let's label them f and g, as illustrated in Figure 57. Denote by $\mathbf{K}_{fg}$ the block of the global stiffness matrix

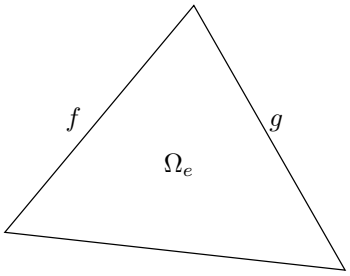


Figure 57: f and g index two faces that are adjacent to the same element, Ω_e.

associated with face f (the row) and face g (the column). Element Ω_e adds the following contribution to this block:

$$\text{contribution of elem } \Omega_e\text{:}\quad \mathbf{K}_{fg} \mathrel{+}= \frac{\partial \mathbf{R}_f^{\Lambda}}{\partial \mathbf{\Lambda}_g}\delta_{fg} - \begin{bmatrix} \dfrac{\partial \mathbf{R}_f^{\Lambda}}{\partial \mathbf{Q}_e} & \dfrac{\partial \mathbf{R}_f^{\Lambda}}{\partial \mathbf{U}_e} \end{bmatrix} \underbrace{\begin{bmatrix} \dfrac{\partial \mathbf{R}_e^{Q}}{\partial \mathbf{Q}_e} & \dfrac{\partial \mathbf{R}_e^{Q}}{\partial \mathbf{U}_e} \\ \dfrac{\partial \mathbf{R}_e^{U}}{\partial \mathbf{Q}_e} & \dfrac{\partial \mathbf{R}_e^{U}}{\partial \mathbf{U}_e} \end{bmatrix}^{-1}}_{\mathbf{A}_e^{-1}} \begin{bmatrix} \dfrac{\partial \mathbf{R}_e^{Q}}{\partial \mathbf{\Lambda}_g} \\ \dfrac{\partial \mathbf{R}_e^{U}}{\partial \mathbf{\Lambda}_g} \end{bmatrix}$$

From a practical standpoint, when assembling $\mathbf{K}$, we just need a certain amount of local (associated with one element) storage to make the above calculation easier. This local storage consists of matrices that contain linearizations of element interior and face residuals with respect to element interior and face states. The number of faces for which storage is needed is the number of faces adjacent to the element Ω_e. However, not all combinations are nonzero: for example, face residuals are directly affected by only the face states on the same face.

In assembling $\mathbf{K}$ we need, for each element Ω_e, to invert the matrix $\mathbf{A}_e$. This is an element-local matrix but it still can be pretty large, especially in multiple dimensions when we have multiple $\mathbf{Q}$ vectors. However, the block-diagonal entries of this matrix associated with $\mathbf{R}_e^Q$ and $\mathbf{Q}_e$ are just mass matrices, so the inversion can be accelerated by a block decomposition.

The system for a primal Newton update is given by

$$\begin{bmatrix} \dfrac{\partial \mathbf{R}^{Q}}{\partial \mathbf{Q}} & \dfrac{\partial \mathbf{R}^{Q}}{\partial \mathbf{U}} & \dfrac{\partial \mathbf{R}^{Q}}{\partial \mathbf{\Lambda}} \\ \dfrac{\partial \mathbf{R}^{U}}{\partial \mathbf{Q}} & \dfrac{\partial \mathbf{R}^{U}}{\partial \mathbf{U}} & \dfrac{\partial \mathbf{R}^{U}}{\partial \mathbf{\Lambda}} \\ \dfrac{\partial \mathbf{R}^{\Lambda}}{\partial \mathbf{Q}} & \dfrac{\partial \mathbf{R}^{\Lambda}}{\partial \mathbf{U}} & \dfrac{\partial \mathbf{R}^{\Lambda}}{\partial \mathbf{\Lambda}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{Q} \\ \Delta\mathbf{U} \\ \Delta\mathbf{\Lambda} \end{bmatrix} + \begin{bmatrix} \mathbf{R}^{Q} \\ \mathbf{R}^{U} \\ \mathbf{R}^{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{104}$$

or in shorthand with $\mathbf{QU} = [\mathbf{Q}, \mathbf{U}]^T$ denoting the combined $\mathbf{Q}$ and $\mathbf{U}$ vectors,

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{QU} \\ \Delta\mathbf{\Lambda} \end{bmatrix} + \begin{bmatrix} \mathbf{R}^{\mathcal{QU}} \\ \mathbf{R}^{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{105}$$

After static condensation, we obtain a primal system for the face unknown updates,

$$\mathbf{K}\,\Delta\mathbf{\Lambda} + \widetilde{\mathbf{R}}^{\Lambda} = \mathbf{0}, \tag{106}$$

where

$$\widetilde{\mathbf{R}}^{\Lambda} \equiv \mathbf{R}^{\Lambda} - \mathbf{C}\mathbf{A}^{-1}\mathbf{R}^{\mathcal{QU}}. \tag{107}$$

After solving this system for $\Delta\mathbf{\Lambda}$, we can back-substitute to get $\Delta\mathbf{QU}$,

$$\Delta\mathbf{QU} = -\mathbf{A}^{-1}(\mathbf{B}\Delta\mathbf{\Lambda} + \mathbf{R}^{\mathcal{QU}}). \tag{108}$$

6.5 Adjoint Discretization

The adjoint system for a scalar output J is obtained by using the transpose of the primal Jacobian,

$$\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}^{\mathcal{QU}} \\ \mathbf{\Psi}^{\Lambda} \end{bmatrix} + \begin{bmatrix} \frac{\partial J}{\partial \mathbf{QU}}^T \\ \frac{\partial J}{\partial \mathbf{\Lambda}}^T \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{109}$$

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

$$\underbrace{\left[\mathbf{D}^T - \mathbf{B}^T\mathbf{A}^{-T}\mathbf{C}^T\right]}_{\mathbf{K}^T} \mathbf{\Psi}^{\Lambda} + \left[\frac{\partial J}{\partial \mathbf{\Lambda}}^T - \mathbf{B}^T\mathbf{A}^{-T}\frac{\partial J}{\partial \mathbf{QU}}^T\right] = \mathbf{0}. \tag{110}$$

So, as expected, the Schur-complement adjoint system matrix is just the transpose of the corresponding matrix for the primal system. The adjoint residual is also of similar form. Once we solve (110) for $\mathbf{\Psi}^{\Lambda}$ we can back-substitute into (109) and solve for $\mathbf{\Psi}^{\mathcal{QU}}$.

6.6 Error Estimation

The adjoint-weighted residual output error estimate in (40) also applies to an HDG discretization,

$$\delta J \approx \underbrace{-(\mathbf{\Psi}_h^Q)^T\mathbf{R}_h^Q}_{\delta J^Q} \underbrace{-(\mathbf{\Psi}_h^U)^T\mathbf{R}_h^U}_{\delta J^U} \underbrace{-(\mathbf{\Psi}_h^\Lambda)^T\mathbf{R}_h^\Lambda}_{\delta J^\Lambda}, \tag{111}$$

where all the residuals are evaluated using the coarse state injected into the fine space, $\mathbf{Q}_h^H, \mathbf{U}_h^H, \mathbf{\Lambda}_h^H$. For the fine space, we increment the approximation order by one on each element and interface. We obtain the fine-space adjoint by solving exactly or approximately (via smoothing) on this fine space. Note that in (111) we separated the error estimate into three components, one for each residual.

6.7 Localization and Adaptation

From (111) we can localize δJ^Q and δJ^U to elements, since both of these expressions involve residuals associated with test functions on element interiors. Similarly, we can localize δJ^Λ to *faces* because δJ^Λ involves residuals obtained from test functions on faces. What we do with these error indicators depends in part on the chosen adaptation mechanics. In the most general case, we could consider a situation where faces and elements are adapted independently, e.g. through approximation order; we could then incorporate some measure of cost to define a merit function to decide which elements or faces needed to be refined.

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

$$\epsilon_e = \epsilon_e^Q + \epsilon_e^U, \tag{112}$$

where ϵ_e^Q and ϵ_e^U are localizations of δJ^Q and δJ^U, respectively. We could incorporate the localization of δJ^Λ from the faces, by, for example, equally distributing this error indicator to the two adjacent elements. However, in this work we simply ignore these face error indicators when calculating the elemental error indicators for adaptation. The justification for this is that δJ^Λ is typically much smaller in magnitude in comparison to δJ^Q and δJ^U – in fact, in certain cases we can prove that $\delta J^\Lambda = 0$, as discussed in Section 6.8.1 below.

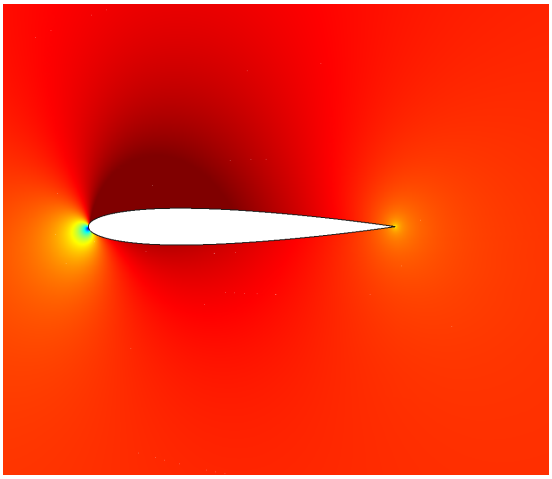
6.8 Examples

In the following examples we demonstrate the utility of output-based adaptation for HDG discretizations of a couple aerodynamic flows. The adaptive strategy is fixed-fraction hanging-node refinement with no coarsening. We present the results in terms of error versus element count to get a baseline comparison of DG and HDG, with the understanding that the number of globally-coupled degrees of freedom can scale with p better for HDG compared to DG, as outlined in Table 3. However, actual benefits of HDG will depend on the efficiency of the local solves, parallelization, and general implementation choices, and making comparisons to other discretizations is still a work in progress.

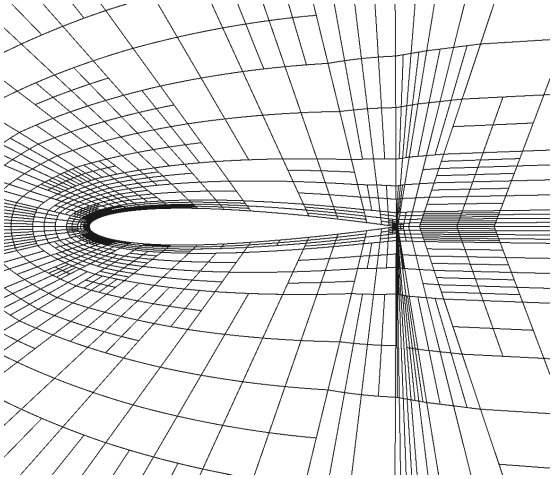
For the below results we use fixed-fraction ($f^{\text{adapt}} = 0.05$) hanging-node adaptation to refine the meshes after computing the output-based error indicator. The HDG discretization, just as DG, extends naturally to hanging nodes. Unknowns are placed on all individual interior faces, so that an element that sits on the "coarse side" of a hanging-node face sees and interacts with multiple independent face states on what would otherwise be just a single face of that element. No major structural changes are necessary to make this happen compared to what is already needed for HDG, as long as data can be linked to all interior faces.

6.8.1 Inviscid Flow over an Airfoil

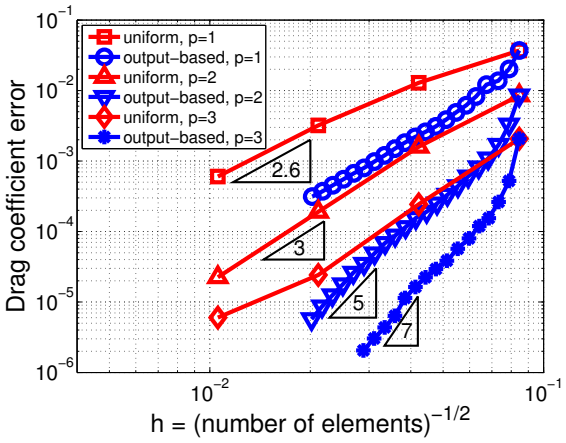
Our first example is inviscid flow over a NACA 0012 airfoil at $M_\infty = 0.5$, $\alpha = 2°$. The output of interest, J, is the drag force coefficient on the airfoil. We are interested in the relative performance of HDG and DG for both uniform and adaptive refinement. Figure 58 presents these results.



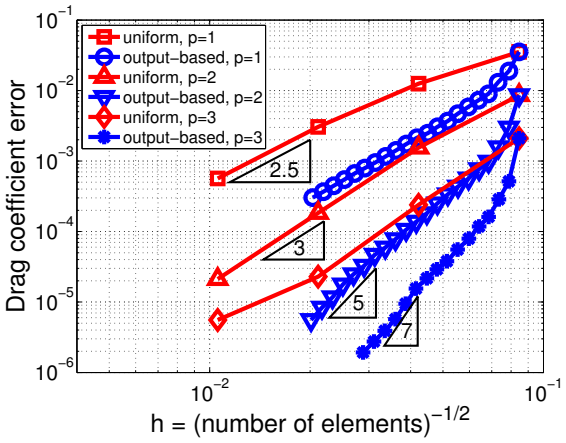
58.1: Mach contours (0–0.6)



58.2: Final HDG drag-adapted mesh for $p = 2$



58.3: HDG Output convergence



58.4: DG Output convergence

Figure 58: HDG and DG adaptation results for an inviscid $M_\infty = 0.5$ flow over a NACA 0012 airfoil at $\alpha = 2°$. The output of interest is the drag force coefficient on the airfoil, and, at each adaptive iteration, the global adjoint system is solved approximately with $\nu = 10$ block Jacobi smoothing iterations on the fine space.

One of the first observations is the striking similarity between the DG and HDG results. We see in the drag coefficient error plots that the two sets of results are nearly identical for the orders tested, $p = 1, 2, 3$. The conclusion from the uniform refinement data is that for a given mesh, DG and HDG produce almost the same answer. An additional conclusion from the output-based adaptive data is that the error indicators obtained by the two discretizations target the same elements for refinement. Hence, the optimal DG also appears to be the optimal HDG mesh.

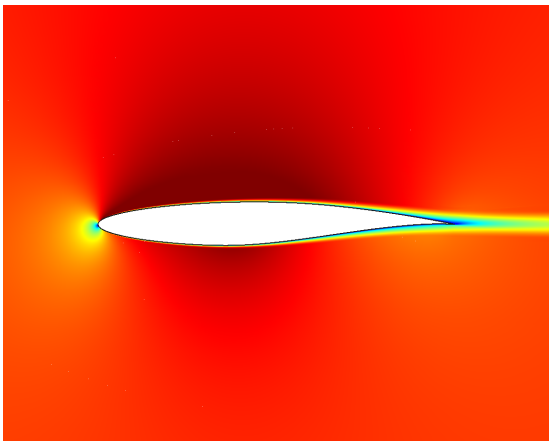
Adaptation in the HDG runs was driven by ϵ_e^U, the localization of the output error contribution from the adjoint-weighted element-interior residuals, δJ^U in (111). There is no approximation here as it turns out that δJ^Λ is zero to residual tolerance. It actually remains zero for any fine space order increment on the faces, under certain conditions that are valid here. The reason for this identically-zero error contribution is that in an inviscid run, the only terms appearing in the weak flux continuity statement, (89), are those involving the stabilization terms. Considering one face, if the same approximation order, p, is used for the states in the two adjacent elements and on the shared face, and if the stabilization matrix is constant, then the integrand in (89) is just a polynomial of order p. Thus, testing against order p functions makes the integrand point-wise zero, so that there is no residual left for the fine space to uncover.

Finally, we note that although the meshes produced by HDG and DG are comparable, the number of globally-coupled degrees of freedom can be lower in the HDG results. For example, at $p = 2$, the final drag-adapted mesh for DG contained 22230 degrees of freedom (not including the state rank s), whereas the final drag-adapted HDG mesh contained 15756 degrees of freedom – the number of elements was nearly the same, 2470 versus 2458. The degree of freedom difference becomes even more favorable for HDG with increasing order, although more studies are required to understand what roles matrix fill and system stiffness should play in this comparison.

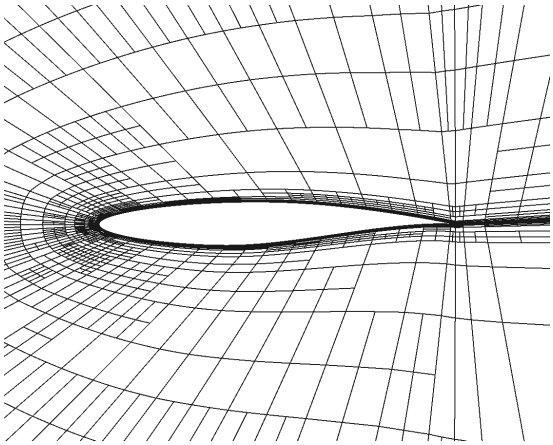
6.8.2 Turbulent Flow over an Airfoil

In this example we apply output-based adaptation to an HDG discretization of turbulent flow over an RAE 2822 airfoil. The conditions are $M_\infty = 0.5$, $Re = 10^5$, $\alpha = 1^\circ$, and we use the Spalart-Allmaras one-equation turbulence model to close the Reynolds-averaged Navier-Stokes equations. For the adaptive runs, we target the drag coefficient output.

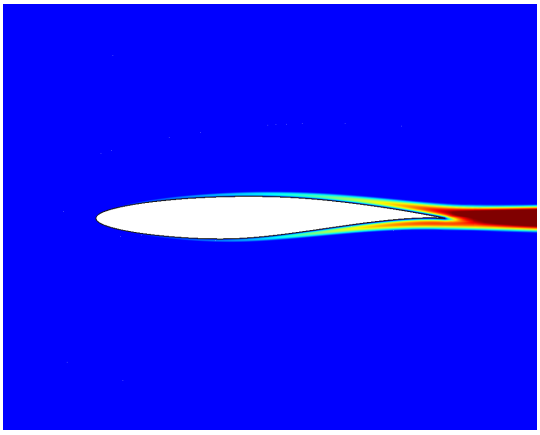
Figure 59 shows the flow-field, mesh, and output convergence results for $p = 2$ approximation. The initial mesh consists of 798 quartic curved elements. As in the previous example, the DG and HDG results are quite close, for both the uniform and adaptive refinement runs. The largest differences occur on the coarsest mesh, where differences in the stabilization are most prominent. We note that output-based adaptation converges rapidly at first and then slows down. The slow-down is due to proximity of the farfield boundary (100 chords), where we impose a full-state boundary condition that attracts refinement.



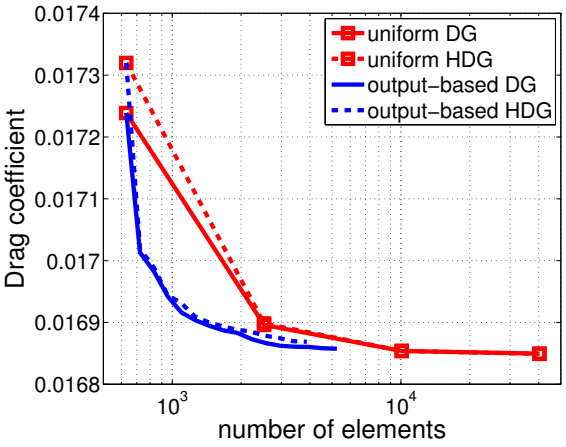
59.1: Adapted-mesh Mach contours (0–0.6)



59.2: Final HDG drag-adapted mesh



59.3: Adapted-mesh SA working variable, $\tilde{\nu}$



59.4: Output convergence

Figure 59: HDG and DG adaptation results for Reynolds-averaged turbulent flow, $M_\infty = 0.5$, over an RAE 2822 airfoil at $\alpha = 1°$, using $p = 2$ approximation. The output of interest is the drag force coefficient on the airfoil, and, at each adaptive iteration, the global adjoint system is solved approximately with $\nu = 10$ element-block Jacobi smoothing iterations on the $p + 1$ fine space.

6.8.3 Scalar Advection-Diffusion

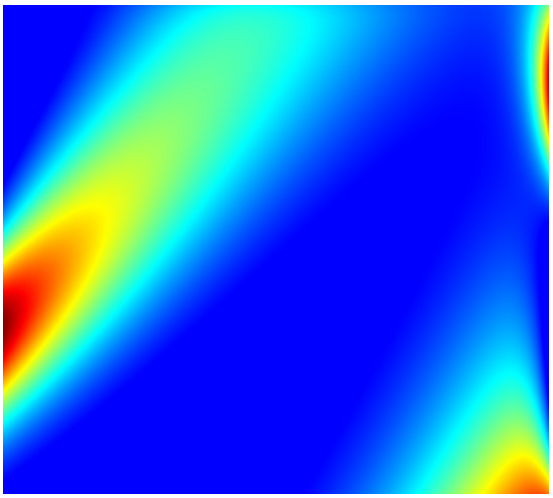
After having looked at a couple aerodynamic flows, we now take a step back to consider a simple advection-diffusion equation,

$$\vec{V} \cdot \nabla u - \mu \nabla^2 u = 0, \tag{113}$$

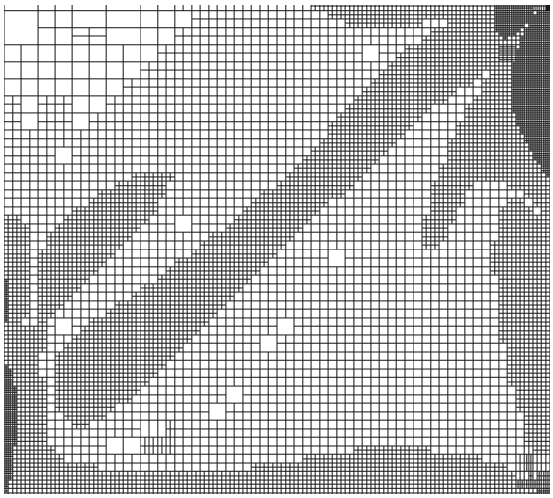
where $\vec{V} = [0.6, 0.8]$, and $\mu = 0.02$. The domain is a unit square, $(x, y) \in [0, 1]^2$, and a Dirichlet boundary condition function is imposed,

$$u = \exp\left[0.5 \sin(-4x + 6y) - 0.8 \cos(3x - 8y)\right]. \tag{114}$$

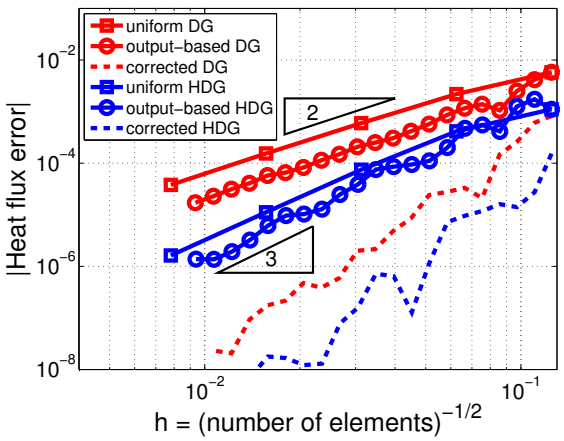
For the stabilization term in HDG, we set $\tau = |\vec{V}| + \nu$, so that the viscous length scale is unity. A sample solution is shown in Figure 60.1. The output of interest is the integral of the flux, $-\mu \nabla u$ over the right boundary of the square.



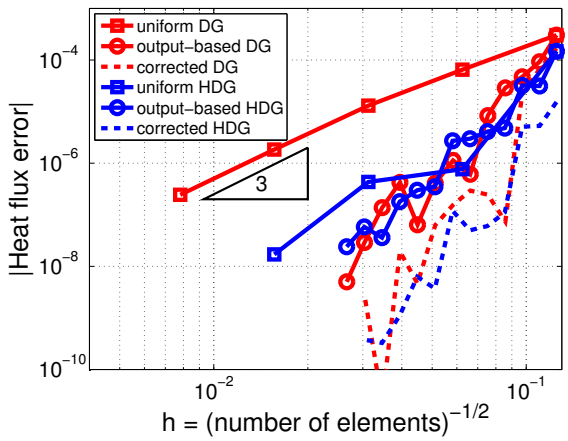
60.1: Scalar distribution



60.2: Final HDG adapted mesh



60.3: $p = 1$ output convergence



60.4: $p = 2$ output convergence

Figure 60: HDG and DG adaptation results for a manufactured solution to a scalar advection-diffusion problem. For the output-based adaptive results, the $p + 1$ fine-space adjoint problem was solved exactly.

Figure 60 compares uniform refinement and output-based adaptive results for DG and HDG discretizations applied to this problem. In the adaptive runs, we employ a fixed-fraction strategy with $f^{\text{adapt}} = 0.1$ and we solve the fine-space adjoint problem exactly. The results here differ from the previous ones in this section in that now we see a difference in the convergence rates for HDG and DG, even with uniform refinement: HDG exhibits an improved rate by 1 compared to DG. This is due in part to the mixed formulation of HDG, in which the flux $\vec{q}$ converges faster than in non-mixed formulations because $\vec{q}$ is approximated as a separate variable. Another way to take advantage of such super-convergence is to locally post-process HDG solutions [78], and this could further tilt the scales in HDG's favor.

We also note from Figure 60 that the raw outputs from the adaptive runs do not always beat uniform refinement. In this problem much of the domain is important for the prediction of the output, and hence uniform refinement does reasonably well. The corrected values, however, converge at an improved order, as expected since we solve the fine-space adjoint problem exactly. Finally, we note that for HDG, the error contribution from the face residuals, δJ^{Λ}, is zero to machine precision, so that using the element-interior error indicator is appropriate.

7 Conclusions

In this set of lecture notes we have presented some of the core ideas, methods, and results for high-order output-based adaptive simulations of aerodynamic flows. Many of the presented methods can be generalized to more than one discretization; we chose the discontinuous Galerkin (DG) finite element method for various reasons, including ease of high-order implementation, adaptive flexibility including natural support for hp-adaptation, parallel scalability, and of course code availability. One notable drawback of DG is its computational expense, and to address this we have also presented an extension of our methods to a hybrid discontinuous Galerkin (HDG) discretization, in which the globally-coupled unknowns scale with p to a dimension one lower compared to DG.

Numerical error estimates that use an adjoint solution to target a specific scalar output are particularly useful for convection-dominated aerodynamic simulations, in which even seemingly-small errors in one area of the computational domain can affect a large portion of the flow field and significantly impact an output of interest. Weighting residuals by the adjoint solution produces an error estimate that takes such propagation effects into account. The result is a type of "error bar" on the chosen output of the numerical simulation. The cost of this error estimate lies in the computation of the adjoint solution via a linear solve, possibly in an enriched "fine" approximation space. For the complex nonlinear systems often encountered in aerodynamics, even a fine-space linear solve is generally cheaper than the solution of the original primal system on the coarse space.

Moreover, because an output-error estimate is residual-based, it can be localized to cells/elements in a finite volume/element calculation for the purpose of mesh adaptation. Targeted local adaptation, such as refinement of a fraction of the elements with the highest indicator, reduces local residuals, which then reduces the local contributions of the targeted elements to the error. As we have shown, this concept extends to a wide variety of complex problems, including three-dimensional, transonic, Reynolds-averaged turbulent flow over a wing, and unsteady simulations on deforming domains. Below we highlight some key conclusions garnered from the presented examples.

- Output-based adaptation can quarantine singularities, as long as they do not span too much of the domain, so that these singularities do not affect the results and so that optimal convergence rates, such as $2p+1$ or $2p$ for inviscid or viscous problems, are recovered. Uniform refinement, on the other hand, is limited by the lowest-order singularity.

- For many steady problems, output-based adaptation shows benefits compared to uniform refinement or heuristic indicators in both degrees of freedom and computational time. The latter is especially true for three-dimensional problems with complex physics, e.g. RANS, in which the cost of an adjoint solve is low relative to that of the primal solve.

- The mechanics of adaptation can be difficult in situations when curved anisotropic elements are required, in particular in three dimensions. We have presented adaptation results using hanging-node refinement of initially-structured quadrilateral and hexahedral meshes, and this works quite well in the near-body region, where the flow follows the body. However, such an approach requires a reasonable initial mesh

and exhibits much more limited resolution capabilities compared to anisotropic unstructured meshes. Improving the robustness of the latter is still an active research area.

- Unsteady problems add another dimension – time – as a candidate for refinement, and this generally bodes well for adaptation, as additional dimensions often translate to a larger portion of the domain being "unimportant" for the prediction of an output. With an ALE formulation, it is possible to handle complicated problems involving geometry deformation, and we have demonstrated adaptation for such problems with dynamic p refinement and coarsening. The adjoint becomes more expensive in the unsteady case due to storage requirements for nonlinear problems. Yet, for sufficiently strict error tolerances, we still observe benefits in computational time relative to uniform refinement and heuristic indicators. Adaptation becomes even trickier in unsteady problems with the added complexity of temporal refinement, and we have presented a projection-based measure of space-time anisotropy that helps guide the space versus time refinement decision.

- Hybrid DG discretizations can yield systems that are smaller in globally-coupled degrees of freedom for moderate-to-high orders relative to standard DG. In addition, an HDG discretization is modular and relatively straightforward, especially in its diffusion treatment. The discrete adjoint extends naturally to HDG, and error estimates now contain contributions from the three different equations – not all of these contributions are of the same magnitude, and some are provably zero. Our examples show that HDG and DG produce comparable solutions on a given mesh, and that they gravitate towards the same adapted meshes under output-based refinement. HDG shows benefits in degrees of freedom, especially for high order, and additional studies are underway to understand the full cost of an HDG discretization, with its two categories of degrees of freedom, and to fairly compare computational time.

Finally, adaptive high-order methods likely are not yet being used to their full potential. Additional challenges that need to be addressed include: reducing the cost of adjoint solves, in particular for unsteady problems; applying adjoint methods to chaotic simulations; calculating bounds instead of just estimates of the output error; investigating the possibility of cheaper surrogates such as the entropy adjoint; developing better adaptive mechanics for complex spatial meshes; and generalization of the unsteady adaptive methods to other discretizations, such as multi-step or multi-stage time integration.

8 Acknowledgments

These notes present research results produced over several years by myself and my research group at the University of Michigan. I gratefully acknowledge the contributions of undergraduate students, graduate research assistants, and post-doctoral scholars to the various research projects summarized in these notes. I am also thankful for last-minute proofreading comments and suggestions from members of my current research group that helped improve these notes. Finally, I would like to acknowledge the financial support of the University of Michigan, the U.S. Air Force Office of Scientific Research (grant FA9550-11-1-0081), the U.S. Department of Energy (grant DE-SC0010341), and student fellowships from the U.S. National Science Foundation and the U.S. Department of Defense.

References

[1] M.J. Aftosmis and M.J. Berger. Multilevel error estimation and adaptive h-refinement for Cartesian meshes with embedded boundaries. AIAA Paper 2002-14322, 2002.

[2] Mark Ainsworth and Bill Senior. An adaptive refinement strategy for *hp*-finite element computations. *Applied Numerical Mathematics*, 26:165–178, 1998.

[3] Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and L. Donatella Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[4] Timothy J. Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25:243–273, 1997.

[5] Pinhas Bar-Yoseph and David Elata. An efficient L2 Galerkin finite element method for multi-dimensional non-linear hyperbolic systems. *International Journal for Numerical Methods in Engineering*, 29:1229–1245, 1990.

[6] Garrett E. Barter. *Shock Capturing with PDE-Based Artificial Viscosity for an Adaptive Higher–Order Discontinuous Galerkin Finite Element Method*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[7] Timothy Barth and Mats Larson. A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes. In R. Herban and D. Kröner, editors, *Finite Volumes for Complex Applications III*, pages 41–63, London, 2002. Hermes Penton.

[8] Timothy J. Barth. Space-time error representation and estimation in Navier-Stokes calculations. In Stavros C. Kassinos, Carlos A. Langer, Gianluca Iaccarino, and Parviz Moin, editors, *Complex Effects in Large Eddy Simulations*, pages 29–48. Springer Berlin Heidelberg, Lecture Notes in Computational Science and Engineering Vol 26, 2007.

[9] F. Bassi and S. Rebay. GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations. In Karniadakis Cockburn and Shu, editors, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, pages 197–208. Springer, Berlin, 2000.

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

[11] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: Basic analysis and examples. *East-West Journal of Numerical Mathematics*, 4(4):237–264, 1996.

[12] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. In A. Iserles, editor, *Acta Numerica*, pages 1–102. Cambridge University Press, 2001.

[13] M.J. Berger and A. Jameson. Automatic adaptive grid refinement for the Euler equations. *AIAA Journal*, 23:561–568, 1985.

[14] Kim S. Bey and J. Tinsley Oden. *hp*-version discontinuous Galerkin methods for hyperbolic conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 133:259–286, 1996.

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

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

[17] Nicholas K. Burgess and Dimitri J. Mavriplis. An *hp*-adaptive discontinuous Galerkin, solver for aerodynamic flows on mixed-element meshes. AIAA Paper 2011-490, 2011.

[18] Gustavo C. Buscaglia and Enzo A. Dari. Anisotropic mesh optimization and its application in adaptivity. *International Journal for Numerical Methods in Engineering*, 40(22):4119–4136, November 1997.

[19] P. J. Capon and P. K. Jimack. On the adaptive finite element solution of partial differential equations using h-r refinement. Technical Report 96.03, University of Leeds, School of Computing, 1996.

[20] M. J. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaptation for flow simulations. *International Journal for Numerical Methods in Fluids*, 25:475–491, 1997.

[21] Marco A. Ceze and Krzysztof J. Fidkowski. Output-driven anisotropic mesh adaptation for viscous flows using discrete choice optimization. AIAA Paper 2010-0170, 2010.

[22] Marco A. Ceze and Krzysztof J. Fidkowski. A robust adaptive solution strategy for high-order implicit CFD solvers. AIAA Paper 2011-3696, 2011.

[23] Marco A. Ceze and Krzysztof J. Fidkowski. An anisotropic hp-adaptation framework for functional prediction. *American Institute of Aeronautics and Astronautics Journal*, 51:492–509, 2013.

[24] Bernardo Cockburn, Bo Dong, Johnny Guzman, Marco Restelli, and Riccardo Sacco. A hybridizable discontinuous Galerkin method for steady-state convection-diffusion-reaction problems. *SIAM Journal on Scientific Computing*, 31(5):3827–3846, 2009.

[25] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[26] L. Demkowicz, W. Rachowicz, and Ph. Devloo. A fully automatic hp-adaptivity. *Journal of Scientific Computing*, 17:117–142, 2002.

[27] Bruno Despres. Lax theorem and finite volume schemes. *Mathematics of Computation*, 73(247):1203–1234, 2003.

[28] Julien Dompierre, Marie-Gabrielle Vallet, Yves Bourgault, Michel Fortin, and Wagdi G. Habashi. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III: Unstructured meshes. *International Journal for Numerical Methods in Fluids*, 39:675–702, 2002.

[29] K. J. Fidkowski, M. A. Ceze, and P. L. Roe. Entropy-based drag error estimation and mesh adaptation in two dimensions. *AIAA Journal of Aircraft*, 49(5):1485–1496, September-October 2012.

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

[31] Krzysztof J. Fidkowski. *A Simplex Cut-Cell Adaptive Method for High–order Discretizations of the Compressible Navier-Stokes Equations*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2007.

[32] Krzysztof J. Fidkowski. Output error estimation strategies for discontinuous Galerkin discretizations of unsteady convection-dominated flows. *International Journal for Numerical Methods in Engineering*, 88(12):1297–1322, 2011.

[33] Krzysztof J. Fidkowski. An output-based dynamic order refinement strategy for unsteady aerodynamics. AIAA Paper 2012-77, 2012.

[34] Krzysztof J. Fidkowski and David L. Darmofal. An adaptive simplex cut-cell method for discontinuous Galerkin discretizations of the Navier-Stokes, equations. AIAA Paper 2007-3941, 2007.

[35] Krzysztof J. Fidkowski and David L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *American Institute of Aeronautics and Astronautics Journal*, 49(4):673–694, 2011.

[36] Krzysztof J. Fidkowski and Yuxing Luo. Output-based space-time mesh adaptation for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 230:5753–5773, 2011.

[37] Krzysztof J. Fidkowski, Todd A. Oliver, James Lu, and David L. Darmofal. *p*-Multigrid solution of high–order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 207:92–113, 2005.

[38] Krzysztof J. Fidkowski and Philip L. Roe. An entropy adjoint approach to mesh refinement. *SIAM Journal on Scientific Computing*, 32(3):1261–1287, 2010.

[39] Luca Formaggia and Simona Perotto. New anisotropic a priori error estimates. *Numerische Mathematik*, 89(4):641–667, 2001.

[40] Luca Formaggia, Simona Perotto, and Paolo Zunino. An anisotropic a posteriori error estimate for a convection-diffusion problem. *Computing and Visualization in Science*, 4:99–104, 2001.

[41] Lori A. Freitag and Carl Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40:3979–4002, 1997.

[42] Neal T. Frink. 3rd AIAA CFD drag prediction workshop gridding guidelines. NASA Langley, 2007. `http://aiaa-dpw.larc.nasa.gov/Workshop3/gridding_guidelines.html`.

[43] Neal T. Frink. Test case results from the 3rd AIAA drag prediction workshop. NASA Langley, 2007. `http://aiaa-dpw.larc.nasa.gov/Workshop3/workshop3.html`.

[44] M. B. Giles and N. A. Pierce. Adjoint equations in CFD: duality, boundary conditions and solution behavior. AIAA Paper 97-1850, 1997.

[45] M. B. Giles and E. Süli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. In *Acta Numerica*, volume 11, pages 145–236, 2002.

[46] M.B. Giles and N.A. Pierce. Analytic adjoint solutions for the quasi-one-dimensional Euler equations. *Journal of Fluid Mechanics*, 426:327–345, 2001.

[47] Andreas Griewank and Andrea Walther. Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software*, 26(1):19–45, 2000.

[48] W. Gui and I. Babuška. The h, p, and $h-p$ versions of the finite element method in 1 dimension. Part III. the adaptive $h-p$ version. *Numerische Mathematik*, 49(6):659–683, 1986.

[49] Wagdi G. Habashi, Julien Dompierre, Yves Bourgault, Djaffar Ait-Ali-Yahia, Michel Fortin, and Marie-Gabrielle Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles. *International Journal for Numerical Methods in Fluids*, 32:725–744, 2000.

[50] K. Harriman, D. Gavaghan, and E. Süli. The importance of adjoint consistency in the approximation of linear functionals using the discontinuous Galerkin finite element method. Technical Report Technical Report NA 04/18, Oxford University Computing Lab Numerical Analysis Group, 2004.

[51] K. Harriman, P. Houston, B. Senior, and Endre Süli. hp-version discontinuous Galerkin methods with interior penalty for partial differential equations with non-negative characteristic form. Technical Report Technical Report NA 02/21, Oxford University Computing Lab Numerical Analysis Group, 2002.

[52] R. Hartmann and P. Houston. Error estimation and adaptive mesh refinement for aerodynamic flows. In H. Deconinck, editor, *36^{th} CFD/ADIGMA course on hp-adaptive and hp-multigrid methods: VKI Lecture Series 2010-01 (Oct. 26-30, 2009)*. von Karman Institute for Fluid Dynamics, 2010.

[53] Ralf Hartmann. Adjoint consistency analysis of discontinuous Galerkin discretizations. *SIAM Journal on Numerical Analysis*, 45(6):2671–2696, 2007.

[54] Ralf Hartmann and Paul Houston. Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. *Journal of Computational Physics*, 183(2):508–532, 2002.

[55] F. Hecht. BAMG: Bidimensional anisotropic mesh generator. INRIA–Rocquencourt, France, 1998. `www.freefem.org`.

[56] V. Heuveline and R. Rannacher. Duality-based adaptivity in the *hp*-finite element method. *Journal of Numerical Mathematics*, 11(2):95–113, 2003.

[57] P. Houston, R. Hartmann, and E. Süli. Adaptive discontinuous Galerkin finite element methods for compressible fluid flows. In M. Baines, editor, *Numerical Methods for Fluid Dynamics VII, ICFD*, pages 347–353, 2001.

[58] P. Houston, B. Senior, and E. Süli. Sobolev regularity estimation for *hp*-adaptive finite element methods. In F. Brezzi, A. Buffa, S. Corsaro, and A. Murli, editors, *Numerical Mathematics and Advanced Applications*, pages 619–644. Springer-Verlag, 2003.

[59] P. Houston and E. Süli. *hp*-adaptive discontinuous Galerkin finite element methods for first-order hyperbolic problems. *SIAM Journal on Scientific Computing*, 23(4):1226–1252, 2001.

[60] Paul Houston, Emmanuil H. Georgoulis, and Edward Hall. Adaptivity and a posteriori error estimation for DG methods on anisotropic meshes. In G. Lube and G. Rapin, editors, *Proceedings of the International Conference on Boundary and Interior Layers (BAIL)*. University of Göttingen, 2006.

[61] Paul Houston, Bill Senior, and Endre Süli. hp-Discontinuous Galerkin finite element methods for hyperbolic problems: Error analysis and adaptivity. *International Journal for Numerical Methods in Fluids*, 40:153–169, 2002.

[62] Paul Houston and Endre Süli. A note on the design of hp-adaptive finite element methods for elliptic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194:229–243, 2005.

[63] Hans Johansen and Philip Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *Journal of Computational Physics*, 147:60–85, 1998.

[64] Steven M. Kast and Krzysztof J. Fidkowski. Output-based mesh adaptation for high order Navier-Stokes simulations on deformable domains. *Journal of Computational Physics*, 252(1):468–494, 2013.

[65] C.M. Klaij, J.J.W. van der Vegt, and H. van der Ven. Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 217:589–611, 2006.

[66] Tobias Leicht and Ralf Hartmann. Multitarget error estimation and adaptivity in aerodynamic flow simulations. *International Journal for Numerical Methods in Fluids*, 56:2111–2138, 2008.

[67] Robert B. Lowrie, Philip L. Roe, and Bram van Leer. Properties of space-time discontinuous Galerkin. Los Alamos Technical Report LA-UR-98-5561, 1998.

[68] James Lu. *An a Posteriori Error Control Framework for Adaptive Precision Optimization Using Discontinuous Galerkin Finite Element Method*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2005.

[69] Karthik Mani and Dimitri J. Mavriplis. Discrete adjoint based time-step adaptation and error reduction in unsteady flow problems. AIAA Paper 2007-3944, 2007.

[70] Karthik Mani and Dimitri J. Mavriplis. Error estimation and adaptation for functional outputs in time-dependent flow problems. *Journal of Computational Physics*, 229:415–440, 2010.

[71] D. J. Mavriplis. Adaptive mesh generation for viscous flows using Delaunay triangulation. *Journal of Computational Physics*, 90:271–291, 1990.

[72] D. Scott McRae. r-Refinement grid adaptation algorithms and issues. *Computer Methods in Applied Mechanics and Engineering*, 2000(4):1161–1182, 189.

[73] D. Meidner and B. Vexler. Adaptive space-time finite element methods for parabolic optimization problems. *SIAM Journal on Control Optimization*, 46(1):116–142, 2007.

[74] Peter K. Moore. Applications of lobatto polynomials to an adaptive finite element method: A posteriori error estimates for hp-adaptivity and grid-to-grid interpolation. *Numerische Mathematik*, 94:367–401, 2003.

[75] Marian Nemec and Michael J. Aftosmis. Error estimation and adaptive refinement for embedded-boundary Cartesian meshes. AIAA Paper 2007-4187, 2007.

[76] Marian Nemec, Michael J. Aftosmis, and Mathias Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. AIAA Paper 2008-0725, 2008.

[77] N. C. Nguyen, J. Peraire, and B. Cockburn. Hybridizable discontinuous Galerkin methods. In Jan S. Hesthaven, Einar M. RÃÿnquist, Barth Timothy J., Michael Griebel, David E. Keyes, Risto M. Nieminen, Dirk Roose, and Tamar Schlick, editors, *Spectral and High Order Methods for Partial Differential Equations*, volume 76 of *Lecture Notes in Computational Science and Engineering*, pages 63–84. Springer Berlin Heidelberg, 2011. 10.1007/978-3-642-15337-2_4.

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

[79] Todd A. Oliver. *A High–order, Adaptive, Discontinuous Galerkin, Finite Elemenet Method for the Reynolds-Averaged Navier-Stokes Equations*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[80] Todd A. Oliver and David L. Darmofal. Impact of turbulence model irregularity on high–order discretizations. AIAA Paper 2009-953, 2009.

[81] Doug Pagnutti and Carl Ollivier-Gooch. A generalized framework for high order anisotropic mesh adaptation. *Computers and Structures*, 87(11-12):670 – 679, 2009.

[82] M. A. Park. Adjoint-based, three-dimensional error prediction and grid adaptation. AIAA Paper 2002-3286, 2002.

[83] M. A. Park. Three–dimensional turbulent RANS adjoint–based error correction. AIAA Paper 2003-3849, 2003.

[84] Michael A. Park. *Anisotropic Output-Based Adaptation with Tetrahedral Cut Cells for Compressible Flows*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2008.

[85] J. Peraire and P.-O. Persson. The compact discontinuous Galerkin (CDG) method for elliptic problems. *SIAM Journal on Scientific Computing*, 2007.

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

[87] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. AIAA Paper 2006-112, 2006.

[88] Niles A. Pierce and Michael B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42(2):247–264, 2000.

[89] W. Rachowicz, L. Demkowicz, and J.T. Oden. Toward a universal $h-p$ adaptive finite element strategy, part 3. design of $h-p$ meshes. *Computer Methods in Applied Mechanics and Engineering*, 77:181–212, 1989.

[90] R. Rannacher. Adaptive Galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128:205–233, 2001.

[91] Sander Rhebergen and Bernardo Cockburn. A spaceâĂŞtime hybridizable discontinuous Galerkin method for incompressible flows on deforming domains. *Journal of Computational Physics*, 231(11):4185 – 4204, 2012.

[92] Thomas Richter. A posteriori error estimation and anisotropy detection with the dual-weighted residual method. *International Journal for Numerical Methods in Fluids*, 62:90–118, 2010.

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

[94] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[95] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 7(3):856–869, 1986.

[96] E. Schall, D. Leservoisier, A. Dervieux, and B. Koobus. Mesh adaptation as a tool for certified computational aerodynamics. *International Journal for Numerical Methods in Fluids*, 45(2):179–196, 2004.

[97] Michael Schmich and Boris Vexler. Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations. *SIAM Journal on Scientific Computing*, 30(1):369–393, 2008.

[98] R. Schneider and P. K. Jimack. Toward anisotropic mesh adaptation based upon sensitivity of a posteriori estimates. Technical Report 2005.03, University of Leeds, School of Computing, 2005.

[99] J.R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22:21–74, 2002.

[100] Kunibert G. Siebert. An a posteriori error estimator for anisotropic refinement. *Numerische Mathematik*, 73:373–398, 1996.

[101] P. Solín and L. Demkowicz. Goal-oriented hp-adaptivity for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 193:449–468, 2004.

[102] Shuyu Sun and Mary Wheeler. Mesh adaptation strategies for discontinuous Galerkin methods applied to reactive transport problems. In H.W. Chu, M. Savoie, and B. Sanchez, editors, *International Conference on Computing, Communication and Control Technologies*, volume 1, pages 223–228, Austin, Texas, August 2004.

[103] Shuyu Sun and Mary F. Wheeler. Anisotropic and dynamic mesh adaptation for discontinuous Galerkin methods applied to reactive transport. Technical Report 05-15, ICES, 2005.

[104] Barna A. Szabo. Estimation and control of error based on p convergence. In I. Babuška, O. C. Zienkiewicz, J. Gago, and E. R. de Oliveira, editors, *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, pages 61–78. John wiley & Sons Ltd., 1986.

[105] J.J.W. van der Vegt. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. National Aerospace Laboratory NLR-TP-98239, 1998.

[106] H. van der Ven and J.J.W. van der Vegt. Space-time discontinuous Galerkin fintie element method with dynamic grid motion for inviscid compressible flows II. Efficient flux quadrature. *Computer Methods in Applied Mechanics and Engineering*, 191:4747–4780, 2002.

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

[108] Z.J. Wang, Krzysztof Fidkowski, Remi Abgrall, Francesco Bassi, Doru Caraeni, Andrew Cary, Herman Deconinck, Ralf Hartmann, Koen Hillewaert, H.T. Huynh, Norbert Kroll, Georg May, Per-Olof Persson, Bram van Leer, and Miguel Visbal. High-order CFD methods: Current status and perspective. *International Journal for Numerical Methods in Fluids*, 2013.

[109] William A. Wood and William L. Kleb. On multi-dimensional unstructured mesh adaptation. AIAA Paper 99-3254, 1999.

[110] Guoping Xia, Ding Li, and Charles L. Merkle. Anisotropic grid adaptation on unstructured meshes. AIAA Paper 2001-0443, 2001.

[111] Nail K. Yamaleev, Boris Diskin, and Eric J. Nielsen. Local-in-time adjoint-based method for design optimization of unsteady flows. *Journal of Computational Physics*, 229:5394–5407, 2010.

[112] M. Yano, J.M. Modisette, and D.L. Darmofal. The importance of mesh adaptation for higher-order discretizations of aerodynamics flows. AIAA Paper 2011-3852, 2011.

[113] Masayuki Yano. *An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.

A Compressible Navier-Stokes Equations

This appendix presents the Euler, compressible Navier-Stokes, and Reynolds-averaged compressible Navier-Stokes equations.

A.1 Euler Equations

The Euler equations of gas dynamics are

$$\begin{array}{lcll} \partial_t \rho & + & \partial_i(\rho u_i) & = & 0 \\ \partial_t(\rho u_j) & + & \partial_i(\rho u_i u_j + p\delta_{ij}) & = & 0 \\ \partial_t(\rho E) & + & \partial_i(\rho u_i H) & = & 0 \end{array}$$

where ρ is the density ρu_j is the j^{th} momentum component, and ρE is the total energy. The pressure and total enthalpy are given by

$$\begin{aligned} p &= (\gamma - 1)\left(\rho E - \frac{1}{2}\rho u_k u_k\right), \\ H &= E + \frac{p}{\rho}. \end{aligned}$$

Note, i, j, k index the spatial dimensions and summation is implied over repeated indices. These equations can be written in compact conservation form as

$$\partial_t \mathbf{u} + \partial_i \mathbf{F}_i = \mathbf{0},$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u_j \\ \rho E \end{bmatrix}, \quad \mathbf{F}_i = \begin{bmatrix} \rho u_i \\ \rho u_i u_j + p\delta_{ij} \\ \rho u_i H \end{bmatrix}.$$

A.2 Compressible Navier-Stokes

The compressible Navier-Stokes equations are given by

$$\begin{array}{lcll} \partial_t \rho & + & \partial_i(\rho u_i) & = & 0 \\ \partial_t(\rho u_i) & + & \partial_i(\rho u_i u_j + p\delta_{ij}) & = & \partial_i \tau_{ij} \\ \partial_t(\rho E) & + & \partial_i(\rho u_i H) & = & \partial_i(\tau_{ij} u_j - q_i) \end{array}$$

where the viscous stress tensor and the heat flux vector are

$$\begin{aligned} \tau_{ij} &= \mu(\partial_i u_j + \partial_j u_i) + \delta_{ij}\lambda \partial_m u_m, \\ q_i &= -\kappa_T \partial_i T. \end{aligned}$$

Relevant physical quantities for air are,

$$\begin{aligned}
\text{Dynamic viscosity:} \quad \mu &= \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^{1.5} \left(\frac{T_{\text{ref}} + T_{\text{s}}}{T + T_{\text{s}}} \right), \\
&\text{(Sutherland's law: } T_{\text{ref}} = 288.15\text{K},\ T_{\text{s}} = 110\text{K)} \\
\text{Bulk viscosity coefficient:} \quad \lambda &= -\frac{2}{3}\mu, \\
\text{Kinematic viscosity:} \quad \nu &= \frac{\mu}{\rho}, \\
\text{Thermal conductivity:} \quad \kappa_T &= \frac{\gamma \mu R}{(\gamma - 1) Pr}, \\
\text{Specific-heat ratio:} \quad \gamma &= 1.4, \\
\text{Prandtl number:} \quad Pr &= 0.71, \\
\text{Gas constant:} \quad R.
\end{aligned}$$

To clarify the viscous implementation, we make the dependence of the viscous flux on the gradient of the state vector explicit by writing the equations in the form

$$\partial_t \mathbf{u} + \partial_i \mathbf{F}_i - \partial_i (\mathbf{K}_{ij} \partial_j \mathbf{u}) = \mathbf{0},$$

where $\mathbf{u} = [\rho, \rho u_j, \rho E]^T$ is the state vector and $\vec{\mathbf{F}}$ is the inviscid flux vector. The viscous flux coefficient matrices can be written as

$$\mathbf{K}_{ij} = \begin{bmatrix} 0 & \mathbf{0} & 0 \\ K_{ij}^{\rho u_d, \rho} & K_{ij}^{\rho u_d, \rho u_c} & \mathbf{0} \\ K_{ij}^{\rho E, \rho} & K_{ij}^{\rho E, \rho u_c} & K^{\rho E, \rho E} \end{bmatrix},$$

where for a given pair i, j,

$$\begin{aligned}
K_{ij}^{\rho u_d, \rho} = (\dim \times 1) \text{ matrix} &= \nu \left(u_d \delta_{ij} - u_i \delta_{jd} + \frac{2}{3} u_j \delta_{id} \right) \\
K_{ij}^{\rho u_d, \rho u_c} = (\dim \times \dim) \text{ matrix} &= \nu \left(\delta_{dc} \delta_{ij} + \delta_{dj} \delta_{ci} - \frac{2}{3} \delta_{di} \delta_{cj} \right) \\
K_{ij}^{\rho E, \rho} = (1 \times 1) \text{ matrix} &= \nu \left(-u_k u_k \delta_{ij} - \frac{1}{3} u_i u_j + \kappa_T T_\rho \right) \\
K_{ij}^{\rho E, \rho u_c} = (1 \times \dim) \text{ matrix} &= \nu \left(-\frac{2}{3} u_i \delta_{cj} + u_j \delta_{ci} + u_c \delta_{ij} + \kappa_T T_{\rho u_c} \right) \\
K_{ij}^{\rho E, \rho E} = (1 \times 1) \text{ matrix} &= \nu \left(\kappa_T T_\rho \right)
\end{aligned}$$

and the temperature derivative vector is

$$T_{\mathbf{u}} = [T_\rho, T_{\rho u_c}, T_{\rho E}] = \frac{\gamma - 1}{R\rho} \left[-E + u_k u_k, -u_c, 1 \right].$$

Note that the indices d, c, k index the spatial dimension.

A.3 Reynolds-Averaged Compressible Navier-Stokes

We use the Spalart-Allmaras one-equation turbulence model to close the Reynolds-averaged Navier-Stokes (RANS) equations. The augmented equations are

$$\begin{array}{lcll} \partial_t \rho & + & \partial_i(\rho u_i) & = & 0 \\ \partial_t(\rho u_i) & + & \partial_i(\rho u_i u_j + p\delta_{ij}) & = & \partial_i \tau_{ij} \\ \partial_t(\rho E) & + & \partial_i(\rho u_i H) & = & \partial_i(\tau_{ij} u_j - q_i) \\ \partial_t(\rho \tilde{\nu}) & + & \partial_i(\rho u_i \tilde{\nu}) & = & \partial_i\left(\frac{\eta}{\sigma}\partial_i \tilde{\nu}\right) + \frac{c_{b2}\rho}{\sigma}\partial_i\tilde{\nu}\partial_i\tilde{\nu} + P - D \end{array}$$

where the Reynolds stress, τ_{ij}, is

$$\tau_{ij} = 2(\mu + \mu_t)\epsilon_{ij}, \qquad \epsilon_{ij} = \frac{1}{2}(\partial_i u_j + \partial_j u_i) - \frac{1}{3}\partial_k u_k \delta_{ij}.$$

μ is the laminar dynamic viscosity and the eddy viscosity, μ_t, is

$$\mu_t = \begin{cases} \rho\tilde{\nu} f_{v1} & \tilde{\nu} \geq 0 \\ 0 & \tilde{\nu} < 0 \end{cases}, \qquad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi \equiv \frac{\tilde{\nu}}{\nu}.$$

The heat flux, q_i, is given by

$$q_i = (k + k_t)\partial_i T, \qquad k = C_p \mu / Pr, \qquad k_t = C_p \mu_t / Pr_t.$$

The diffusivity for the SA working variable, $\tilde{\nu}$, is given by η/σ, where

$$\eta = \begin{cases} \mu(1+\chi) & \chi \geq 0 \\ \mu(1 + \chi + \chi^2) & \chi < 0 \end{cases}$$

The production term, P, is

$$P = \begin{cases} c_{b1}\tilde{S}\rho\tilde{\nu} & \chi \geq 0 \\ c_{b1} S \rho \tilde{\nu} g_n & \chi < 0 \end{cases}, \qquad \tilde{S} = \begin{cases} S + \bar{S} & \bar{S} \geq -c_{v2} S \\ S + \dfrac{S(c_{v2}^2 S + c_{v3}\bar{S})}{(c_{v3} - 2c_{v2})S - \bar{S}} & \bar{S} < -c_{v2} S \end{cases},$$

where $S = \sqrt{2\Omega_{ij}\Omega_{ij}}$ is the vorticity magnitude and

$$g_n = 1 - \frac{1000\chi^2}{1 + \chi^2}$$

Also,

$$\bar{S} = \frac{\tilde{\nu} f_{v2}}{\kappa^2 d^2}, \qquad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}.$$

The destruction term, D, is given by

$$D = \begin{cases} c_{w1} f_w \dfrac{\rho\tilde{\nu}^2}{d^2} & \chi \geq 0 \\ -c_{w1}\dfrac{\rho\tilde{\nu}^2}{d^2} & \chi < 0 \end{cases}$$

where

$$f_w = g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{1/6}, \qquad g = r + c_{w2}(r^6 - r), \qquad r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}.$$

The closure coefficients are

$$\begin{array}{rclrcl} c_{b1} &=& 0.1355 & c_{w2} &=& 0.3 \\ \sigma &=& 2/3 & c_{w3} &=& 2 \\ c_{b2} &=& 0.622 & c_{v1} &=& 7.1 \\ \kappa &=& 0.41 & c_{v2} &=& 0.7 \\ c_{w1} &=& \dfrac{c_{b1}}{\kappa^2} + \dfrac{1+c_{b2}}{\sigma} & c_{v3} &=& 0.9 \\ Pr_t &=& 0.9 & & & \end{array}$$

In discretizing the $\tilde{\nu}$ viscous term, we have to be aware that we store $\rho\tilde{\nu}$, so that

$$F_{\tilde{\nu}i}^v = \frac{\eta}{\sigma}\partial_i\tilde{\nu} = -\frac{\eta\tilde{\nu}}{\rho\sigma}\partial_i\rho + \frac{\eta}{\rho\sigma}\partial_i(\rho\tilde{\nu}),$$

from which the contributions to the diffusion matrix can be read off directly.

The SA working variable, $\tilde{\nu}$, will generally be orders of magnitude smaller than the other state components. Scaling or "non-dimensionalization" of $\tilde{\nu}$ is used to make the stored values of $\tilde{\nu}$ closer to the other state components, which helps during the discrete linear solves and when using finite residual convergence tolerances. Instead of $\tilde{\nu}$ we store $\tilde{\nu}'$, given by

$$\tilde{\nu}' = \frac{\tilde{\nu}}{ND},$$

where ND is a scaling factor, usually on the order of $\nu\kappa_{\mathrm{SA}}$, where ν is the laminar kinematic viscosity. κ_{SA} is a user-prescribed factor, typically 100 or 1000, that makes $\tilde{\nu}'$ on the order of unity. The associated changes in the above expressions are obtained by re-writing all of them in terms of $\tilde{\nu}'$. In addition, the SA equation is divided by ND,

$$\partial_t(\rho\tilde{\nu}') + \partial_i(\rho u_i\tilde{\nu}') \;=\; \partial_i\left(\frac{\eta}{\sigma}\partial_i\tilde{\nu}'\right) + ND\frac{c_{b2}\rho}{\sigma}\partial_i\tilde{\nu}'\partial_i\tilde{\nu}' + \frac{P-D}{ND}.$$

This has the effect of bringing the SA equation residual to the same order as that of the other equations.

B Unsteady Adaptive Strategy

This appendix describes the merit-function-based adaptive strategy used to adapt the spatial and temporal discretizations in an unsteady simulation. This strategy is geared for a slab-based temporal discretization (e.g. DG-in-time), where the size of the time slabs can vary in time, and for dynamic hanging-node or order refinement in space.

B.1 Greedy Fixed Growth Selection Algorithm

The error-addressed versus cost-added figure of merit is used in a fixed-growth adaptive strategy in which some combination of time slabs and space-time elements are marked for coarsening or refinement. The user specifies the fraction of degrees of freedom to be coarsened (f^{coarsen}), as well as the growth factor f^{growth}, which dictates the total number of degrees of freedom in the next mesh as $D^{\text{next}} = f^{\text{growth}} D^{\text{current}}$ (where D^{current} is the number of current degrees of freedom). The coarsening and refinement budgets are then

$$\begin{aligned} B^{\text{coarsen}} &= f^{\text{coarsen}} D^{\text{current}}, \\ B^{\text{refine}} &= (f^{\text{growth}} - 1) D^{\text{current}} + B^{\text{coarsen}}. \end{aligned}$$

In practice, we typically use a coarsening fraction of $\sim 5\%$ ($f^{\text{coarsen}} = 0.05$) and a growth factor of 1.30-1.35. With these budgets defined, the following algorithm is then used to decide which space-time elements or time slabs to adapt:

1. **Sort**
   Sort all space-time elements and time slabs based on the figure of merit: the amount of output error addressed divided by the degrees of freedom added if the element/slab were to be refined. For temporal refinement, the latter is approximated as the degrees of freedom in the targeted slab k, $\text{dof}_k \equiv \sum_e \text{dof}(p_{e,k})$, and for spatial refinement as the number of additional degrees of freedom $\text{dof}(p_{e,k}+1) - \text{dof}(p_{e,k})$ associated with an order increase of element e, k.

2. **Coarsen**
   Set coarsening degree-of-freedom tally to zero. Choose an unmarked space-time element or time slab with the lowest merit function. If a time slab was chosen, mark it for a factor of 2 coarsening and add $0.5\,\text{dof}_k$ to the coarsening tally. If a space-time element was chosen, mark it for an order decrement and add $\text{dof}(p_{e,k}) - \text{dof}(p_{e,k}-1)$ to the coarsening tally. If the tally meets or exceeds the coarsening budget, B^{coarsen}, stop. Otherwise choose the next space-time element or time slab and repeat.

3. **Refine**
   Set refinement degree-of-freedom tally to zero. Choose an unmarked space-time element or time slab with the highest merit function. If a time slab was chosen, mark it for a factor of 0.5 refinement and add dof_k to the refinement tally. If a space-time element was chosen, mark it for an order increment and add $\text{dof}(p_{e,k}+1) - \text{dof}(p_{e,k})$ to the refinement tally. If the refinement budget, B^{refine}, is met or exceeded, stop. Else, choose the next element or time slab.

B.2 Temporal-Mesh Optimization

When we say "factor of 2 coarsening" and "factor of 0.5 refinement", this essentially means that the width of a slab marked for coarsening will be doubled, while the width of a slab marked for refinement will be halved. If only refinement occurs, this is exactly the case, as each marked slab will be perfectly bisected; however, when coarsening also occurs, the

boundaries of a coarsened slab will typically encroach on those of the neighboring slabs, and the entire temporal grid must be shuffled to make room for the coarsened slab. We shuffle time slabs using one-dimensional metric-based meshing, as described below.

1. For each time slab k, define $\widetilde{\Delta t}_k^{\text{desired}} = c\Delta t_k^{\text{current}}$ where c is 0.5 for refinement, 2 for coarsening, and 1 if the time slab is not marked. $\widetilde{\Delta t}_k^{\text{desired}}$ represents the new time step size that we desire on the current time slab k. The given choices of c are consistent with the choices made for the degree-of-freedom counts above.

2. Define $N^{\text{desired}} = \lceil \sum_k (\Delta t_k^{\text{current}} / \widetilde{\Delta t}_k^{\text{desired}}) \rceil$, where $\lceil\ \rceil$ is the greatest integer (ceiling) function. This will be the total number of time slabs on the new temporal mesh. Note that the ceiling function ensures that this number is an integer.

3. To ensure that the desired time step size is consistent with this total number of time steps, define the new desired time step size as

$$\Delta t_k^{\text{desired}} = \widetilde{\Delta t}_k^{\text{desired}} \frac{\sum_k (\Delta t_k^{\text{current}} / \widetilde{\Delta t}_k^{\text{desired}})}{N^{\text{desired}}}.$$

4. Finally, define a function $n(t)$ that is piecewise-constant over the current time slabs, and that takes on the value $1/\Delta t_k^{\text{desired}}$ on each current slab k. Define the new time slab breakpoints as times t_l where $\int_0^{t_l} n(t)dt$ is an integer.

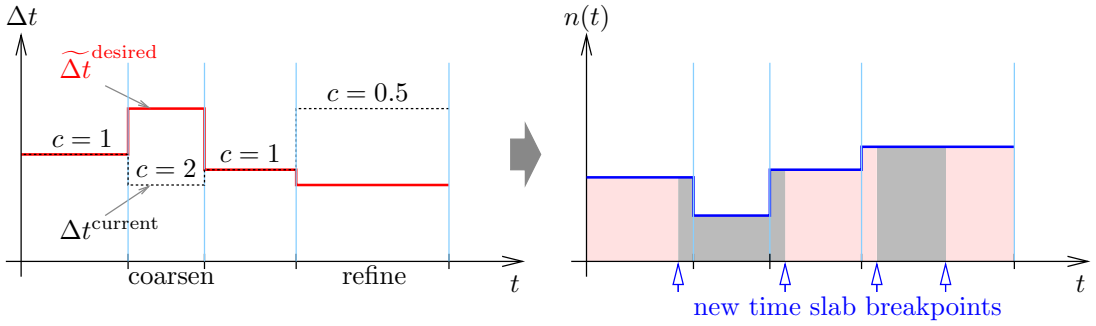


Figure 61: Demonstration of the one-dimensional remeshing algorithm used to define new time slab breakpoints.